FONDAMENTA

Parte IV

Kernel

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.
<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>
<ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/>
<ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/>
La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.
<http://www.pluto.linux.it/ildp/AppuntiLinux/>
<http://ildp.psy.unipd.it/AppuntiLinux/>
<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>
<http://linux.calion.com/pluto/ildp/AppuntiLinux/>

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 è solitamente `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.

15.1 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 diversi gestiscono diversamente lo stesso tipo di unità periferica. A causa della frammentazione in diversi 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 una 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 diversi modi differenti e conflittuali, ma si tratta di situazioni rare.

15.2 Ricompilazione del kernel

Le distribuzioni di 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 installa un tipico programma distribuito in forma sorgente, come descritto nel capitolo 12.

15.3 Kernel monolitico

Il procedimento descritto nelle sezioni seguenti serve per generare un *kernel* monolitico, cioè un *kernel* in un unico file.

15.3.1 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.

15.3.2 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.

15.3.3 Collocazione dei sorgenti e link 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 *link* simbolico che permetta di raggiungere i sorgenti nel modo prestabilito.
Occorre inoltre verificare che tre *link* 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.

15.3.4 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 precedentemente impostata, e alla fine la nuova configurazione viene salvata nello stesso file. Di conseguenza, ripetendo il procedimento **`make config`**, le scelte predefinite corrisponderanno alle scelte 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 Window System.

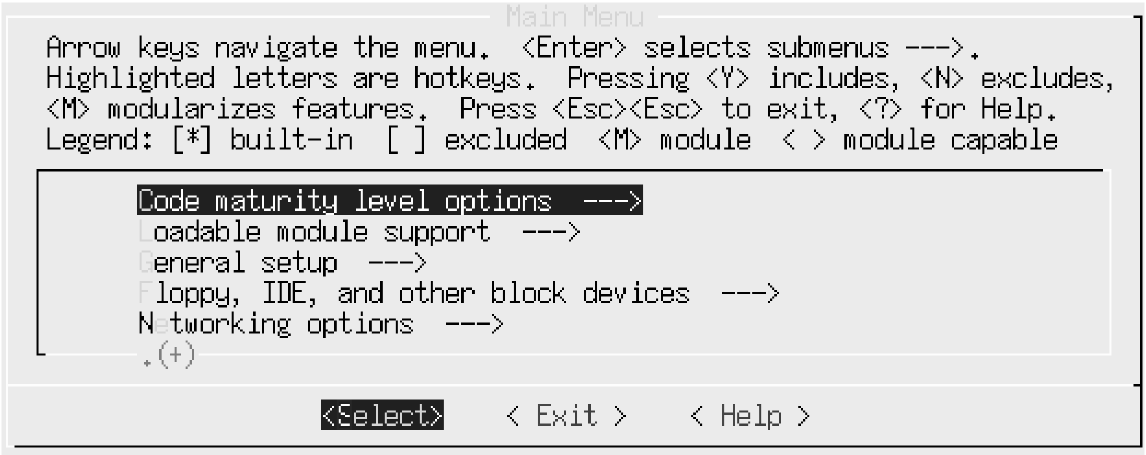

Figura 15.1: Il menu principale della configurazione del *kernel* attraverso il comando **`make menuconfig`**.

15.3.5 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 tranquillamente avviare 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 naturalmente vale 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, 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 sulla directory radice o in `/boot/` e rinominarlo `vmlinuz` (come di consueto), oltre che ripetere l'installazione del sistema di caricamento LILO.

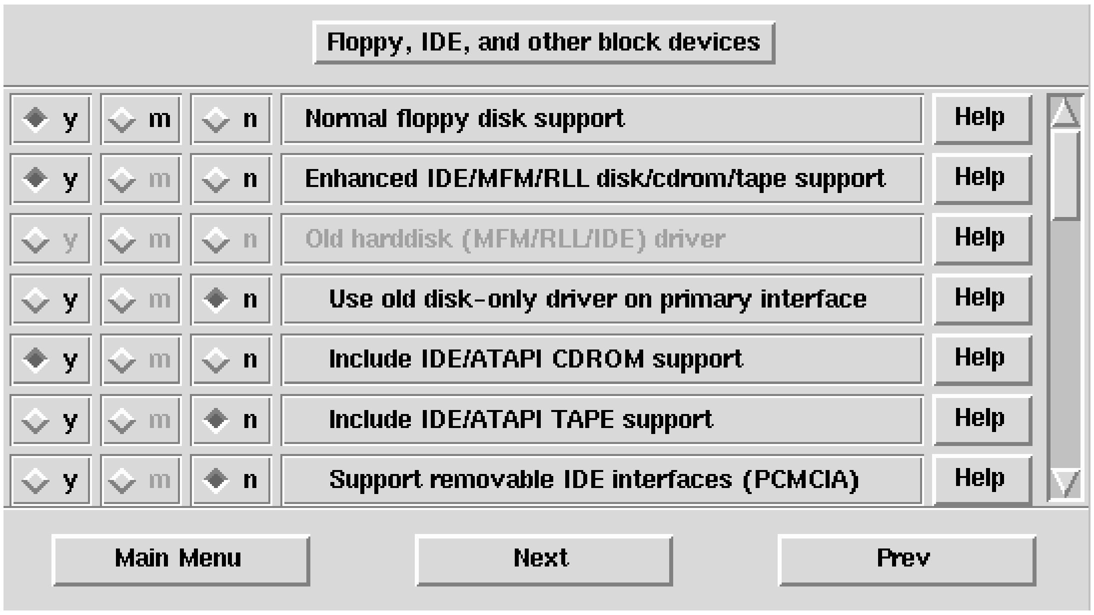

Figura 15.2: Uno dei menu della configurazione del *kernel* attraverso il comando **`make xconfig`**.

15.3.6 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 *boot*.[1]

```
# cp /usr/src/linux/arch/i386/boot/zImage /dev/fd0
```

Per utilizzare correttamente questo dischetto di *boot* è molto probabile che si debba intervenire prima con il programma **`rdev`** (11.1.1).

15.4 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.

15.4.1 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*: `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 Linux che si ha installato, è ragionevole supporre che questi programmi di gestione dei moduli siano già stati installati correttamente.

15.4.2 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
```

[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 eseguire il *mount* di un dischetto del genere.

Quello che si ottiene sono una serie di file oggetto, il cui nome ha una 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.

15.5 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
Loadable module support
General setup
Plug and Play support
Floppy, IDE, and other block devices
Networking options
SCSI support
Network device support
ISDN subsystem
CD-ROM drivers (not for SCSI or IDE/ATAPI drives)
Filesystems
Native Language Support
Character 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 computer i386 o superiore con un BUS PCI;
- un disco fisso IDE standard;
- un CD-ROM ATAPI, cioè connesso allo stesso *controller* IDE del disco fisso;
- una 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 con un *Internet Service Provider*;
- una scheda di rete Ethernet compatibile 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.

15.5.1 Code maturity level options

- *Prompt for development and/or incomplete code/drivers* [N]

  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.

15.5.2 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 *daemon* necessario all'utilizzo dei moduli (in pratica si tratta di **`kerneld`**). In generale, se si utilizzano i moduli, conviene attivare questa opzione.

15.5.3 General setup

- *Kernel 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 computer 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.
- *Networking support* [Y]
  Permette di attivare la gestione della rete. È importante attivare questa opzione anche se non si intende usare il proprio computer in alcuna rete: alcuni programmi rischierebbero di non funzionare.
- *Limit memory to low 16MB* [N]
  Questa opzione deve essere attivata solo nel caso si siano riscontrati problemi incomprensibili a carico della memoria e si intenda tentare di fare funzionare il sistema utilizzando solo i primi 16MB di memoria RAM. Generalmente non si seleziona questa opzione.
- *PCI support* [Y]
  Se il proprio computer ha un *bus* PCI, è necessario attivare questa opzione, altrimenti le schede inserite negli *slot* 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 BIOS support* [Y]
    Se è stata attivata l'opzione *PCI support*, si può attivare anche questa in modo da utilizzare il BIOS per individuare i dispositivi PCI e determinare la loro configurazione.
    Le prime schede madri che incorporavano un *bus* PCI avevano un BIOS imperfetto che potrebbe causare dei problemi attivando questa opzione.
  - *PCI direct access support*
    Quando non è possibile utilizzare il BIOS per individuare i dispositivi PCI, si può tentare di attivare questa opzione che salta le funzioni del BIOS stesso. Se viene attivata assieme all'opzione *PCI BIOS support*, l'utilizzo del BIOS ha la precedenza e questa opzione viene praticamente ignorata.
- *MCA support* [N]
  Questa opzione permette di attivare la gestione del *bus* MCA (*Micro Channel Architecture*), utilizzato in alcuni computer 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. Vale la pena di attivare questa opzione, soprattutto in considerazione del fatto che alcuni programmi non sono in grado di funzionare se non è disponibile l'IPC.
- *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. Linux ha utilizzato i formati **`a.out`**, ma ormai, quasi tutto tende a essere convertito nel formato ELF.
  Questa opzione permette di poter 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, conviene attivare questa opzione.

- *Kernel support for JAVA binaries* [Y]

  L'attivazione di questa opzione permette l'avvio automatico dell'interprete del formato *Java bytecode* quando si tenda di avviare un file compilato in Java.

- *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.

- *Processor type* [386]

  È possibile ottimizzare il funzionamento del *kernel* fornendo l'indicazione del tipo di microprocessore installato. Se si intende ottenere un *kernel* che possa funzionare su un numero più ampio possibile di computer, conviene specificare **`386`**; in tutti gli altri casi è meglio specificare il tipo esatto.

  ```
  (X) 386
  ( ) 486
  ( ) Pentium
  ( ) PPro
  ```

- *Video mode selection support*

  Questa opzione abilita la selezione di una particolare modalità video, legata alla gestione a caratteri, attraverso una istruzione da inviare al *kernel* nella fase di caricamento (*bootprompt*).

- *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. Quest'opzione deve essere attivata nel caso si utilizzi un PC e anche in alcuni computer Alpha.

15.5.4 Plug and Play support

La gestione del *Plug and Play* permette al *kernel* di configurare automaticamente alcuni dispositivi.

- *Plug and Play support* [y]

  Attivando questa opzione si abilita la gestione del *Plug and 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.

15.5.5 Floppy, IDE, and other 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 tipico dispositivo a blocchi è una unità a disco.

- *Normal 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 support* [Y]

  Questa opzione permette di utilizzare la maggior parte di unità IDE/EIDE (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 normali dischi fissi. Può essere utilizzata questa opzione 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 pratica, può essere usata sia come opzione alternativa a *Enhanced IDE/MFM/RLL disk/cdrom/tape support* che come opzione aggiuntiva.

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 IDE/ATAPI, cioè connesso alla interfaccia IDE/EIDE, si deve attivare questa opzione.

- *Include IDE/ATAPI TAPE support* [y]

  Se si utilizza una unità a nastro IDE/ATAPI, cioè connessa alla interfaccia IDE/EIDE, si deve attivare questa opzione.

- *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 equivalenti unità IDE-ATAPI. Ciò può essere utile quando si dispone di una nuova unità IDE-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 scheda madre o di una scheda che incorpora un *controller* IDE gestito dall'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 alla identificazione e correzione automatica del problema.

- *CMD640 enhanced support*

  Questa opzione permette di includere la gestione delle modalità PIO sui *controller* IDE con integrato CMD640. Se si ha un tale *controller* e il BIOS non è in grado di gestire automaticamente questo meccanismo, conviene attivare l'opzione, altrimenti no.

- *RZ1000 chipset bugfix/support*

  Se si dispone di una scheda madre o di una 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 alla identificazione e correzione automatica del problema.

- *Intel 82371 PIIX (Triton I/II) DMA support*

  Se si dispone di una scheda madre PCI che include l'integrato Intel 430FX PCI Triton e si utilizza un disco fisso IDE/EIDE, conviene attivare questa opzione in modo da permettere l'utilizzo del *bus-mastering DMA data transfers*.

- *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 Linux di accedere alla terza o quarta unità disco.

- *Loopback device support* [Y]

  Attivando questa opzione, il *kernel* permetterà di eseguire il *mount* di 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.

- *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 una unica partizione).

- *RAM disk support* [Y]

  Attivando questa opzione diventa possibile la gestione di dischi RAM. Di solito, i dischetti utilizzati per installare 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 *RAM disk* iniziale, cioè attivato dallo stesso *boot loader* (LILO oppure Loadlin).

- *XT harddisk support* [N]

  Questa opzione permette di utilizzare i vecchi *controller* a 8 bit dei primi IBM XT. È molto improbabile che un computer in grado di far funzionare Linux abbia a disposizione tali vecchie schede *controller*.

- *SyQuest EZ parallel port disk support*

  Attivando questa opzione si abilita l'utilizzo di una unità SyQuest EZ esterna, connessa alla porta parallela.

15.5.6 Networking options

La configurazione dei servizi di rete è importante anche se il proprio computer è isolato.

- *Kernel/User network link driver*

  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 *daemon* **arpd** che aiuta a mantenere un *ARP cache* interno (una mappa delle corrispondenze tra indirizzi IP e indirizzi hardware della rete locale) occorre attivare questa opzione.

- *Routing messages*

  Questa opzione può essere attivata solo se in precedenza si attiva anche *Kernel/User network link driver*. 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 computer che protegge una rete locale dal resto del mondo: tutto il traffico da e per i computer della rete locale viene filtrato dal *firewall*. Se si vuole configurare il proprio sistema 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 i computer della rete locale possono comunicare con quelli della rete esterna, ma i computer 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*

  Rispondendo affermativamente si consente l'attribuzione di indirizzi diversi alla stessa unità fisica di rete. Si tratta di una funzionalità speciale.

- *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 il computer non è connesso a una rete. Ciò attiverà il cosiddetto *loopback* che in pratica permette di comunicare con se stessi. Ma a perte questo, la cosa più importante è la possibilità di accedere attraverso la linea telefonica a un nodo di Internet.

- *IP: forwarding/gatewaying* [Y]

  Se si vuole utilizzare il computer come *router* per l'instradamento di pacchetti tra due reti locali distinte, ovvero tra la rete Internet e la rete locale, si deve rispondere affermativamente alla domanda.

  Nei *kernel* più recenti, questa opzione è stata eliminata e di conseguenza, le funzionalità di inoltro sono sempre attivate.

- *IP: multicasting* [y]

  Si tratta del codice necessario per raggiungere diversi computer di una rete con un unico indirizzo. Se si intende usare **gated**, il *daemon* che aggiorna le tabelle di instradamento (*routing*) del proprio computer, 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: firewalling* [y]

  Se si vuole configurare il proprio computer come *firewall* per proteggere una rete locale basata su TCP/IP, si deve rispondere affermativamente a questa domanda. Per la gestione del *firewall* è necessaria anche l'*utility* **`ipfwadm`** per consentire il blocco selettivo del traffico di pacchetti basato sul tipo, sull'origine e sulla destinazione. È necessario abilitare questa funzione se si intende utilizzare il mascheramento IP (*IP masquerading*) che significa che i computer della rete locale possono comunicare con quelli della rete esterna, ma i computer 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.

- *IP: firewall packet netlink device*

  Questa opzione può essere attivata solo se in precedenza si attiva anche *Kernel/User network link driver*. 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: 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 *daemon* **`klogd`** (26.1.6) che si occupa del *log* dei messaggi del *kernel*.

- *IP: masquerading* [y]

  Quando si costruisce un *kernel* per un computer 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 i computer della rete locale non siano visibili all'esterno. È poi compito del *firewall* dirigere i pacchetti di dati al computer corretto nella rete locale. All'esterno, sembra che tutto il traffico sia svolto solo dal computer *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: transparent proxy support*

  Attivando questa opzione si fa in modo che i computer di una rete locale, protetta dal *proxy* per il quale si compila il *kernel*, comunichino come se avessero un accesso diretto alla rete esterna, utilizzando gli indirizzi reali dei computer remoti. In pratica, i computer della rete locale possono comportarsi come se il *proxy* non ci fosse.

- *IP: always defragment*

  Questa opzione permette di gestire la ricomposizione automatica dei pacchetti IP frammentati.

  Quando si utilizza il proprio computer 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 particolarmente importante la ricomposizione dei pacchetti se si utilizza il sistema del mascheramento IP. (*IP: masquerading*).

  Non si deve attivare questa opzione in un qualsiasi computer che svolge funzioni di *router* e tanto meno in un normale computer connesso alla rete senza particolari ruoli.

- *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 computer 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: optimize as router not host* [N]

  Permette di ottimizzare il funzionamento del computer che svolge esclusivamente la funzione di *router*. Non deve essere attivata questa opzione nel caso di un computer 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 computer presente in un punto diverso da quello in cui si trova fisicamente. Ciò 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: multicast routing* [N]

  Attivando questa opzione, si abilita la possibilità di eseguire il *routing* di pacchetti destinati a più indirizzi (*multicasting*).

- *IP: aliasing support* [N]

  Attraverso l'attivazione di questa opzione, è possibile assegnare a una sola interfaccia di rete, più di un indirizzo IP.

- *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: PC/TCP compatibility mode* [N]

  Se sono state riscontrate difficoltà per connettersi attraverso un programma di **telnet** da un computer 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 NCSA Telnet, si può leggere la sezione 79.3 o eventualmente il file `.../Documentation/networking/ncsa-telnet`.

- *IP: reverse ARP* [N]

  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 computer 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 computer nella rete è data dal protocollo BOOTP e dal suo successore DHCP.

- *IP: Path MTU Discovery (normally enabled)* [Y]

  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 computer *host* gradiscono blocchi piccoli, si aggiusta questa dimensione dei blocchi di dati a valori minori.

  Alcune versioni Dos di NCSA Telnet, 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.

- *IP: Drop source routed frames* [Y]

  Di solito, quando viene prodotto un *frame* IP (cioè un pacchetto), viene specificato solo la sua destinazione e i vari *host* incontrati lungo la sua strada si occuperanno dell'instradamento (*routing*) 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 è adesso di decidere se si intendono rispettare queste richieste di instradamento quando arrivano questi 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 derivare.

- *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 computer che abbiano almeno 16Mb di memoria centrale.

- *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 i computer 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 il computer Linux agisca come *server* sia per i file che per la stampa, per i computer MAC, occorre il pacchetto Nettalk.

- *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 il computer Linux con la radio amatoriale.

  - *AX.25 DAMA Slave support*
  - *Amateur Radio NET/ROM*
  - *Amateur Radio X.25 PLP (Rose)*

- *Bridging*

  Questa opzione permette di utilizzare il proprio computer come un *bridge* Ethernet. In tal modo, i segmenti di rete fisica che vengono connessi appariranno come una unica rete.

15.5.7 SCSI support

- *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 una 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 CD *jukebox*.

- *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.

  ```
  [ ] 7000FASST SCSI support
  [ ] Adaptec AHA152X/2825 support
  [ ] Adaptec AHA1542 support
  [ ] Adaptec AHA1740 support
  [ ] Adaptec AIC7xxx support
  [ ] AdvanSys SCSI support
  [ ] Always IN2000 SCSI support
  [ ] AM53/79C974 PCI SCSI support
  [ ] BusLogic SCSI support
  ```

```
[ ] DTC3180/3280 SCSI support
[ ] EATA-DMA (DPT, NEC, AT&T, SNI, AST, Olivetti, Alphatronix) support
[ ] EATA-PIO (old DPT PM2001, PM2012A) support
[ ] EATA ISA/EISA/PCI (DPT and generic EATA/DMA-compliant boards) support
[ ] Future Domain 16xx SCSI support
[ ] Generic NCR5380/53c400 SCSI support
[ ] NCR53c406a SCSI support
[ ] NCR53c7,8xx SCSI support
[ ] NCR53C8XX SCSI support
[ ] IBMMCA SCSI support
[ ] IOMEGA Parallel Port ZIP drive SCSI support
[ ] PAS16 SCSI support
[ ] Qlogic FAS SCSI support
[ ] Qlogic ISP SCSI support
[ ] Seagate ST-02 and Future Domain TMC-8xx SCSI support
[ ] Tekram DC-390(T) (AMD PCscsi) SCSI support
[ ] Trantor T128/T128F/T228 SCSI support
[ ] UltraStor 14F/34F support
[ ] UltraStor SCSI support
```

15.5.8 Network device support

- *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 computer Linux a una rete, occorre una scheda di rete; se si vuole connettere il proprio computer Linux a un nodo 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 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 linea seriale (SLIP o PPP) che consente un migliore adattamento alla linea e di conseguenza migliora le prestazioni della 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. **`10-base-2`**, **`10-base-T`** e **`100-base-...`** sono dei tipi comuni di connessioni Ethernet. Se il proprio computer Linux viene connesso a una rete Ethernet e di conseguenza si ha installato 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]

    ```
    [ ] Cabletron E21xx support
    [ ] DEPCA, DE10x, DE200, DE201, DE202, DE422 support
    [ ] EtherWORKS 3 (DE203, DE204, DE205) support
    [ ] EtherExpress 16 support
    [ ] HP PCLAN+ (27247B and 27252A) support
    ```

[2]Volendo seguire l'esempio iniziale, occorre selezionare il tipo NE2000.

```
[ ] HP PCLAN (27245 and other 27xxx series) support
[ ] HP 10/100VG PCLAN (ISA, EISA, PCI) support
[*] NE2000/NE1000 support
[ ] SK_G16 support
```

  – *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]
- *PLIP (parallel port) support*

  Il protocollo PLIP (*Paralel Line Internet Protocol*) è usato per creare una mini rete di due computer. Le porte parallele vengono connesse usando un cavo **`null printer`** o **`Turbo 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 computer Linux e un computer Dos che utilizza un *packet driver* adatto come quello della Crynwr (79.1).

  Dal momento che di solito si ha a disposizione una sola porta parallela e con quella si intende utilizzare la stampante oltre che una eventuale connessione PLIP, per evitare conflitti è opportuno compilare sia il supporto per il protocollo PLIP che per la stampa, come modulo, **`M`**, in modo da poterli caricare e scaricare a seconda delle necessità. Essendo fonte di conflitti, sarebbe meglio fare a meno di includere questa funzione se non è strettamente necessaria.

- *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`**.

- *Radio network interfaces*

  La selezione di questa opzione permette di indicare delle interfacce di rete che utilizzano un collegamento in radiofrequenza.

- *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 Internet, oppure a qualche altro computer Unix locale, oppure se si vuole configurare il proprio computer 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 **`nullmodem`**). 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.

- *Keepalive and linefill* [y]

  Permette di aggiungere delle capacità addizionali al *driver* SLIP di 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.

- *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.

- *WAN drivers* [N]

  Selezionando questa opzione, è possibile scegliere un *driver* per la connessione del proprio computer come un *router* WAN.

- *LAPB over Ethernet drivers* [N]

  Questa opzione permette di attivare il protocollo LAPB per ottenere una connessione *point-to-point* con un altro computer connesso alla propria rete locale.

- *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 normale linea telefonica utilizzata attraverso modem.

15.5.9 ISDN subsystem

- *ISDN support*

  Se si utilizza una connessione telefonica ISDN si deve attivare questa opzione.

- *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 *daemon* **`pppd`**: **`ipppd`**.

- *Use VJ-compression with syncronous PPP*

  Questa opzione permette di abilitare la compressione di intestazione Van Jacobson per le connessioni sincrone PPP.

- *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.

- *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. Nelle versioni più vecchie del *kernel* era presente l'opzione **`CONFIG_ISDN_DRV_TELES`** che si riferiva a una parte delle interfacce comprese in questa voce.

- *AVM-B1 with CAPI2.0 support*

  Questa opzione permette di attivare la gestione delle schede ISDN AVM B1.

15.5.10 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
  [ ] MicroSolutions backpack 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
  [ ] Soft configurable cdrom interface card support
  [ ] Sony CDU31A/CDU33A CDROM support
  [ ] Sony CDU535 CDROM support
  ```

15.5.11 Filesystems

- *Quota support*

  Questa opzione permette di attivare la gestione della limitazione di spazio a disposizione nel *filesystem*, concesso a ogni utente. Questo spazio limitato messo a disposizione si chiama *diskquota* e per la sua gestione occorrono delle *utility* aggiuntive. La gestione di questo meccanismo è utile solo per un sistema multiutente.

- *Mandatory lock support* [N]

  L'attivazione di questa opzione abilita la gestione di un sistema di *lock* dei file più sicuro di quello consueto. Il *lock* dei file è il meccanismo usato per prevenire conflitti nel momento in cui più processi elaborativi utilizzano lo stesso file in scrittura nello stesso momento.

  Se si attiva questa opzione, occorre utilizzare i *daemon* per la gestione dei *filesystem* aggiornati per questa modalità di accesso ai file. Fino a che non si è sicuri di avere tutto il software necessario correttamente aggiornato, conviene non attivare questa opzione.

- *Minix fs support* [Y]

  Il *filesystem* Minix può essere considerato superato dal momento che Linux ha a disposizione il tipo Ext2, ma è ancora diffuso il suo utilizzo per i dischetti. In particolare, le immagini dei dischetti per l'avvio e l'installazione di Linux della distribuzione Slackware sono in formato Minix. In generale, conviene attivare questa opzione, in modo da poter utilizzare questo tipo di *filesystem*.

- *Second extended fs support* [Y]

  Questa opzione permette di attivare la gestione del *filesystem* Ext2 che è lo standard per Linux. Sotto questo aspetto, è praticamente obbligatorio attivare questa opzione.

- *ISO9660 cdrom filesystem support* [Y]

  Questa opzione permette di eseguire il *mount* di 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]

- *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 eseguire il *mount* di un *filesystem* Dos-FAT normale, occorre attivare questa opzione insieme a quella precedente: *DOS FAT fs support*.

- *umsdos: Unix like fs on top of std MSDOS FAT fs* [Y]

  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, occorre attivare questa opzione, e insieme a questa, anche *DOS FAT fs support* e *MSDOS fs support*.

- *VFAT (Windows-95) fs support* [Y]

  Per poter eseguire il *mount* di 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.

- */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.

- *NFS filesystem support* [Y]

  Se il proprio computer è connesso a una rete di qualunque tipo (SLIP, PLIP, PPP o Ethernet), attivando questa opzione è possibile eseguire il *mount* di parte del *filesystem* di un altro computer che consente la condivisione dei suoi dati attraverso il protocollo NFS. Il computer 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.

  Per poter utilizzare il protocollo NFS occorre ovviamente aver attivato la gestione dei servizi di rete con tutto ciò che serve per permettere al proprio computer di comunicare attraverso il tipo di rete che si dispone.

  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.

  - *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 computer 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 *daemon* **`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, NT e Lan Manager per le comunicazioni attraverso le reti Ethernet. Attivando questa opzione, è possibile eseguire il *mount* di directory condivise da computer 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 computer 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 aver attivato anche tutto quello che serve ad avviare i 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.

- *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.

- *System V and Coherent filesystem support*

  L'attivazione di questa opzione permette di eseguire il *mount* di *filesystem* del tipo utilizzato da alcuni Unix commerciali che funzionano su computer basati su microprocessori Intel. Si tratta in particolare di SCO, Xenix e Coherent.

- *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* (15.5.5 in modo da poter accedere alle immagini di dischi utilizzate con un eventuale emulatore Amiga. I dischetti veri e propri, non possono essere utilizzati.

- *ROM filesystem support* [N]

  Si tratta della gestione di un *filesystem* a sola lettura e di dimensioni molto ridotte, pensato principalmente per un *ramdisk* iniziale (**`initrd`**), ma potrebbe essere utilizzato anche per altri tipi di supporti a sola lettura.

- *Kernel automounter support* [N]

  Permette la gestione interna al *kernel* del *mount* automatico di *filesystem* remoti. Generalmente si utilizza un *daemon* per questo scopo, mentre in questo caso si vuole che sia proprio il *kernel* a occuparsene. Se si attiva questa opzione, per gestire il sistema sono poi necessari altri programmi di contorno. Trattandosi del *mount* di *filesystem* remoti, occorre ovviamente attivare anche la gestione di questi attraverso l'opzione *NFS filesystem support*.

- *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).

- *Macintosh partition map support* [N]

15.5.12 Native Language Support

In questa sezione della configurazione viene definito tutto ciò che riguarda la nazionalità. In particolare, il set di caratteri: quello più adatto all'Italia e a buona parte dell'Europa è ISO-8859-1, corrispondente a ISO-Latin-1.

15.5.13 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 generalmente veniva sottintesa. Si tratta della esplicita richiesta 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 Linux per scopi specifici in cui l'interazione con terminale non avviene.

- *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 sia il terminale virtuale su cui si sta lavorando in quel momento.

- *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.

- *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*.

- *Parallel printer support* [Y]

  Questa opzione permette di attivare la gestione delle porte parallele, in modo da consentirne l'utilizzo per la stampa. Di solito, questa opzione deve essere attivata, a meno di trovarsi nella necessità di utilizzare l'unica porta parallela per una connessione PLIP. Un'altra alternativa possibile, è quella di compilare questa gestione della porta parallela per la stampa come modulo, in modo da poterlo disattivare e riattivare durante il funzionamento del sistema.[3]

  - *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. Attivando l'opzione si ottiene la possibilità di specificare il tipo di mouse.

[3]Nei nuovi *kernel* si sta sperimentando la possibilità di inserire sia la gestione della stampa che di altre attività senza dovere necessariamente intervenire con i moduli.

```
[ ] 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 una unità a nastro QIC-02 che non è connessa attraverso una scheda SCSI, si deve attivare questa opzione.

- *Ftape (QIC-80/Travan) support*

  L'attivazione di questa opzione permette l'utilizzo di unità a nastro connesse al *controller* dei dischetti e di altre unità connesse attraverso un *controller* proprietario. In questo secondo caso, occorre modificare il file `.../drivers/char/ftape/Makefile` manualmente prima della compilazione del *kernel*.

- *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 computer 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 computer a batteria, non è utile l'attivazione di questa opzione.

- *Watchdog Timer Support* [N]

  Attivando questa opzione si attiva un sistema, attraverso il quale, alcuni tipi di errori generano il riavvio del sistema. Può essere utile l'attivazione di questa opzione nel caso di un computer connesso a una rete che deve essere in grado di sbloccarsi autonomamente senza l'intervento umano.

- *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.

- */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.

- *PC joystick support* [y]

  Attivando questa opzione, è possibile accedere al *joystick* attraverso un file di dispositivo apposito.

15.5.14 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.

15.5.15 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.

15.6 PLIP e stampa

La gestione della stampa su porta parallela e l'utilizzo di una connessione PLIP sono cose conflittuali. Solitamente, per poter creare un *kernel* adatto alla gestione della porta parallela sia per la stampa che per un collegamento PLIP, occorre attivare la gestione dei moduli (15.5.2) e indicare che il supporto al PLIP (15.5.8) e alla stampante parallela (15.5.13) avvengano attraverso moduli.

Lo sviluppo del *kernel* tende verso una soluzione più semplice del problema, ma per il momento non conviene tentare di incorporare entrambe le gestioni in un unico *kernel* monolitico.

La sezione 58.3 descrive l'utilizzo di questo tipo di connessione.

Parametri di boot

Il *kernel* è in grado di ricevere alcune opzioni in fase di avvio che possono servire per scopi differenti. In particolare, per gestire alcuni dispositivi è necessario informare il *kernel* su alcune loro caratteristiche (tipicamente l'indirizzo di I/O e il livello di IRQ).
Nel capitolo 11 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.

16.1 Parametri di uso generale

Alcuni parametri di avvio non riguardano dispositivi specifici, ma il sistema in generale.

16.2 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. Questi dati possono essere memorizzati all'interno del *kernel* stesso, anche per mezzo del programma **`rdev`**.

- Selezione del *filesystem root*.

  `root=`⟨*dispositivo*⟩

  Permette di specificare un dispositivo differente da quello predefinito per montare il *filesystem root*. 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 root* 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 root* in lettura-scrittura.

  Questa indicazione può essere memorizzata direttamente nel *kernel* attraverso **`rdev`**.

  `rdev -R` ⟨*kernel*⟩ `0`

Esempi

Il *filesystem root* è la seconda partizione del primo disco fisso IDE.

```
root=/dev/hda2
```

Il *filesystem root* viene aperto inizialmente in lettura e scrittura (per qualche ragione).

```
rw
```

16.3 Memoria

- Memoria RAM disponibile.

  `mem=`⟨*dimensione*⟩

  In caso di necessità, permette di definire la dimensione di memoria RAM effettivamente a disposizione. Si può indicare un numero esadecimale nella forma **`0x...`**, oppure un numero decimale normale, eventualmente seguito dalla lettera **`k`**, che sta a indicare Kilobyte, oppure dalla lettera **`M`**, che sta a indicare Megabyte. 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 root* come *ramdisk*.

  `load_ramdisk={0|1}`

  Permette di definire se si vuole caricare il *filesystem root* come *ramdisk*. **`load_ramdisk=1`** significa che si intende attivare un *ramdisk*, mentre assegnando il valore zero ciò non accade. Il valore predefinito è **`load_ramdisk=0`**.

16.4 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 microprocessore i386 e successivi dispongono di una istruzione denominata **`hlt`** che permette loro di entrare in uno stato di riposo in attesa di una 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 (zero corrisponde a infinito ed è di fatto il valore predefinito).

- 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 in modo che il *kernel* non esegua alcun tentativo di identificazione di componenti in quella zona. Di solito, dopo una 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

Riserva gli indirizzi di I/O da 0x300 a 0x340.

```
reserve=0x300,64
```

16.5 Adattatori SCSI

Segue l'elenco di alcuni adattatori SCSI che possono richiedere l'utilizzo di una istruzione di avvio perché vengano riconosciuti correttamente.

- Adaptec AHA-151x, AHA-152x, aic6260, aic6360, SB16-SCSI

  `aha152x=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`[,`⟨*scsi-id*⟩`]]`

- Adaptec AHA-154x

  `aha1542=`⟨*indirizzo-I/O*⟩

- BusLogic

  `BusLogic=`⟨*indirizzo-I/O*⟩

- EATA

  `eata=`⟨*indirizzo-I/O*⟩`[,`⟨*indirizzo-I/O-2*⟩`[,...]]`
- Future Domain TMC-8xx, TMC-950

  `tmc8xx=`⟨*indirizzo-memoria*⟩`,`⟨*livello-IRQ*⟩
- Future Domain TMC-16xx, TMC-3260, AHA-2920

  `fdomain=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩`[,`⟨*scsi-id*⟩`]`
- IOMEGA Parallel Port / ZIP drive

  `ppa=`⟨*indirizzo-I/O*⟩
- NCR-5380

  Le schede che utilizzano questo integrato possono richiedere un indirizzo di I/O, oppure di memoria, un livello di IRQ e un canale DMA. Si possono avere quindi due sintassi per il parametro da passare al *kernel*.

  `ncr5380=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩`,`⟨*canale-DMA*⟩

  `ncr5380=`⟨*indirizzo-memoria*⟩`,`⟨*livello-IRQ*⟩`,`⟨*canale-DMA*⟩

  L'indicazione di un livello di IRQ pari a 255 significa che questo non viene utilizzato; un livello pari a 254 è una richiesta esplicita di autodeterminazione del valore esatto.
- NCR-53c400

  `ncr53c400=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩

  `ncr53c400=`⟨*indirizzo-memoria*⟩`,`⟨*livello-IRQ*⟩

  Per quanto riguarda il livello di IRQ, valgono le stesse considerazioni fatte per il *driver* NCR-5830.
- NCR-53c406a

  `ncr53c406a=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`]`

  Se viene indicato un livello di IRQ pari a zero, viene disabilitato l'uso di questo.
- Pro Audio Spectrum / Studio 16

  `pas16=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩

  Un livello di IRQ pari a 255 indica la scelta di non volerlo utilizzare.
- Seagate ST-0x

  `st0x=`⟨*indirizzo-memoria*⟩`,`⟨*livello-IRQ*⟩
- Trantor T128

  `t128=`⟨*indirizzo-memoria*⟩`,`⟨*livello-IRQ*⟩
- Ultrastor

  `u14-34f=`⟨*indirizzo-I/O*⟩`[,`⟨*indirizzo-I/O-2*⟩`[,...]]`
- Western Digital WD7000

  `wd7000=`⟨*livello-IRQ*⟩`,`⟨*canale-DMA*⟩`,`⟨*indirizzo-I/O*⟩

Esempi

Scheda SCSI Adaptec AHA-1522: indirizzo di I/O 340, livello di IRQ 11, identificativo SCSI 7.

```
aha152x=0x340,11,7
```

Scheda SCSI Adaptec AHA-1542: indirizzo di I/O 330.

```
aha1542=0x330
```

Scheda SCSI Future Domain TMC-800: indirizzo di memoria ca000, livello di IRQ 10.

```
rmc8xx=0xca000,10
```

Scheda SCSI Adaptec AHA-2920: indirizzo di I/O 0xffa0, livello di IRQ 9.

```
fdomain=0xffa0,9
```

Scheda NCR-5380: indirizzo di I/O 350, livello di IRQ 5, canale DMA 3.

```
ncr5380=0x350,5,3
```

16.6 Dischi IDE, XT e simili

I computer i386 e successivi sono generalmente dotati di controller 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*.

16.6.1 Dischi IDE

Nelle sintassi seguenti, **`hdx`** rappresenta un dispositivo corrispondente a un disco fisso, dove *x* è una lettera da **`a`** ad **`h`**. La sigla **`iden`** rappresenta un dispositivo corrispondente a un controller 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.

16.6.2 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=`**.

16.6.3 Vecchi controller XT a 8 bit

- Caratteristiche della scheda *controller*.

  `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

Controller WD1002: indirizzo di I/O 320, livello di IRQ 5, canale DMA 3.

```
xd=2,5,0x320,3
```

16.7 Schede controller per CD-ROM non standard

Rientrano in questa categoria quelle schede di controllo di lettori CD-ROM che non appartengono alla categoria IDE/ATAPI e nemmeno SCSI.

- Aztech

  `aztcd=`⟨*indirizzo-I/O*⟩

- Sony CDU-31A e CDU-33A

  `cdu31a=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`[,PAS]]`

  Se la scheda non utilizza un IRQ, si può indicare uno zero in corrispondenza del suo argomento. Se si tratta di una scheda Pro Audio Spectrum si deve indicare come terzo argomento la stringa **`PAS`**.

- Sony CDU-535

  `sonycd535=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`]`

  Se la scheda non utilizza un IRQ, si può indicare uno zero in corrispondenza del suo argomento.

- GoldStar

  `gscd=`⟨*indirizzo-I/O*⟩

- ISP16

  Si tratta di una interfaccia per l'utilizzo di diversi tipi di lettore CD-ROM. Sono possibili due sintassi.

  `isp16=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`[,`⟨*canale-DMA*⟩`[,`⟨*tipo*⟩`]]]`

  `isp16=noisp16`

  L'ultimo argomento, il tipo, è un nome che rappresenta il tipo di lettore connesso. Può trattarsi di **`Sanyo`**, **`Panasonic`**, **`Sony`** e **`Mitsumi`**. L'uso della parola chiave **`noisp16`** disabilita l'utilizzo della scheda e dei lettori.

- Mitsumi

  `mcd=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`]`

- Optics Storage
  `optcd=`⟨*indirizzo-I/O*⟩
- Phillips CM206
  `cm206=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`]`
  Eventualmente si può utilizzare la sintassi seguente per attivare l'autorilevamento.
  `cm206=auto`
- Sanyo
  `sjcd=`⟨*indirizzo-I/O*⟩`[,`⟨*livello-IRQ*⟩`[,`⟨*canale-DMA*⟩`]]`
- SoundBlaster Pro
  `sbpcd=`⟨*indirizzo-I/O*⟩`,`⟨*tipo*⟩
  L'indirizzo di I/O è quello specifico dell'interfaccia del CD-ROM, e non quello della gestione dell'audio. Il tipo identifica il CD-ROM e può trattarsi di: **`SoundBlaster`**, **`LaserMate`**, **`SPEA`**.

Esempi

CD-ROM Aztech: indirizzo di I/O 220.

```
aztcd=0x220
```

CD-ROM Sony CDU-33A con la sua scheda originale: indirizzo di I/O 340, livello di IRQ non utilizzato.

```
cdu31a=0x340,0
```

CD-ROM Mitsumi: indirizzo di I/O 340, livello di IRQ 11.

```
mcd=0x340,11
```

CD-ROM Panasonic: livello di I/O 230.

```
sbpcd=0x230,SoundBlaster
```

16.8 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 / 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

Scheda NE2000: indirizzo I/O 300, livello di IRQ 11.

```
ether=11,0x300,eth0
```

Due schede NE2000: la prima configurata con indirizzo I/O 300 e livello di IRQ 11, la seconda con indirizzo di I/O 320 e livello di IRQ 10.

```
ether=11,0x300,eth0 ether=10,320,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=0,0,eth1
```

Scheda 3c503: indirizzo di I/O 300, livello di IRQ 9, ricetrasmettitore esterno.

```
ether=9,0x300,0,1,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.

```
ether=3,0,0,0,eth0
```

16.9 Controller dei floppy disk

`floppy=two_fdc`
Specifica la presenza di due *controller* per i dischetti. Con questa istruzione si ritiene implicitamente che il secondo *controller* utilizzi l'indirizzo di I/O 0x370.

`floppy=`⟨*indirizzo-I/O*⟩`,two_fdc`
Viene specificata la presenza di un secondo *controller* con un indirizzo di I/O iniziale particolare.

`floppy=thinkpad`
Questa istruzione serve a gestire l'unità a dischetti dei computer Thinkpad.

`floppy=L40SX`
Questa istruzione serve a gestire l'unità a dischetti dei computer portatili IBM L40SX.

16.10 Mouse particolari

`bmouse=`⟨*livello-IRQ*⟩
Permette di definire il livello di IRQ di una interfaccia Bus Mouse.

`msmouse=`⟨*livello-IRQ*⟩
Permette di definire il livello di IRQ di una interfaccia MS Bus Mouse.

16.11 Stampa

`lp=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩`[,`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩`]...`
Attraverso questa istruzione è possibile indicare esplicitamente la presenza di una o più porte parallele per la stampa. In questo modo si evita l'autorilevamento e si lascia la possibilità di utilizzare una porta aggiuntiva per scopi diversi dalla stampa. Il livello di IRQ può essere zero, in tal caso si intende specificare che quella porta non ne fa uso.

`lp=0`
Disabilita la gestione della stampante.

16.12 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*

Moduli

Quando si ha raggiunto una certa dimestichezza con la ricompilazione del *kernel*, si potrebbe considerare l'utilizzo dei moduli come una inutile complicazione.
Ci sono tuttavia situazioni in cui l'uso dei moduli è una necessità, prima tra tutte l'installazione di Linux attraverso le distribuzioni che fanno uso di moduli per ottenere un unico dischetto di avvio, oppure per ridurre 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 Linux.
Questo capitolo, pur trovandosi in questa parte iniziale di questo documento, non è rivolto ai principianti che potrebbero trovare alcuni punti particolarmente complessi (come il problema del *ramdisk* iniziale). Tuttavia, quando si installa Linux, potrebbe essere necessario conoscere l'uso dei parametri di alcuni moduli, il cui elenco si trova nella parte finale del capitolo.

17.1 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 di Linux sono quello che in altri sistemi viene definito *driver*.
Nella sezione 15.4 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 Linux organizzata con un *kernel* modulare, è consigliabile sfruttare quel *kernel* già predisposto, assieme ai suoi moduli.

17.1.1 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".

17.1.2 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 generalmente creato 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
```

17.1.3 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`.

17.1.4 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 informazioni.

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
```

17.1.5 Gestione automatica

Gli strumenti e la configurazione descritti nelle sezioni precedente, possono essere gestiti in modo automatico attraverso il *daemon*, **`kerneld`**. Questo viene generalmente avviato 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/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.

17.1.6 # 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 semplicemente specificando il nome del file del modulo senza l'estensione: in quest'ultimo caso, **`insmod`** cerca il file (con la sua naturale estensione) all'interno delle directory standard per i moduli.

Quando nel *kernel* è attivato il supporto del *kernel daemon* e il *daemon* **`kerneld`** è in funzione, 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.

17.1.7 # rmmod

`rmmod [-r]` ⟨*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 nella riga di comando, questi vengono scaricati nell'ordine in cui appaiono.

Se viene usata l'opzione **`-r`**, viene tentata una rimozione ricorsiva di tutti i moduli utilizzati da quelli indicati nella riga di comando.

Quando nel *kernel* è attivato il supporto del *kernel daemon* e il *daemon* **`kerneld`** è in funzione, non dovrebbe essere necessario l'utilizzo di **`rmmod`** per scaricare i moduli.

> **`rmmod`** è in realtà solo un *link* a **`insmod`** che quindi cambia il suo comportamento quando viene avviato utilizzando quel nome.

Esempi

`#` ***`rmmod plip`***

Scarica il modulo **`plip`**.

17.1.8 $ 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 4KB);
- **`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
```

17.1.9 # 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 generalmente utilizzato 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 *log* di 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 uno spezzone di uno degli *script* della procedura di inizializzazione del sistema, in cui si avvia la generazione del file delle dipendenze tra i moduli se il programma esiste.

17.1.10 # 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/`,...) 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/`.

17.1.11 /etc/conf.modules

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 alla fine, la barra inclinata rovescia (\), subito prima del codice *newline*.

Le righe di questo file vengono interpretate attraverso una *shell*, e questo permette di utilizzare le tecniche di sostituzione comunemente fornite 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 guida interna: **`depmod(1)`**.

In linea di massima, si possono accumulare più direttive dello stesso tipo.

17.1.11.1 alias

`alias` ⟨*alias*⟩ ⟨*modulo-reale*⟩

La direttiva **`alias`** permette di definire un nome alternativo a un nome di un modulo reale. Ciò può essere utile a vario titolo, e in ogni caso sono definiti 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
...
```

Si può osservare per esempio che si può fare riferimento al modulo **isofs** anche attraverso il nome **iso9960**. Tuttavia, gli alias non sono semplicemente di aiuto agli "smemorati", ma 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.

17.1.11.2 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 utilizza il canale di I/O all'indirizzo 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.

17.1.12 # kerneld

kerneld [debug] [keep] [delay=⟨*secondi*⟩] [type=⟨*numero-messaggio*⟩]

kerneld è il *daemon* che si occupa di gestire automaticamente i moduli del *kernel*, sempre che quest'ultimo sia stato compilato in modo da includere questa possibilità di gestione automatizzata. **kerneld** viene normalmente attivato attraverso la procedura di avvio 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 daemon per la gestione dei moduli"
    /sbin/kerneld
fi
```

Vedere eventualmente **kerneld(1)**.

17.2 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 eseguire il *mount* del *filesystem* principale.

Per fare un esempio concreto, basta pensare alla realizzazione di un *kernel* tuttofare per una distribuzione Linux. È impensabile che si realizzi un *kernel* in grado di montare un *filesystem root* 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 *ramdisk* iniziale, o *initrd*, e naturalmente, il *kernel* deve essere in grado di montare un tale *filesystem*.

- *Initial RAM disk (initrd) support* (15.5.5) **Y**

17.2.1 Ram disk iniziale

Il metodo per realizzare un *ramdisk* iniziale è descritto nella documentazione allegata ai sorgenti del *kernel*: `/usr/src/linux/Documentation/initrd.txt`.

In breve si tratta di predisporre un *ramdisk* con un *filesystem* Ext2 o Minix, contenente il minimo indispensabile a caricare i moduli necessari prima del *mount* del *filesystem root*. Serviranno quindi i moduli e il programma **insmod**. A parte questo, serve una *shell* minima e le eventuali librerie.

Il *ramdisk* realizzato per questo scopo deve contenere il programma (o lo *script*) `/linuxrc`, che verrà avviato appena montato il *ramdisk*. Quando `/linuxrc` termina la sua esecuzione il *ramdisk* viene generalmente smontato.

Di solito, il *ramdisk* 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**.

17.2.2 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 root*.

Il file del *ramdisk* viene collocato nella directory `/boot`. Si tratta di un *ramdisk* in formato Ext2 e compresso con **gzip**. Per analizzarne il contenuto si deve decomprimere e montare come disco 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 *ramdisk* 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
```

17.3 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.

17.3.1 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 automatico l'utilizzo dei due moduli nel momento in cui si utilizza il programma **`ifconfig`** per configurare le schede.

17.3.2 plip

Il modulo **`plip`** permette di gestire una o più porte parallele per una connessione *point to point* 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.

17.3.3 Adattatore SCSI

Convenzionalmente, si tende ad assegnare l'alias **`scsi_hostadapter`** al modulo necessario per pilotare la eventuale scheda SCSI presente nel proprio computer.

```
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, se contemporaneamente si vuole utilizzare per questo un modulo. Il problema è già stato affrontato nella discussione sul *ramdisk* iniziale.

17.4 Parametri dei moduli

I moduli, come porzioni del *kernel* avviate quando il sistema è già in funzione, hanno poche possibilità di individuare automaticamente i dispositivi per i quali sono predisposti. Ciò significa che spesso i moduli richiedono all'avvio l'indicazione di alcune informazioni, definite parametri.
Qui verranno mostrati solo i parametri di alcuni moduli, in particolare ciò che potrebbe essere necessario per condurre a termine un'installazione di Linux.
È importante tenere presente che gli indirizzi di I/O sono 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.

17.4.1 Adattatori di rete

Segue l'elenco di alcuni moduli per la gestione delle connessioni in rete e i parametri che con questi possono essere utilizzati. Questi moduli appartengono al tipo **`net`**.

- 3Com 3c501
  `3c501 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- 3Com 3c503 EtherLink II
  `3c503 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩ `xcvr=`{`0`|`1`}
  Ricetrasmettitore: 0=BNC, 1=AUI
- 3Com 3c505 EtherLink Plus
  `3c505 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- 3Com 3c507 EtherLink 16
  `3c507 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- 3Com 3c509 EtherLink III
  `3c509 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- 3Com 3c590/3c595 Vortex
  `3c59x io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- Ansel Communications 3200 EISA
  `ac3200 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩ `mem=`⟨*indirizzo-memoria*⟩
- Apricot 82596
  `apricot io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩

- Allied Telesis AT1700
  `at1700 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- DEC EtherWORKS DE100, DE101, DE200, DE201, DE202, DE210, DE422
  `depca io=`⟨*indirizzo-I/O*⟩
- DEC EtherWORKS 3: DE203, DE204, DE205
  `io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- de4x5 DEC EtherWORKS DE425, DE434, DE435, DE450, DE500
  `io=`⟨*indirizzo-I/O*⟩
- Cabletron E2100
  `e2100 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩ `mem=`⟨*indirizzo-mem*⟩ `xcvr=`{`0`|`1`}
  Ricetrasmettitore: 0=interno (BNC), 1=esterno (AUI).
- EtherExpress Pro/10
  `eepro io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- EtherExpress
  `eexpress io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- ICL EtherTeam 16i/32 EISA
  `eth16i io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- Fujitsu FMV-181/182/183/184
  `fmv18x io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- HP PCLAN/plus
  `hp-plus io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- HP LAN
  `hp io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- HP10/100VG ANY LAN: J2577, J2573, 27248B, J2577, J2573, J2585
  `hp100 hp100_port=`⟨*indirizzo-I/O*⟩
- NE1000, NE2000, e compatibili
  `ne io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- NI5210 Ethernet
  `ni52 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
  `memstart=`⟨*indirizzo-mem-iniziale*⟩ `memend=`⟨*indirizzo-mem-finale*⟩

- PLIP - Parallel Link Internet Protocal
  `plip io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- SMC Ultra, SMC EtherEZ ISA
  `smc-ultra io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- SMC 9000 series Ethernet
  `smc9194 io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
  `ifport=`{`0`|`1`|`2`}

  Interfaccia: 0=auto, 1=TP, 2=AUI/BNC.
- DEC 21040 e la maggior parte dei modelli 21*40 (Ethernet)
  `tulip io=`⟨*indirizzo-I/O*⟩
- AT&T GIS (nee NCR) WaveLAN Ethernet-like ricetrasmettitori in radiofrequenza
  `wavelan io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
- WD8003, WD8013 e compatibili
  `wd io=`⟨*indirizzo-I/O*⟩ `irq=`⟨*livello-IRQ*⟩
  `mem=`⟨*indirizzo-mem-iniziale*⟩ `mem_end=`⟨*indirizzo-mem-finale*⟩

  L'indirizzo finale viene indicato solo quando la dimensione della memoria non è standard.

Esempi

Scheda 3c503: indirizzo di I/O 300, livello di IRQ 9, ricetrasmettitore esterno (AUI).
Modulo **`3c503`**.

```
io=0x300 irq=9 xcvr=1
```

Scheda 3Com 3c509: indirizzo di I/O 210, IRQ 10.
Modulo **3c509**.

```
io=0x210 irq=10
```

Scheda compatibile NE2000: indirizzo di I/O 300, IRQ 11.
Modulo **ne**.

```
io=0x300 irq=11
```

Connessione PLIP su porta parallela: indirizzo di I/O 378, IRQ 7.
Modulo **plip**.

```
io=0x378 irq=7
```

17.4.2 Adattatori SCSI

Segue l'elenco di alcuni moduli per la gestione di schede SCSI e dei parametri che possono essere utilizzati. Questi moduli appartengono al tipo **scsi**.

- Adaptec AHA-152x
  `aha152x aha152x=`⟨*indirizzo-I/O*⟩`,`⟨*irq*⟩`,`⟨*scsi-id*⟩`,`⟨*reconnect*⟩`,`⟨*parità*⟩
- Adaptec AHA-1542
  `aha1542 bases=`⟨*indirizzo-I/O*⟩
- Future Domain TMC-8xx
  `controller_type=2 base_address=`⟨*indirizzo-memoria*⟩ `irq=`⟨*livello-IRQ*⟩
  Esempio. Future domain TMC-800: indirizzo di I/O ca000, livello di IRQ 10.

  ```
  controller_type=2 base_address=0xca000 irq=10
  ```

- Seagate ST-01/02
  `seagate controller_type=1 base_address=`⟨*indirizzo-memoria*⟩ `irq=`⟨*livello-IRQ*⟩
- Pro Audio Spectrum/Studio 16
  `pas16 pas16=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩
- IOMEGA PPA3/Parallel ZIP
  `ppa_base=`⟨*indirizzo-I/O*⟩
- Trantor T128/T128F/T228
  `t128 t128=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩
- NCR-5380
  `NCR5380 ncr5380=`⟨*indirizzo-I/O*⟩`,`⟨*livello-IRQ*⟩`,`⟨*canale-DMA*⟩
- Buslogic
  `BusLogic buslogic=`⟨*indirizzo-I/O*⟩

Esempi

Adaptec AHA-1522: indirizzo di I/O 330, IRQ 11, identificatore SCSI 7.
Modulo **aha152x**.

```
aha152x=0x330,11,7
```

Adaptec AHA-1542: indirizzo di I/O 330.
Modulo **aha1542**.

```
bases=0x330
```

17.4.3 CD-ROM

Segue l'elenco di alcuni moduli per la gestione di CD-ROM connessi a schede proprietarie (non si tratta dei normali CD-ROM IDE/ATAPI) e i parametri che con questi possono essere utilizzati. Questi moduli appartengono al tipo **`cdrom`**.

- Aztech CD268
  `aztcd aztcd=`⟨*indirizzo-I/O*⟩
- Sony CDU-31A / CDU-33A
  `cdu31a cdu31a_port=`⟨*indirizzo-I/O*⟩ `cdu31a_irq=`⟨*livello-IRQ*⟩
- Philips/LMS cm20
  `cm206 cm206=`⟨*indirizzo-I/O*⟩
- GoldStar R420
  `gscd gscd=`⟨*indirizzo-I/O*⟩
- Mitsumi
  `mcd mcd=`⟨*indirizzo-io*⟩`,irq=`⟨*livello-IRQ*⟩
- Mitsumi XA/Multisession
  `mcdx mcdx=`⟨*indirizzo-io*⟩`,irq=`⟨*livello-IRQ*⟩
- SoundBlaster Pro/Matsushita/Panasonic/Longshine/CreativeLabs/TEAC/ECS-AT
  `sbpcd sbpcd=`⟨*indirizzo-I/O*⟩`,type=`{0|1|2|3}
  tipo: 0=LaserMate, 1=SoundBlaster, 2=SoundScape, 3=Teac16bit
- Sanyo CDR-H94A
  `sjcd sjcd_base=`⟨*indirizzo-I/O*⟩
- Sony CDU-535
  `sonycd535 SONYCD535=`⟨*indirizzo-I/O*⟩

Esempi

CD-ROM Aztech: indirizzo di I/O 220.
Modulo **`aztcd`**.

```
aztcd=0x220
```

CD-ROM Sony CDU 33: indirizzo di I/O 340, livello di IRQ non utilizzato.
Modulo **`cdu31a`**.

```
cdu31a_port=0x340 cdu31a_irq=0
```

CD-ROM Mitsumi: indirizzo di I/O 340, livello di IRQ 11.
Modulo **`mcd`**.

```
mcd=0x340,11
```

CD-ROM Panasonic su scheda SoundBlaster: indirizzo di I/O 230.
Modulo **`sbpcd`**.

```
sbpcd=0x230,1
```

17.5 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*

Parte V

Processi di elaborazione

Appunti Linux

Via Turati, 15 - I-31100 Treviso - daniele@calion.com

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.

<*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/*>

La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.

<*http://www.pluto.linux.it/ildp/AppuntiLinux/*>

<*http://ildp.psy.unipd.it/AppuntiLinux/*>

<*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

<*http://linux.calion.com/pluto/ildp/AppuntiLinux/*>

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 *root*, cioè quello che genera tutti gli altri, è **`init`** che a sua volta è attivato direttamente dal *kernel*.

18.1 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;

18.1.1 /proc/

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 esecuzione.
Il listato seguente mostra il contenuto del file `/proc/1/status`.

```
Name:   init
State:  S (sleeping)
Pid:    1
PPid:   0
Uid:    0       0       0       0
Gid:    0       0       0       0
VmSize:      880 kB
VmLck:         0 kB
VmRSS:       280 kB
VmData:      208 kB
VmStk:        12 kB
VmExe:        20 kB
VmLib:       588 kB
SigPnd: 00000000
SigBlk: 00000000
SigIgn: d7f0dafc
SigCgt: 280b2403
```

18.2 Nascita e morte di un processo

Come si era detto, 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`**.

18.2.1 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, denominato `core` e contenente l'immagine del processo interrotto. Questi file servono quindi solo a documentare un incidente di funzionamento e 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
```

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.

18.3 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 normalmente definito IPC, o *Inter Process Communication*, ma questa definizione viene normalmente confusa con un tipo particolare di comunicazione definito IPC di System V.
I metodi normalmente utilizzati sono di tre tipi: invio di segnali, *pipe* e System V IPC.

18.3.1 Segnali

I segnali sono dei messaggi elementari che possono essere inviati a un processo, permettendo a questo di essere informato di una particolare condizione manifestatasi 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 normalmente prevedibili per un determinato tipo di segnale.
Segue un breve elenco dei segnali più importanti.

- **`SIGINT`**

  È un segnale di interruzione intercettabile, normalmente inviato attraverso la tastiera del terminale, con la combinazione [*Ctrl+c*], al processo in esecuzione in primo piano (*foreground*). Di solito, il processo che riceve questo segnale viene interrotto.

- **`SIGTERM`**

  È un segnale di conclusione intercettabile, normalmente inviato 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 eventualmente avviati in *background* ancora attivi ricevono questo segnale. Può essere generato anche a causa della "morte" del processo controllante.

La tabella 18.1 elenca i segnali descritti dallo standard POSIX.1, mentre l'elenco completo può essere ottenuto consultando **`signal(7)`**.
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.

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 in *background* che necessita dell'input.
SIGTTOU	D	Processo in *background* che necessita dell'output.

Tabella 18.1: Segnali gestiti da Linux.

L'utente ha a disposizione in particolare due mezzi per inviare segnali ai programmi:

- la combinazione di tasti [*Ctrl+c*] che di solito viene interpretata dalla *shell* che a sua volta, di conseguenza, invia un segnale **`SIGINT`** al processo in esecuzione in primo piano (*foreground*);
- l'uso di **`kill`** (programma o comando interno di *shell*) per inviare un segnale particolare a un processo particolare.

18.3.2 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 miofile | 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 far ricorso alle funzionalità della *shell*. Questi, sono dei file speciali definiti proprio **`FIFO`** e vengono creati attraverso il programma **`mkfifo`**. Nell'esempio seguente vengono mostrati 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 miofile >> fifo1 &
```

Invia `miofile` 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 *named pipe*, o *pipe* con nome, e si contrappongono a quelle normali che a volte vengono dette *pipe* anonime.

18.3.2.1 Broken pipe

Quando all'interno di una *pipeline* di qualunque tipo un processo viene interrotto, 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.

18.3.3 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.

18.4 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.

---

[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.

Procedura di inizializzazione del sistema (System V)

Quando 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* normalmente contenuti 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 solitamente le azioni necessarie per fermare l'attività del sistema, in modo da permetterne lo spegnimento; il livello **`6`** normalmente riavvia il sistema; il livello **`1`** mette il sistema in condizione di funzionare in modalità monoutente.

> Le distribuzioni 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 e dei livelli di esecuzione costituisce il punto su cui si distinguono maggiormente le distribuzioni di 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 Linux) mostra una interpretazione della soluzione del problema, non il risultato generalmente accettato.

19.1 init

`init` è il processo principale che genera tutti gli altri. All'avviamento del sistema legge il file `/etc/inittab` il quale contiene le informazioni per attivare gli altri processi necessari per il sistema compresa la gestione dei terminali. Per prima cosa viene determinato il livello iniziale di esecuzione (*run level*), ottenendo l'informazione dalla riga **`initdefault`** di `/etc/inittab`. Quindi vengono attivati i processi di attivazione del sistema e infine i processi che combaciano con il livello di esecuzione attivato.

19.1.1 # 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`**.

19.1.2 /etc/inittab

Il file `inittab` descrive quali processi vengono avviati al momento del *boot* del sistema e durante il funzionamento normale del sistema stesso. **`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 del *record* 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 **`utmp`** e **`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 identificata da 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 del *boot* del sistema. Il campo del livello di esecuzione viene ignorato.
- **`bootwait`**
  Il processo viene eseguito al momento del *boot* 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* cosiddetti "a richiesta" che vengono presi in considerazione quando viene eseguito **`init`** seguito da una lettera **`a`**, **`b`** o **`c`** che rappresentano appunto tre livelli possibili 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 automaticamente definito all'avviamento del sistema. Se non viene specificato, **`init`** richiede l'inserimento di questo valore attraverso la console.
- **`sysinit`**
  Il processo viene eseguito al momento del *boot* 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, e, contemporaneamente, è 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 ad 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 iniziale di esecuzione: **`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/rc1` 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/rc0` 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/rc6` 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 ad **1**. Responsabile dell'attivazione delle console, come della connessione di tutti i terminali, è, in questo caso, **mingetty** (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 Window System (RedHat).

19.1.3 /etc/initscript

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.

19.2 /etc/rc.d/rc*

Si tratta di una serie di *script* di *shell* eseguiti direttamente o indirettamente attraverso le indicazioni contenute nel file `/etc/inittab`. Questo gruppo di *script* può essere diverso da un sistema all'altro a seconda della distribuzione di Linux o a seconda della versione.

19.2.1 /etc/rc.d/rc.local

Una convenzione diffusa fa sì che questo *script* faccia parte della procedura di avvio della maggior parte delle distribuzioni Linux (anche se non è detto che debba trovarsi necessariamente nella directory `/etc/rc.d/`). Questo *script* viene eseguito alla fine della procedura stessa ed è per questo disponibile per le variazioni locali da apportare al sistema. Volendo fare un'associazione con il sistema Dos, si potrebbe paragonare questo *script* al file `AUTOEXEC.BAT`.

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 da Linux è quello del *filesystem* virtuale montato nella directory `/proc/`.

20.1 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) 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*.

```
  4:39pm  up  1:50,  6 users,  load average: 0.40, 0.95, 0.61
61 processes: 60 sleeping, 1 running, 0 zombie, 0 stopped
CPU states: 27.6% user,  3.2% system,  5.2% nice, 69.6% idle
Mem:   22724K av,  21612K used,   1112K free,  14500K shrd,   1584K buff
Swap:  16628K av,   6168K used,  10460K free                   6260K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
  688 daniele    0   0  1880 1836   888 S       0 22.3  8.0   3:32 xv
  417 root       1   0  3104 2624  1296 S       0  5.0 11.5   4:56 X
  447 daniele    0   0  5176 4408  1228 S       0  2.6 19.3   7:20 wine
  421 daniele    0   0   648  604   420 S       0  0.2  2.6   0:07 fvwm95-2
  533 daniele   12   0   492  468   352 R       0  0.2  2.0   7:54 top
  428 daniele    0   0   616  604   452 S       0  0.1  2.6   0:03 FvwmTaskBar
    1 root       0   0   188  136   120 S       0  0.0  0.5   0:01 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
```

Figura 20.1: Il programma **top**.

20.1.1 Intestazioni

I programmi che visualizzano la situazione dei processi, utilizzano spesso delle sigle per identificare alcune caratteristiche. La tabella 20.1 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 effettivamente utilizzata.
SWAP	La dimensione della memoria di *swap* 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 20.1: 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 20.2.

20.1.2 $ 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 eventualmente iniziare 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.

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 completamente spostato nella memoria virtuale).
N	Ha un valore *nice* positivo (rallentato).

Tabella 20.2: Lo stato del processo espresso attraverso una o più lettere alfabetiche.

`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 effettivamente in esecuzione, 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`** o **`cu`**.

`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 20.3.

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 20.3: 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 da 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.

20.1.3 $ 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.

20.1.4 $ top

`top` [⟨*opzioni*⟩]

Visualizza la situazione sull'utilizzo delle risorse di sistema attraverso una tabella dell'attività principale della CPU, cioè dei processi che la 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.

20.2 Accesso ai file

A volte è importante conoscere se un file sia 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.

20.2.1 # 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 normali per buona parte delle sue funzionalità.

Quando si utilizza **fuser** per ottenere l'elenco dei processi che accedono a determinati file, 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 *root*;
- **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 una unità di memorizzazione montata nel *filesystem*, o una directory che costituisce il *mount point* 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

MIOFILE=./miofile

if fuser -s $MIOFILE
then
    echo "Il file $MIOFILE è in uso";
else
    # esegue qualche azione sullo stesso
    ...
fi
```

20.3 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 in particolare, 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
```

20.3.1 $ 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.

20.3.2 $ 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 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.

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 immediatamente sottomessi agli ordini.
La tabella 18.1 elenca i segnali descritti dallo standard POSIX.1, mentre l'elenco completo può essere ottenuto consultando **`signal(7)`**.

21.1 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 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`**).

21.2 Segnali attraverso la tastiera

Alcuni segnali possono essere inviati al programma con il quale si interagisce attraverso alcune 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 dal tipo di configurazione del terminale che si utilizza effettivamente. Questa può essere controllata o modificata attraverso il programma **`stty`** (25.2.1). Come si può vedere dall'esempio seguente, alcune combinazioni di tasti (rappresentate nella forma *^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
```

21.3 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`**,

[*Ctrl+z*]

sospende il processo in primo piano (*foreground*) attraverso un segnale **`SIGTSTP`**,

`fg`

fa riprendere il processo sospeso attraverso un segnale **`SIGCONT`**.

21.4 Kill

Il modo normale per inviare un segnale a un processo è l'uso di **`kill`** che, 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'utilizzo 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*.

21.4.1 $ 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.

21.4.2 $ 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.

21.4.3 # 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 20.2.1.

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.

22.1 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 a **`ksh`** e **`csh`**, 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 **`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.

22.1.1 Foreground e background

L'attività di un *job* può avvenire in *foreground* (in primo piano) o in *background* (in sottofondo). 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 in *background*.
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* in *background*, bisogna almeno provvedere a ridirigere l'input e l'output.

22.1.2 Avvio di un job in background

Un programma viene avviato esplicitamente come *job* in *background* quando alla fine della riga di comando viene aggiunto il simbolo **`&`**. Per esempio:

`#` ***`make zImage > ~/make.msg &`***

avvia in *background* 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* in *background*, la *shell* restituisce una riga contenente il numero del *job* e il numero del primo processo generato da questo *job* (PID). Per esempio:

```
[1] 173
```

rappresenta il *job* numero 1 che inizia con il processo 173.
Se viene avviato un *job* in *background* e questo 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* in *background* 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
```

22.1.3 Sospensione di un job in foreground

Se è stato avviato un *job* in *foreground* 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 in *background*. Quando un *job* viene sospeso, la *shell* genera una riga come nell'esempio seguente:

```
[1]+ Stopped pippo
```

dove il *job* **`pippo`** è stato sospeso.

22.1.4 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 *foreground*, di conseguenza, quello che si ottiene è solo l'elenco dei *job* in *background*.

Alcune opzioni

`-l`

Permette di conoscere anche il numero PID del processo leader di ogni *job*.

`-p`

Emette solo i numeri PID del processo leader di ogni *job*.

Esempi

`$ jobs`

Si ottiene l'elenco normale dei *job* in *background*. 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 -l`

Si ottiene l'elenco dei *job* in *background* con in più l'indicazione del PID del processo leader di ogni *job*.

```
[1]    232 Running                  yes >/dev/null &
[2]-   233 Stopped (tty output)     mc
[3]+   235 Stopped (tty input)      unix2dos
```

22.1.5 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 22.1.

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 22.1: Elenco dei parametri utilizzabili come riferimento ai *job* di *shell*.

22.1.6 fg

`fg [⟨job⟩]`

Il comando **`fg`** porta in *foreground* un *job* che prima era in *background*. Se non viene specificato il *job* su cui agire, si intende quello attuale.

22.1.7 bg

`bg [⟨job⟩]`

Il comando **`bg`** permette di fare riprendere (in *background*) 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 in *background* 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.

22.1.8 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 omonimo, permette di inviare un segnale al processo leader 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`**.

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.

22.2 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 un semplice 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 una 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
```

22.2.1 trap

`trap [-l]` `[`⟨*comando*⟩`]` `[`⟨*segnale*⟩`]`

Il comando espresso come argomento di **`trap`**, viene eseguito quando la *shell* riceve il/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`**.

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 il *daemon cron*, mentre la priorità può essere modificata attraverso il valore *nice*.
La tabella 23.1 elenca i programmi e i file a cui si accenna in questo capitolo.

Nome	Descrizione
cron	*Daemon* per l'esecuzione dei comandi pianificati.
crontab	Modifica la tabella della pianificazione dei comandi.
/etc/crontab	Tabella di pianificazione di sistema.
/var/spool/cron/	Directory contenente le tabelle *crontab* 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 23.1: Riepilogo dei programmi e dei file per la gestione dello *scheduling*.

23.1 Cron

Nel bel mezzo della notte, mentre si sta lavorando isolati da qualunque rete, potrebbe capitare di notare una intensa attività del disco fisso senza apparente giustificazione. Di solito si tratta di *cron*.
Cron è il *daemon* che si occupa di eseguire dei comandi in momenti determinati in base a quanto stabilito all'interno dei 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 dover necessariamente sapere 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 *daemon cron* dell'avvenuto cambiamento nella situazione dei piani di esecuzione.[1]
L'output dei comandi che *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 *mail* all'utente cui appartiene il *crontab*.
Il *daemon cron* viene di norma avviato durante la procedura di inizializzazione del sistema. Di questo *daemon* ne esistono almeno due tipi diversi per Linux: Vixie Cron e Dillon's Cron, dai nomi dei loro autori. Nelle sezioni seguenti si fa riferimento in particolare a quello di Paul Vixie.
Vedere **`cron(1)`**, **`crontab(1)`** o **`crontab(8)`**, e **`crontab(5)`**.

23.1.1 # cron (Vixie)

`cron`

`cron` è un *daemon* funzionante in *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 nella interpretazione dei file *crontab*, le altre notizie sul suo utilizzo sono riportate in occasione della presentazione di quei file.

23.1.2 $ 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 *crontab* di un altro utente. Di solito, **`crontab`** viene utilizzato con l'opzione **`-e`** per modificare o creare il file *crontab*.

[1] Indipendentemente dal fatto che il *daemon cron* necessiti o meno di essere avvisato.

I file *crontab* vengono poi utilizzati dal *daemon cron* che si occupa di eseguire i comandi lì indicati.

Sintassi

`crontab [-u` ⟨*utente*⟩`]` ⟨*file*⟩
Sostituisce il *crontab* con il contenuto del file indicato come argomento.

`crontab -l [`⟨*utente*⟩`]`
Visualizza il *crontab* dell'utente.

`crontab -e [`⟨*utente*⟩`]`
Crea o modifica il *crontab* dell'utente.

`crontab -r [`⟨*utente*⟩`]`
Cancella il *crontab* dell'utente.

Utilizzo

`$` ***`crontab -e`***
Inizia la modifica del file *crontab* dell'utente.

`$` ***`crontab -l`***
Visualizza il contenuto del *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 *crontab* dell'utente.

23.1.3 /var/spool/cron/⟨*utente*⟩

Si tratta dei file *crontab* degli utenti normali, creati normalmente attraverso il programma **`crontab`**. Ogni utente ha il proprio, e i comandi contenuti al suo interno vengono eseguiti come se fossero stati avviati dall'utente proprietario del file *crontab*.
Le righe vuote sono ignorate e così anche quelle il cui primo carattere diverso da *blank* (spazi e/o caratteri di tabulazione) è il simbolo **#** (commenti).
Un *record* significativo può essere un assegnamento di una variabile di ambiente o un comando *cron*.

23.1.3.1 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 *daemon cron* utilizza una serie di variabili di ambiente per determinare il proprio comportamento. Alcune di queste ricevono un valore predefinito da *cron* stesso e tutte, tranne **`LOGNAME`**, possono essere modificate attraverso un assegnamento all'interno del file *crontab*.

- **`SHELL`**
  Il valore iniziale è `/bin/sh` stabilendo così che i comandi *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 *home* dell'utente.
- **`MAILTO`**
  Non viene preassegnata da *cron* e se risulta definita, ma non vuota, viene utilizzata per determinare il destinatario dei messaggi di posta generati da *cron* stesso. 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 è lo stesso utente a cui appartiene il file *crontab*.

---

[2] La possibilità di inserire degli assegnamenti di variabili di ambiente all'interno di un file *crontab* è una particolarità del *cron* di Paul Vixie.
[3] Normalmente, `/bin/sh` è un *link* alla *shell* predefinita, ovvero a **`bash`**, che se avviata così, si comporta in modo compatibile con **`sh`**.

23.1.3.2 Comandi cron

Un tipico file *crontab* può contenere solo comandi *cron*. Il formato di questo può essere brevemente riassunto 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 possibile valore 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 6:10' 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 6:10' 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 6:10' 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 6:10' dei primi cinque giorni di gennaio, febbraio, marzo, maggio, giugno e luglio.

- Passo

  Invece di indicare momenti precisi, è possibile indicare una ripetizione o un passo. Questo può essere rappresentato con una barra inclinata 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 6:10', le 6:20', le 6:30', le 6:40', le 6:50' e le 6:00'. 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 di un carattere *newline* 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 caratteri *newline*.

23.1.3.3 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 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/daily.job >> $HOME/tmp/out 2>&1

# Esegue alle 14:15’ del primo giorno di ogni mese.
# L’output viene inviato tramite posta all’utente ‘‘tizio’’.
15 14 1 * *     $HOME/bin/monthly

# Esegue alle 22 di ogni giorno lavorativo (da lunedì al venerdì).
# In particolare viene inviato un messaggio di posta 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 4:05’ di ogni domenica.
5 4 * * 0       echo "Buona domenica"
```

23.1.4 /etc/crontab

Il file `/etc/crontab` ha un formato leggermente diverso da quello dei file *crontab* normali. In pratica, dopo l’indicazione del quinto campo 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?
```

23.2 At

L’esecuzione di una 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 da *cron*, per questo scopo gli si affianca **atrun**, eseguito periodicamente da *cron*.

atrun può essere immaginato come una sorta di *daemon* con lo scopo di eseguire dei *job* accodati su diversi tipi di code da **at** o **batch**. In realtà si tratta di un programma che viene messo in funzione periodicamente (normalmente una volta al minuto) da *cron*. Per verificarlo basta analizzare il contenuto di `/etc/crontab` o del *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, a *job* accodati da **batch**.

La differenza tra questi due sta nel fatto che il primo accoda *job* da eseguire in un determinato momento, 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, non vengono posti limiti all'utilizzo di questi programmi.

23.2.1 Ambiente

Nel momento in cui si decide di fare eseguire un comando in un momento successivo a quello attuale, si pone 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 dello UID, della directory corrente, della maschera *umask* (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.

23.2.2 Restituzione dell'output

Quando si pianifica l'esecuzione di un comando in un momento successivo, si pone il problema di stabilire dove debba essere diretto il suo output. Sarebbe buona norma stabilirlo 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 (più precisamente attraverso `/bin/mail`).

23.2.3 # 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 eseguito periodicamente da *cron*.

23.2.4 $ 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 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 ragionevolmente utilizzato 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 ulteriori indicazioni: **today** (oggi); **tomorrow** (domani).

⟨*mmggaa*⟩ | ⟨*mm*⟩`/`⟨*gg*⟩`/`⟨*aa*⟩ | ⟨*gg*⟩`.`⟨*mm*⟩`.`⟨*aa*⟩

Una data può essere espressa con un gruppo di sei cifre, eventualmente separate 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, o 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`

Invia un messaggio, attraverso il sistema di *mail*, 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* indicato. 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 13:30.

`$` ***`at 15:00 tomorrow`***

Eseguirà domani alle 15:00 i comandi che verranno indicati di seguito (si conclude con una combinazione [*Ctrl+d*]).

`$` ***`at 10:00 10.11.99`***

Il 10 novembre 1999 alle 10:00 eseguirà i comandi specificati.

23.2.5 $ batch

`batch [`⟨*opzioni*⟩`] [`⟨*orario*⟩`]`

`batch` è normalmente un *link* al programma **`at`** (23.2.4). 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 una attività ridotta.

23.2.6 $ atq

`atq [`⟨*opzioni*⟩`]`

`atq` è normalmente un *link* al programma **`at`**. Quando **`at`** viene avviato usando il nome **`atq`**, emette l'elenco dei *job* in coda. Se viene specificata una coda particolare, attraverso l'opzione **`-q`**, si limita ad analizzare quella particolare coda.

23.2.7 $ atrm

`atrm` ⟨*job*⟩`...`

`atrm` è normalmente un *link* al programma **`at`**. Quando **`at`** viene avviato usando il nome **`atrm`**, elimina dalla coda i *job* specificati nella riga di comando.

23.2.8 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 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 una 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=ISO-8859-1; 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 *umask*.

```
umask 2
```

Quindi seguono una serie di assegnamenti di variabili esportate che riproducono l'ambiente del momento in cui il *job* è stato sottomesso.

```
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 sottometteva il *job*, era quella corrente. Se l'operazione fallisce, viene interrotto lo *script*.

```
cd /home/daniele || {
         echo 'Execution directory inaccessible' >&2
         exit 1
}
```

23.3 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 parla 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. Pretendere di portare agli stessi livelli di priorità altri programmi potrebbe significare bloccare il sistema. 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 in sottofondo (*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.

23.3.1 $ 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.

23.3.2 $ 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).

23.3.3 $ 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 in *background*.
In base a questo principio, cioè quello per cui si usa **`nohup`** per avviare un programma in *background* in modo che continui a funzionare anche quando l'utente si scollega, la priorità di esecuzione viene modificata, aumentando il valore *nice* di 5.
Il comando indicato come argomento non viene messo automaticamente in *background*, 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, `~/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 in *background* 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 (**`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`** avviata poco prima.

Si conclude l'attività della *shell* provocando un segnale di interruzione 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 in *background* 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 Window System e si chiude una finestra di terminale, un eventuale programma in *background* viene sicuramente eliminato, a meno di usare **`nohup`**.

Orologio di sistema

L'orologio del sistema non serve solo a fornire l'indicazione della data e dell'ora corrente. Da esso dipende anche il buon funzionamento di **`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.

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 da 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 un *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*)
  L'orario 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 all'orario universale.
- DST (*Daylight Savings Time*)
  Non rappresenta un orario preciso, ma uno spostamento all'indietro di un'ora. In pratica, si abbina questa sigla a quella della zona oraria, a indicare che l'ora solare corrispondente si ottiene sommando un'ora.
- CET (*Center Europe Time*)
  L'orario dell'europa centrale.

24.2 Distinzione tra hardware e software

Si distingue tra due orologi indipendenti: quello contenuto dell'hardware del computer 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 il computer è spento. In pratica, all'avvio del sistema, attraverso la procedura di inizializzazione, viene allineato l'orario del *kernel* con quello hardware.

24.2.1 Orologio hardware

L'orologio hardware è quello che appartiene alla parte fisica del computer, e normalmente è incorporato nella scheda madre. È alimentato attraverso una piccola batteria, in modo da poter funzionare anche quando il computer è spento.

Chi utilizza l'architettura PC tende a chiamarlo "orologio del BIOS" o "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, normalmente residente in 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 l'orario universale (UTC). Se non ci sono problemi di conflitti con altri sistemi operativi differenti da Linux, è importante che si utilizzi come riferimento l'orario 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 sull'orario 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 dell'orario 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 l'orario 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 è in funzione si provocano degli squilibri, sia nel sistema di pianificazione dei processi (**`cron`**) che in alte situazioni (anche X Window System può risentirne). Il minimo che può capitare è di osservare una intensa attività del sistema dovuta all'avvio di processi da parte di **`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 all'ora universale, è necessaria una opportuna conversione che avviene per mezzo di quanto indicato nel file `/etc/localtime`, che generalmente è un *link* simbolico al file giusto, contenuto nella directory `/usr/share/zoneinfo/` (queste collocazioni sono definite in base alla gerarchia standard di 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 normale 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 e/o l'ora attuale. La data permette all'utente *root* di modificare la data e l'ora del sistema.

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 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.

Direttiva	Descrizione
%%	Rappresenta un singolo simbolo di percentuale.
%n	Rappresenta un codice di *newline*.
%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**.

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 (e/o 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.

24.4.2 # clock

`clock [-u] [-r | -w | -s | -a]`

`clock` permette di accedere all'orologio hardware del computer.

Alcune opzioni

`-u`

Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita all'ora 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 sull'ora universale.

clock -w -u

Aggiorna l'orologio hardware, con riferimento all'orario 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 all'ora 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.

Alcune opzioni

`-u | --utc`

Stabilisce che l'informazione data-orario contenuta nell'orologio hardware deve essere (oppure è) riferita all'ora 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`**.

`--adjust`

Aggiusta la data dell'orologio hardware in funzione del contenuto del file `/etc/adjtime`.

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, in cui 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.

Parte VI

Terminali a caratteri

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.

<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>

<ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/>

<ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/>

La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.

<http://www.pluto.linux.it/ildp/AppuntiLinux/>

<http://ildp.psy.unipd.it/AppuntiLinux/>

<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>

<http://linux.calion.com/pluto/ildp/AppuntiLinux/>

Gestione del terminale a caratteri

Il terminale, in qualunque forma esso sia (console, terminale remoto, applicazione a finestra all'interno di X Window System) 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 ed è quella che precede immediatamente il *login*, cioè il sistema di riconoscimento dell'utente che si accinge ad utilizzare il sistema.

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/keytables/	Directory dei file di mappa delle varie nazionalità.
loadkeys	Modifica la mappa della tastiera.
dumpkeys	Emette la mappa della tastiera in funzione.
stty	Definisce le caratteristiche della connessione del terminale.
/etc/termcap	Database obsoleto delle caratteristiche dei terminali.
/usr/share/terminfo/?/*	Database delle caratteristiche dei terminali.
$TERM	Variabile che definisce il tipo di terminale in uso.
clear	Ripulisce lo schermo.
reset	Reinizializza l'impostazione del terminale.
getty	Attiva la gestione della console o del terminale.
agetty	Programma alternativo a **`getty`**.
/etc/issue	Messaggio introduttivo prima del *login*.

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

Quando si pensa alla gestione della tastiera, occorre ricordare che ciò che conta è fare in modo che funzioni correttamente la tastiera connessa al proprio computer. Questo è importante perché, quando si utilizza una connessione remota, per esempio attraverso **`telnet`**, la tastiera che si adopera dipende, per la sua configurazione, dal computer a cui è fisicamente connessa, e non dal computer *host* 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 un collegamento remoto viene dato un comando per modificare la modalità di funzionamento o la mappa della tastiera, l'effetto si risentirà sulla console del computer che riceve il comando e non nel terminale remoto.

Queste considerazioni permettono anche di comprendere che il collegamento remoto è indipendente da qualunque configurazione che riguardi la tastiera di un certo computer *host*, perciò, una configurazione errata che renda inutilizzabile una console, può essere corretta attraverso una connessione remota.

25.1.2 Console e Terminali virtuali

Linux, come altri sistemi Unix, consente di gestire diversi terminali virtuali sull'unica console effettivamente esistente. Attraverso alcune combinazioni di tasti si riesce a passare da un terminale virtuale all'altro. 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 corrisponde, quasi sempre, al terminale 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 tutti i terminali virtuali, oppure solo su quello attivo.

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 il terminale virtuale attivo, e quando viene utilizzato a distanza, attraverso **`telnet`** o un altro programma simile, ha effetto sul terminale virtuale attivo nel computer *host* 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 di X Window System.

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 sotto X Window System. Questa modalità non può essere utilizzata per i terminali virtuali normali.

`-k`

L'opzione **`-k`** attiva la modalità *keycode*, o MEDIUMRAW. Questa modalità non può essere utilizzata per i terminali virtuali normali.

`-a`

L'opzione **`-a`** attiva la modalità ASCII o XLATE. Questa modalità è quella normale quando si utilizzano i terminali 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 computer 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, il computer 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 un collegamento remoto attraverso **`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 di un terminale virtuale, quello attivo, o meglio, quello 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 Emitter Diod*), 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 rapporto 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 **`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 del primo terminale 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 i tasti premuti alle applicazioni.
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 normalmente contenuti nella directory `/usr/lib/kbd/keytables/`.
Il sistema della mappa della tastiera che si descrive qui e nelle sezioni seguenti, riguarda solo i terminali virtuali di Linux e non l'impostazione fatta da X Window System 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 sotto X Window System.

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 potrebbe essere desiderabile una sua modifica.

All'interno del file sono ammessi i commenti, prefissati con un punto esclamativo (**`!`**) oppure con il simbolo **`#`**.

Le righe che definiscono qualcosa possono essere continuate con una barra inclinata rovescia (\) che precede il codice *newline*.

Possono essere usate diverse definizioni, secondo le sintassi seguenti.

`charset "iso-8859-`*x*`"`

In questo caso si definisce il set 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 della 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 questo termine, modificatori, 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	Shift	1
Alt destro	AltGr	2
Control	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 indica 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 loro combinazioni, che si intendono utilizzare. Questo viene definito attraverso una 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 seguente, 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 una 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 della eventuale combinazione con i modificatori previsti con l'istruzione **`keymaps`**.

Quando si vuole indicare una 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 una istruzione **`keycode`** normale per le situazioni normali (tasto premuto da solo, con lo *shift* e 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 scoprire alcune cose interessanti. In particolare, si scopre l'utilizzo 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 il primo terminale virtuale, e questo viene chiaramente definito 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 il tredicesimo terminale 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 tutti i terminali 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 all'interno della directory `/usr/lib/kbd/keytables/`.
È importante considerare che la modifica interviene su tutti i terminali virtuali, e se si tenta qualcosa del genere attraverso una connessione remota si interviene sul computer con il quale si è connessi, e non con 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* 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/keytables/it.map***

Carica la mappa contenuta nel file `/usr/lib/kbd/keytables/it.map`.

$ ***loadkeys it.map***

Esattamente come nell'esempio precedente.

$ ***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* l'attuale mappa della tastiera.
Vedere **dumpkeys(1)**.

25.2 Configurazione del terminale

Le caratteristiche dei terminali possono essere molto diverse e questo è il principale problema che si pone di fronte alla ricerca verso una standardizzazione nel comportamento dei programmi per i sistemi Unix. Il primo programma di configurazione da prendere in considerazione è **stty**, dal quale dipende la definizione di alcune funzionalità fondamentali, quali la combinazione [*Ctrl+c*] come segnale di interruzione. La fase successiva è la definizione delle particolarità dei terminali, attraverso una sorta di *database*, in modo da permettere ai programmi di potervisi adattare.

25.2.1 $ 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 utilizzare questo programma è quello di conoscere le combinazioni di tasti che si possono utilizzare per generare dei segnali particolari. Utilizzando **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*].
Vedere **info stty** oppure **stty(1)**.

25.2.2 Termcap e terminfo

Il primo tipo di terminale, la telescrivente, non poneva particolari problemi 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 dal codice *newline*.

Quando il terminale attuale viene ancora usato 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 una sorta di *database* contenente tutte le informazioni necessarie.

Di questo *database* ne esistono due tipi: *termcap* e *terminfo*. Il primo è il più antico e ormai obsoleto ma viene mantenuto per assicurare la compatibilità con i programmi più vecchi.

Il *database 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 *database 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 questo *database* dovrebbe essere la directory `/usr/share/terminfo/`, ma se la propria distribuzione Linux non è organizzata perfettamente, potrebbe trovarsi in `/usr/lib/terminfo/`. Se ci si trova in difficoltà potrebbe essere conveniente la creazione di un *link* 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 *link* 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 *database terminfo*, ma questo può essere alterato utilizzando la variabile di ambiente **`TERMINFO`**, per indicare una directory differente. Volendo è possibile personalizzare il *database* creando una struttura analoga a partire da `~/.terminfo/`, cioè dalla directory `.terminfo/` nella propria directory *home*.

25.2.3 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 non risponde nel modo consueto, lo schermo viene ridisegnato solo parzialmente,...

25.2.4 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 all'interno del *database* di terminali. Per esempio, il fatto che su una tastiera sia presente il tasto [*Canc*], non vuol necessariamente dire che poi questo darà i risultati che ci si aspetta: la sua pressione potrebbe non avere alcun effetto, oppure dare un effetto differente.

Tutto questo dipende dal tipo di terminale scelto nel *database* di terminali: se questo è in grado di gestire il segnale generato dal tasto [*Canc*] della propria tastiera e dal significato che a questo gli può venire attribuito.

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 una 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 tipo di programma è dato dai cloni 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.

25.2.5 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, come 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 *database* di configurazione dei terminali, a nulla serve un **reset**.

Vedere: **tput(1)**, **clear(1)** e **reset(1)**.

25.3 Mouse

Il mouse, in un terminale a caratteri, non è una cosa tanto comune. È normale in un ambiente grafico, ma nel caso di Linux c'è la possibilità di usarlo nelle console virtuali.

[1] Per terminare l'utilizzo di **mc** si può scrivere semplicemente il comando **exit** seguito da [*Invio*].

Per gestire un mouse in un terminale a caratteri, è necessario un *daemon* 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 Linux è **`gpm`**. **`gpm`** ha in particolare il vantaggio di poter essere utilizzato anche con i programmi che non sono fatti per il mouse, per le funzioni di copia-incolla di testo.

25.3.1 # gpm

`gpm [`⟨*opzioni*⟩`]`

`gpm` è un programma *daemon* 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.

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).

`-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.
bm	BusMouse	Busmouse Microsoft e compatibili.
ps2	PS/2	Busmouse PS/2.
ncr		NCR3125pen.
wacom		Tavoletta Wacom.

Tabella 25.3: Elenco dei nomi di tipo 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 su primo e terzo tasto.

Utilizzo

Il funzionamento è relativamente semplice. Quando il mouse è controllato dal programma, dipende dal programma 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 al sistema 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.

Problemi

I problemi più gravi che si possono porre utilizzando **gpm** sono dovuti al possibile conflitto con X Window System, quando si utilizza un *bus* mouse, come nel caso del PS/2.

Quando si deve lavorare con X Window System e **gpm** non può conviverci, si pongono due possibili soluzioni: fare a meno di **gpm**, oppure interromperne il funzionamento prima dell'avvio di X Window System. La seconda soluzione potrebbe essere un problema se non si hanno i privilegi dell'utente *root*.

Esempi

`# gpm -t ps2`

Avvia **gpm** predisponendolo per utilizzare un mouse PS/2.

`# 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**.

25.4 Programmi getty

Il programma di gestione del terminale è quello che consente di collegarsi con il sistema operativo e di poter interagire con questo. Quello originariamente utilizzato per questo scopo è **getty**, ma quasi tutte le distribuzioni Linux preferiscono utilizzare programmi alternativi, come **agetty(1)**, **mingetty(8)** e **mgetty(8)**.

25.4.1 # getty

Si tratta del programma che attiva la gestione della console o del terminale. Questa è la versione normale. Vedere **getty(1)**.

25.4.2 # agetty

`agetty [`⟨*opzioni*⟩`]` ⟨*baud-rate*⟩`[,... ]` ⟨*porta*⟩ `[`⟨*variabile-term*⟩`]`

Si tratta di uno di quei programmi che attivano la gestione della console o del terminale. In pratica, è una di quelle versioni alternative al normale **getty**.

agetty apre una connessione con un terminale ed esegue il comando `/bin/login` che si occupa di richiedere il *login*, cioè il nome identificativo dell'utente, e la *password*. Di solito, **agetty** viene avviato direttamente da **init** durante il processo di inizializzazione del sistema.

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`.

⟨*baud-rate*⟩

È un elenco, separato da virgole, di una o più velocità di trasmissione espresse in baud. Ogni volta che **agetty** riceve un codice **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 **`getty`** 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`**.

25.4.3 /etc/issue

Il file `/etc/issue` viene usato per emettere un messaggio introduttivo prima della richiesta del *login*, da parte del programma **`getty`** o di un suo equivalente.

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 25.4 è adatto sia a **`agetty`** che a **`mingetty`**, mentre quello della tabella 25.5 è adatto solo a **`getty`**.

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 del computer: *hostname*.
\o	Inserisce il nome di dominio del computer.
\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 25.4: Elenco dei codici di *escape* utilizzabili all'interno del file `/etc/issue` quando si utilizzano i programmi **`agetty`** e **`mingetty`**.

Codice	Descrizione
\\	Inserisce la barra inclinata rovescia (\).
\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 *at* (`@`).
@@	Inserisce un simbolo *at* (`@`).
@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 del nodo di sistema.
@t	Inserisce l'orario.
@u	Inserisce il numero di utenti connessi.
@v	Inserisce la versione.

Tabella 25.5: Elenco dei codici di *escape* e dei parametri utilizzabili all'interno del file `/etc/issue` quando si utilizza il programma **`getty`**.

25.4.4 /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 26.2.5.

25.5 Monitoraggio di una sessione di lavoro

L'attività svolta durante una sessione di lavoro attraverso un terminale potrebbe essere volontariamente registrata in modo da annotare automaticamente le operazioni svolte, eventualmente anche a titolo di prova, come potrebbe essere l'esecuzione di un test di esame.

25.5.1 $ 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 digita ed emesso attraverso il terminale, viene memorizzato di 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 Strumenti per la gestione delle console virtuali

Le console virtuali di 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 possibili console virtuali, e quelle rimanenti potrebbero essere usate 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 di *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 in *background* (**`&`**) e l'output viene emesso attraverso la decima console virtuale (che si presume fosse precedentemente inutilizzata).

25.6.1 # 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. L'argomento, indica il numero della console virtuale da utilizzare (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, trattandola 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.

25.6.2 # 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 nella undicesima console virtuale.

Parte VII

Accounting

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.

<*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/*>

La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.

<*http://www.pluto.linux.it/ildp/AppuntiLinux/*>

<*http://ildp.psy.unipd.it/AppuntiLinux/*>

<*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

<*http://linux.calion.com/pluto/ildp/AppuntiLinux/*>

Registrazione e controllo degli eventi e degli accessi

In ogni sistema operativo multiutente c'è la necessità di controllare gli accessi, attraverso un sistema di registrazione degli utenti, e degli eventi, attraverso un sistema di *log*. Nei sistemi Unix un utente che può accedere ha un *account*: letteralmente, un conto.
La tabella 26.1 elenca i programmi e i file a cui si accenna in questo capitolo.

Nome	Descrizione
syslogd	*Daemon* per la registrazione del *log* di sistema.
/etc/syslog.conf	Configurazione di **`syslogd`**.
logger	Aggiunge delle annotazioni al *log* di sistema.
klogd	*Daemon* 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/⟨*utente*⟩	Posta dell'utente.
~/.hushlogin	*Login* 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.
last	Permette di analizzare il contenuto di `/var/log/wtmp`.
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.
whoami	Emette il nome dell'utente.
logname	Emette il nome dell'utente.

Tabella 26.1: Riepilogo dei programmi e dei file per la gestione della registrazione degli utenti e del controllo degli accessi.

26.1 Log di sistema

Il *log* di sistema, o *syslog*, è la procedura di registrazione degli eventi importanti all'interno di un cosiddetto file di *log*. Questo sistema è gestito principalmente dal *daemon* **`syslogd`**, che viene configurato attraverso `/etc/syslog.conf`. Altri programmi o *daemon* possono aggiungere annotazioni al *log* 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 Linux o dei sistemi Unix in generale, la consultazione dei file di *log* può essere di aiuto al principiante che si trova in difficoltà e non sa quale sia la causa del mancato funzionamento di qualcosa.

26.1.1 # syslogd

`syslogd [`⟨*opzioni*⟩`]`

È il *daemon* che si occupa della registrazione del *log* di sistema. Di norma viene avviato durante la fase di inizializzazione del sistema (*init*). Utilizza un file di configurazione che, se non specificato diversamente, 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 a **`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 *log socket* diverso da quello predefinito che è `/dev/log`.

26.1.2 /etc/syslog.conf

È il file di configurazione utilizzato da **`syslogd`** per definire in che modo devono essere gestiti i messaggi del sistema di *log*. Se si vogliono apportare modifiche a questo file, è poi 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, questo segnale può anche provocare la conclusione del funzionamento di questo 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* composte 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 una annotazione attraverso l'azione indicata nel secondo campo. Questo primo campo si divide in due sottocampi, uniti da un singolo punto (**`.`**): servizi e 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 una virgole (**`,`**).

Le parole chiave riferite alle priorità possono essere le 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 servizi.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.priorità, definiti 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 inclinata 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, o *fifo*, generata attraverso **`mkfifo`** (42.1.1).

- **`@`**

  Se il primo carattere è il simbolo *at* (**`@`**), si intente che la parte restante sia l'indicazione di un computer *host* 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 ad ora, 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 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 servizi.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, al computer dinkel.brot.dg
#
*.warning                  @dinkel.brot.dg
```

I messaggi il cui livello di priorità raggiunge o supera il livello **`warning`**, vengono inviati al computer *host* **`dinkel.brot.dg`**.

```
*.*                        @dinkel.brot.dg
*.=debug                   /var/log/debug
*.=info;*.=notice          /var/log/messages
*.warning                  /var/log/syslog
```

Invia tutti i messaggi al computer *host* **`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)`**.

26.1.3 Rotazione dei file del log di sistema

Per archiviare i file generati da **`kerneld`**, se la propria distribuzione Linux non gestisce già questo problema, si possono copiare i file di *log* altrove, eventualmente 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

26.1.4 Riservatezza delle informazioni

Le informazioni che vengono memorizzate nel *log* di sistema, potrebbero essere delicate, sia per la sicurezza del sistema, sia per i singoli utenti. Per questo, è bene ricordare che i file del *syslog* non dovrebbero essere accessibili in lettura agli utenti normali.

26.1.5 $ logger

`logger [`⟨*opzioni*⟩`] [`⟨*messaggio*⟩`]`

Permette di aggiungere delle annotazioni all'interno del *log* di 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 *log* di sistema.

26.1.6 # klogd

`klogd [`⟨*opzioni*⟩`]`

È il *daemon* specifico di Linux per l'intercettazione e la registrazione dei messaggi del *kernel*. Di norma viene avviato dalla procedura di inizializzazione del sistema, subito dopo **`syslogd`**.

Alcune opzioni

`-f` ⟨*file-di-log*⟩

Specifica un file di *log* particolare invece di dirigere i messaggi direttamente al *daemon* della gestione del *log* di sistema, cioè **`syslogd`**.

26.2 Login

Il *login* è la procedura attraverso la quale un utente, precedentemente registrato, 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.

26.2.1 $ login

`login [`⟨*utente*⟩`]`

Permette l'accesso dell'utente al sistema. Se si dispone di più di un *account*, lo si può utilizzare durante una sessione di lavoro, per accedere temporaneamente in modo diverso ed essere identificati diversamente.

Di solito non si usa direttamente: è compito del programma di gestione del terminale, **`getty`** o simili, di avviarlo e fare quindi in modo che la prima cosa che appare sullo schermo sia la richiesta del *login*.

Ogni utente registrato nel sistema, cioè ogni utente per il quale è stato previsto un *account*, 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 particolare configurazione dell'utente a cui appartiene. La directory *home* è collocata normalmente in `/home/`⟨*nome-utente*⟩`/` e questa, se la *shell* lo consente, viene abbreviata utilizzando il simbolo tilde (`~`). La directory *home* dell'utente *root* è speciale e dovrebbe trovarsi in `/root/`.

Durante un accesso normale da parte di un qualunque utente, 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 esiste della posta non letta);
- 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 massimo di dieci tentativi. Tutti gli errori sono riportati all'interno del *log* di 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 lo si intende chiudere. 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 registrati all'interno del *log* di sistema. Se esiste il file `~/.hushlogin`, viene eseguito un accesso silenzioso, nel senso che vengono disattivati:

- il controllo per la presenza di posta;
- 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 *home* dell'utente, viene utilizzata la directory radice (**`/`**).

> Quanto detto dovrebbe essere sufficiente per capire che la semplice rimozione della *shell* o della directory *home* da un *record* del file `/etc/passwd`, non è un sistema per impedire l'accesso a un utente.

26.2.2 /etc/passwd

È un elenco di utenti, *password*, directory *home*, *shell* e altre informazioni personali utilizzate da **`finger`** (67.3.2). La struttura dei *record* (righe) di questo file è molto semplice.

⟨*utente*⟩**:**⟨*password*⟩**:**⟨*UID*⟩**:**⟨*GID*⟩**:**⟨*dati-personali*⟩**:**⟨*directory-home*⟩**:**⟨*shell*⟩

Descrizione dei campi

⟨*utente*⟩
: È il nome utilizzato per identificare l'utente.

⟨*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.

⟨*UID*⟩
: È il numero identificativo dell'utente (*User ID*).

⟨*GID*⟩
: È il numero identificativo del gruppo a cui appartiene l'utente (*Group ID*).

⟨*dati-personali*⟩
: Di solito, questo campo contiene solo l'indicazione del nominativo completo dell'utente (nome e cognome), ma può contenere anche altre informazioni normalmente inserite attraverso **`chfn`** (27.1.6).

⟨*directory-home*⟩
: La directory assegnata all'utente.

⟨*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 *home* 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.

26.2.3 /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*⟩

Descrizione dei campi

⟨*gruppo*⟩

È il nome utilizzato per identificare il gruppo.

⟨*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`** (26.3.2).

⟨*GID*⟩

È il numero identificativo del gruppo.

⟨*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 ad esso appartiene solo un utente omonimo (**`tizio`** appunto).

```
users::100:tizio,caio,semproni
```

Al gruppo **`users`**, in questo caso, appartengono tre utenti: **`tizio`**, **`caio`** e **`semproni`**.

26.2.4 /etc/shadow

Il file `/etc/shadow` appare in quei sistemi in cui è attivata la gestione delle *shadow password*. 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.

26.2.5 /var/run/utmp

È il file che contiene l'elenco degli accessi in essere nel sistema. Non è un normale file di testo e per l'estrazione delle informazioni in esso contenute si usano *utility* apposite. È possibile tuttavia che siano di più gli utenti effettivamente presenti nel sistema 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 a prendersi cura di questo file, azzerandolo o ricreandolo, a seconda della necessità.

26.2.6 /var/log/wtmp

Il file `wtmp` ha una struttura analoga a quella di `/var/log/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 normale file di testo, 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
```

26.2.7 /etc/motd

Il contenuto di questo file viene visualizzato da **`login`** al termine della procedura di *login*, prima dell'avvio della *shell* associata all'utente. Questo file contiene, o dovrebbe contenere, il cosiddetto messaggio del giorno.

26.2.8 /etc/nologin

Se esiste questo file, **`login`** non accetta nuovi accessi al sistema, e a ogni tentativo visualizza il suo contenuto.
Se si desidera chiudere il sistema è possibile creare questo file scrivendoci all'interno il motivo, o una breve spiegazione di quello che sta avvenendo.

26.2.9 /etc/securetty

Contiene l'elenco dei ***terminali sicuri***, cioè di quelli da cui si permette l'accesso come 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, i sei terminali virtuali, quattro terminali seriali e quattro pseudo-terminali che accedono dalla rete locale oppure da X Window System.

```
console
tty1
tty2
tty3
tty4
tty5
tty6
ttyS0
ttyS1
ttyS2
ttyS3
ttyp0
ttyp1
ttyp2
ttyp3
```

26.2.10 /var/mail/⟨*utente*⟩

Il file corrispondente al nome dell'utente, contenuto in `/var/mail/`, viene usato per accumulare la posta a lui diretta.
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.

26.2.11 ~/.hushlogin

Se esiste il file `.hushlogin` all'interno della directory *home* di un certo utente, quando quell'utente accede, **`login`** non visualizza alcun messaggio introduttivo.

26.2.12 /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, a ogni *login*.
Se questo file non esiste, conviene crearlo manualmente in uno dei due modi seguenti.

```
# cp /dev/null /var/log/lastlog
# touch /var/log/lastlog
```

26.3 Cambiamento di identità

Alcuni programmi consentono di ottenere i privilegi di un altro utente, come se si eseguisse un altro *login*. Questa possibilità rappresenta generalmente un problema di sicurezza. Per mezzo di questi programmi, può capitare di riuscire ad ottenere i privilegi dell'utente *root* anche quando si accede da un terminale che non viene considerato sicuro, e quindi non incluso nell'elenco di `/etc/securetty`.

26.3.1 $ su

`su [`⟨*opzioni*⟩`] [`⟨*utente*⟩`]`

`su` permette a un utente di diventare temporaneamente un altro. Avvia una *shell* con UID e GID reale ed efficace. 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 *root*.
Per terminare l'attività come altro 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*.

26.3.2 $ newgrp

`newgrp [`⟨*gruppo*⟩`]`

Permette di cambiare il gruppo a cui appartiene l'utente. L'utente non cambia, la directory *home* nemmeno, cambia solo il GID. Un utente può cambiare gruppo solo se nel file `/etc/group` sono diversi i gruppi a cui può appartenere l'utente.
In alcuni sistemi Unix non è possibile cambiare gruppo se quel gruppo non ha una *password*.

26.4 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**.
Queste notizie possono essere utili anche per la preparazione di *script*, per esempio quando c'è la necessità di verificare se questo viene avviato dall'utente *root* o da un altro.

26.4.1 $ last

`last [`⟨*opzioni*⟩`] [`⟨*nome*⟩`... ]`

Visualizza il contenuto del file di *log* degli accessi (*login*) e disconnessioni (*logout*) per le informazioni riguardanti gli utenti e/o i terminali. Il file dal quale queste informazioni vengono attinte è `/var/log/wtmp`.

Alcune opzioni

`-`⟨*numero*⟩ | `-n` ⟨*numero*⟩

Limita il numero di elementi visualizzati allo specifico valore numerico indicato.

Esempi

$ ***last***

Visualizza gli ultimi eventi del *log* degli accessi.

$ ***last tizio root***

Visualizza gli accessi e disconnessioni da parte degli utenti **tizio** e **root**.

26.4.2 $ users

`users [`⟨*file*⟩`]`

Visualizza i nomi degli utenti che accedono attualmente al computer *host* corrente. Se un utente ha attivato più sessioni, 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`. Una possibilità comune è quella di visualizzare attraverso **users** il contenuto di `/etc/wtmp`.

26.4.3 $ w

`w [`⟨*opzioni*⟩`] [`⟨*utente*⟩`]`

Visualizza i nomi degli utenti che accedono attualmente e varie informazioni sulla loro attività.
Vedere **w(1)**.

26.4.4 $ 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 **info who** oppure **who(1)**.

26.4.5 $ whoami

`whoami`

Visualizza il nome dell'utente associato con l'attuale UID efficace. È equivalente a **`id -un`**.

26.4.6 $ logname

`logname`

Emette il nome dell'utente, così come appare dal file `/var/run/utmp`.

Account

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 26.2.2, 26.2.3 e 26.2.4.
La tabella 27.1 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 *home*.
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 27.1: Riepilogo dei programmi e dei file per la gestione della registrazione degli utenti.

27.1 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 del loro *account*. Il più importante è naturalmente la *password*.

27.1.1 # adduser | useradd

`adduser`
`useradd`

Il programma in questione può avere due nomi alternativi: **`adduser`** o **`useradd`**. Questo permette a *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 *home*.
Per convenzione, il programma (o *script* che sia) inserisce automaticamente nella directory *home* 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 *home* di ogni nuovo utente.

> Per mantenere la compatibilità con alcuni vecchi programmi, il nome dell'utente non deve superare gli otto caratteri.

27.1.2 /etc/skel/*

La directory `/etc/skel/` viene normalmente utilizzata come directory *home* tipica, per i nuovi utenti. In pratica, quando si aggiunge un nuovo utente e gli si prepara la sua directory *home*, 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 *home*.
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.

27.1.3 $ passwd

`passwd [`⟨*utente*⟩`]`

Permette di cambiare la *password* registrata all'interno di `/etc/passwd`. Solo *root* può cambiare la *password* di un altro utente. Gli utenti normali (tutti escluso *root*) devono utilizzare una *password* di almeno sei caratteri composti sia da maiuscole che minuscole o simboli diversi. Alcune *password* simili al nome utilizzato per identificare l'utente, non sono valide.
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`**.

27.1.4 $ 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 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`.

27.1.5 /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 esista una configurazione generalizzata per ognuna di queste, e che nella directory `/etc/skel/` siano stati predisposti tutti i file di configurazioni personalizzabili necessari. Quindi, un file `/etc/shells` con un semplice elenco di tutte le *shell* disponibili è troppo.

27.1.6 $ chfn

`chfn [⟨opzioni⟩] [⟨utente⟩]`

Consente di modificare le informazioni *finger* registrate all'interno del file `/etc/passwd`. Si tratta in pratica del nominativo 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 l'utente che ha eseguito il comando. Solo l'utente *root* può cambiare le informazioni di un altro utente.

Le informazioni *finger* sono solo una convenzione, attraverso cui si suddivide il campo relativo ai dati personali dell'utente.

Esempi

L'esempio seguente mostra le azioni compiute da un utente per definire le proprie informazioni *finger*.

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 *finger* possono essere delicate, specialmente quando si tratta di indicare il numero dell'abitazione di un utente. Per questo, quando si tratta di *account* presso computer raggiungibili attraverso una rete estesa, quale Internet per esempio, occorre prudenza.

27.1.7 $ 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⟩]`

.

27.1.8 $ id

`id [⟨opzioni⟩] [⟨utente⟩]`

Visualizza lo UID (*User ID*) e il GID (*Group ID*) reale e/o 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**.

27.2 Account importanti

Osservando il file `/etc/passwd` si possono notare diversi utenti fittizi standard che hanno degli scopi particolari.

```
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.

```
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 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 organizzato nella distribuzione RedHat. In generale, questo utente potrebbe avere il numero UID 65535 (−1), oppure 65534 (−2).

Segue la descrizione di alcuni di questi utenti e gruppi.

27.2.1 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.

27.2.2 bin

L'utente **`bin`** non esiste nella realtà. Si tratta di un nome fittizio definito per assegnare ai file eseguibili (*binary*) un proprietario diverso da *root*. Di solito, con Linux, questi eseguibili appartengono al gruppo **`bin`**, mentre l'utente proprietario resta *root*.

27.2.3 uucp

L'utente fittizio **`uucp`** viene spesso utilizzato come proprietario di file (soprattutto file di dispositivo) destinati alla comunicazione. Di solito, con Linux, si assegna il gruppo **`uucp`** ai dispositivi di comunicazione `/dev/cua*`, in modo da poter controllare gli utenti che accedono a queste risorse (tipicamente il modem), cambiando il gruppo di questi utenti in **`uucp`**, oppure permettendogli di cambiare gruppo attraverso **`newgrp`** (26.3.2).

27.2.4 tty

Di solito, al gruppo **`tty`** appartengono i file di dispositivo utilizzabili come canali per la connessione di un terminale.

27.2.5 disk

Di solito, al gruppo **`disk`** appartengono i file di dispositivo che si riferiscono a unità a dischi, compresi i CD-ROM.

27.2.6 floppy

Di solito, al gruppo **`floppy`** appartengono i file di dispositivo che si riferiscono alle unità a dischetti.

27.2.7 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, il computer che offre il servizio NFS lo tratta come utente **`nobody`**.

Di solito, per tradizione, questo utente dovrebbe avere il numero UID −1 (65535), oppure −2 (65534), ma alcune distribuzioni Linux attribuiscono a questo un altro numero (nell'esempio visto sopra era 99).

27.3 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 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 dal file `/etc/passwd` e dal file `/etc/group`;
- l'eliminazione dei file appartenenti a quell'utente.

27.3.1 Collocazione dei dati dell'utente

I file di un utente possono trovarsi ovunque gli sia stato consentito di scriverli. In particolare:

- la directory *home* per i suoi file personali;
- la directory `/var/mail/` per la posta;
- la directory `/var/spool/cron/` e `/var/spool/at/` per eventuali applicazioni a esecuzione pianificata.

Per cercare tutti i file appartenenti a un certo utente, è possibile usare il programma **`find`** in uno dei modi seguenti.

find / -uid ⟨*numero-utente*⟩

find / -user ⟨*utente*⟩

27.3.2 Eliminazione dei dati dell'utente

Volendo, si potrebbe costruire uno *script* per l'eliminazione automatica di tutti i file appartenenti a un determinato utente. 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
        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 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 \( -path "/dev" -prune -o -print \) \
> $ELENCO_FILE_UTENTE
    #-----------------------------------------------------------------
    # Comprime i file generando un file compresso con lo stesso nome
    # dell'utente da eliminare e con estensione ``.tgz''.
    # Si utilizza ``tar'' e in particolare:
    # ``-z'' permette di comprimere automaticamente l'archivio
    #        attraverso ``gzip'';
    #-----------------------------------------------------------------
    if tar -c -f ~/$NOME_UTENTE.tgz -v -z `cat $ELENCO_FILE_UTENTE`
    then
        #-------------------------------------------------------------
        # Se è andato tutto bene, elimina i file
        # (togliere il commento).
        #-------------------------------------------------------------
        #rm `cat $ELENCO_FILE_UTENTE`
        echo
    fi

#=======================================================================
# Fine.
#=======================================================================
```

27.4 Trucchi

Alcuni accorgimenti nella gestione degli utenti e dei gruppi possono essere utili in situazioni particolari. Nelle sezioni seguenti se ne descrivono alcuni.

27.4.1 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*.

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/`.

```
ciao::505:100:Ciao a tutti:/tmp:/usr/local/bin/ciao
```

Naturalmente, il numero UID 505 e GID 100 sono solo un esempio.

27.4.2 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:34gdf6r:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
bravo:e445gsdfr:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
charlie:t654df7u7:1000:1000:Gruppo di lavoro:/home/tutti:/bin/sh
```

Se esiste la necessità o l'utilità si possono assegnare anche directory *home* e/o *shell* differenti.

27.4.3 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 lo stesso numero (ID).
Questa tecnica si combina con una maschera *umask* **`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.
Questo metodo di comportamento è quello predefinito della distribuzione RedHat.

27.4.3.1 Esempio di gestione dei gruppi privati

Si vuole 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 al gruppo delle directory stesse, cioè quello del progetto per cui sono state predisposte.

Parte VIII

Shell (bash)

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.
<*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
<*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>
<*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>
<*ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/*>
<*ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/*>
<*ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>
<*ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/*>
La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.
<*http://www.pluto.linux.it/ildp/AppuntiLinux/*>
<*http://ildp.psy.unipd.it/AppuntiLinux/*>
<*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>
<*http://linux.calion.com/pluto/ildp/AppuntiLinux/*>

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*.

28.1 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.

28.2 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*.

28.2.1 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 dalla apparizione sullo schermo del terminale di un messaggio di invito o *prompt*. Questo, è per lo più composto di simboli e informazioni utili all'utente per tenere d'occhio il contesto su 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 una interazione con l'utente attraverso l'inserimento di comandi.

28.2.2 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.

28.2.3 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 fastidiosi equivoci: alle volte non si è in grado di capire il motivo per il quale un certo programma non funziona esattamente come ci si aspetterebbe.

28.2.4 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`**.

28.2.5 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 loro contenuto modificano 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*.

28.2.6 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*.

28.2.7 Script

Con il termine *script* si identifica un programma scritto ed eseguito nella sua forma sorgente senza l'intervento di alcuna compilazione.
Le *shell* sono normalmente 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.

28.2.8 Sostituzione / 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*.

28.2.8.1 Caratteri jolly o globbing

I caratteri jolly 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/o directory che si ottengono dall'espansione di questi simboli.
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.

28.2.8.2 Variabili e parametri

Come accennato, le *shell* permettono di creare o modificare il contenuto di variabili di ambiente. Queste variabili possono essere utilizzare 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*.

28.2.8.3 Sostituzione di comandi

Le *shell* più recenti consentono di fare generare 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.

28.2.8.4 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 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 utilizzare una *shell* particolare, ci si adatta a utilizzare una serie di tecniche consuete, perdendo di vista la sintassi vera dei comandi.

28.2.9 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 normalmente passati 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 analizzare 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 cosiddette ***parole***.

La conoscenza del modo in cui una *shell* suddivide una riga di comando in parole, è molto importante.

Bash: avvio e conclusione

`bash` è la *shell* standard di Linux e di molti altri sistemi Unix. **`bash`** (*Bourne Again SHell*) è compatibile con **`sh`** (Bourne Shell) e incorpora alcune funzionalità di **`ksh`** (Korn Shell) e **`csh`** (C Shell). **`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: **`info bash`** e **`bash(1)`**.

29.1 Avvio di bash

`bash` [⟨*opzioni*⟩] [⟨*file-script*⟩] [⟨*argomenti*⟩]

Si distinguono fondamentalmente due tipi di modalità di funzionamento di **`bash`**: interattiva e non interattiva. Quando viene avviata con l'indicazione del nome di un file, la *shell* **`bash`** tenta di eseguirlo come uno *script* (in tal caso non conta che il file sia eseguibile 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).

`bash` è potenzialmente compatibile con diversi 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 29.1 mostra, in particolare, la sequenza di file di configurazione utilizzati a seconda del tipo di emulazione preferito.

29.1.1 Shell interattiva

La *shell* è interattiva quando interagisce con l'utente, e di conseguenza mostra un *prompt*. **`bash`** può eventualmente essere avviata in modo esplicitamente interattivo utilizzando l'opzione **`-i`**.[1]

Quando **`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.

29.1.1.1 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*.

`bash`, nel momento dell'avvenuto *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 dell'esecuzione:

- se esiste, legge ed esegue il contenuto di `~/.bash_logout`.

29.1.1.2 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`.

29.1.1.3 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
...
```

[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.

Il significato è semplice: viene controllata l'esistenza del file `~/.bashrc` e se viene trovato viene caricato ed eseguito.

29.1.2 Shell non interattiva

Una *shell* non interattiva è di norma dedicata a eseguire uno *script*. Nel momento dell'avviamento in questa modalità, **`bash`** controlla il contenuto della variabile di ambiente **`BASH_ENV`** e se 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 da **`bash`** prima di eseguire lo *script*. In situazioni normali questa variabile è solitamente vuota, o inesistente.

29.1.3 Compatibilità con sh e POSIX

Se **`bash`** viene avviata come **`sh`** (per esempio attraverso un *link* simbolico), si comporta come la *shell* Bourne, per quanto riguarda l'utilizzo dei file di configurazione, e poi 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 **`bash`** viene avviata in modalità POSIX, attraverso l'opzione **`--posix`**, 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 29.1: A seconda del modo con cui **`bash`** viene avviata si utilizzano diversi tipi di file di configurazione.

29.2 Opzioni

`bash` interpreta due tipi di opzioni: a singolo carattere e multicarattere. Le opzioni multicarattere devono necessariamente precedere quelle a singolo carattere.

Alcune opzioni multicarattere

`--norc`
: Riguarda la modalità interattiva: non esegue il file di configurazione `~/.bashrc`. Quando si avvia **`bash`** utilizzando il nome **`sh`** per mantenere la compatibilità con la *shell* Bourne, questo file di configurazione non deve essere letto, quindi in questo caso, quest'opzione è predefinita.

`--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 **`bash`** funzioni come se fosse stata avviata come *shell* di *login*.

`--noediting`
: Quando è avviata in modalità interattiva, non usa la libreria GNU **`readline`** per leggere le linee di comando.

`--posix`
: Fa in modo che **`bash`** si adegui 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*.

29.3 Interpretazione degli argomenti successivi

Se restano degli argomenti dopo le opzioni, e non sono state usate le opzioni **`-c`** o **`-s`**, il primo degli argomenti viene interpretato come nome di un file di comandi di *shell*: uno *script* di *shell*. Se **`bash`** 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. **`bash`** legge ed esegue i comandi di questo *script* e quindi termina l'esecuzione. Il valore restituito da **`bash`** alla fine della sua esecuzione è quello dell'ultimo comando eseguito dallo *script*.

29.4 Uso sommario della tastiera

`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 29.2 mostra un elenco delle funzionalità dei tasti e 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 29.2: Elenco delle funzionalità dei tasti e delle combinazioni più importanti di **`bash`**.

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 precedentemente inserite. 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, **`info bash`** e **`bash(1)`**, in corrispondenza della libreria **`readline`**.

29.5 Invito o prompt

Quando **`bash`** 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 contenere alcuni caratteri speciali contrassegnati dal carattere di *escape* (\), che acquistano il significato indicato qui di seguito.

Codici speciali per il prompt

`\t`

Rappresenta l'orario attuale nel formato hh:mm:ss (ore, minuti, secondi).

`\d`

Rappresenta la data nel formato: "settimana mese giorno" (per esempio: **`Tue Apr 15`**).

`\n`
Rappresenta il codice *newline*.

`\s`
Corrisponde al nome della *shell*.

`\w`
Corrisponde alla directory corrente.

`\W`
Corrisponde al nome base (*basename*) della directory corrente.

`\u`
Corrisponde al nome dell'utente.

`\h`
Corrisponde al nome del computer (*host*).

`\#`
Corrisponde al numero del comando attuale.

`\!`
Corrisponde al numero del comando nello storico.

`\$`
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 **$**.

`\`*`nnn`*
Rappresenta un carattere corrispondente al numero ottale fornito dopo la barra.

`\\`
Rappresenta una barra inclinata rovescia, \.

`\[`
Rappresenta l'inizio di una serie di caratteri non stampabili. Questo metodo di delimitazione viene utilizzato per incorporare una sequenza di controllo per il terminale.

`\]`
Rappresenta la fine di una una serie di caratteri non stampabili.

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, della 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 del computer, la directory corrente e il simbolo $ o # a seconda del tipo di utente.

```
daniele@dinkel:~$
```

29.6 Conclusione

La conclusione del funzionamento di **`bash`** 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 ad un procedimento di sostituzione: quando la *shell* ha terminato l'interpretazione di una istruzione, questi devono essere rimossi in modo da non lasciare tracce per un eventuale programma che dovesse ricevere questi dati in forma di argomenti.

30.1 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 rovescia), gli apici semplici e gli apici doppi (o virgolette).

30.1.1 Escape e continuazione

La barra inclinata rovescia (\) rappresenta il carattere di *escape*. Serve per preservare il carattere seguente, 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 immediatamente seguito dal codice *newline*: rappresenta una continuazione nella riga successiva.[1]

> Il simbolo \ utilizzato per interrompere la riga 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 rovescia 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.

30.1.2 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 da al codice *newline*.

30.1.3 Apici doppi

Racchiudendo una serie di caratteri tra una coppia di doppi apici 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 doppi apici, mentre la barra obliqua rovescia (\) 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"
```

30.2 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.

30.2.1 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 **`$`** che di per se non è parte del nome del parametro. Tuttavia, per maggiore chiarezza espressiva, dal momento che non è possibile assegnare un valore a queste entità, quando nelle documentazioni si fa riferimento a un parametro, si utilizza quasi sempre il suo nome (o il simbolo che rappresenta) preceduto dal simbolo **`$`**.
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.

30.2.1.1 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, **`$2`** è il secondo, e così di seguito.
Quando viene eseguita una funzione, questi parametri vengono temporaneamente rimpiazzati con gli argomenti forniti alla funzione.
Quando si utilizza un parametro composto da più di una cifra numerica, occorre racchiudere questo numero tra parentesi graffe: **`${10}`**, **`${11}`**,...

30.2.1.2 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 **`bash`** viene avviata con un file di comandi, **`$0`** conterrà il nome di quel file. Se **`bash`** 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 doppi apici, rappresenta una singola 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 un singolo spazio. 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 normalmente utilizzato il carattere <*Spazio*> per staccare i vari parametri posizionali. Per cui, in pratica, la maggior parte delle volte, **"$*"** equivale a **"$1 $2..."**.

- **$@**
  Rappresenta l'insieme di tutti i parametri posizionali a partire dal primo. Quando viene utilizzato all'interno di doppi apici, rappresenta una serie di parole, ognuna composta dal contenuto del rispettivo parametro posizionale. Di conseguenza, **"$@"** equivale a **"$1" "$2"....** Questo rappresenta una eccezione rispetto agli altri parametri: l'espansione di questo parametro genera diverse parole.
- **$#**
  Restituisce il numero della quantità di parametri posizionali.
- **$?**
  Restituisce lo stato dell'ultima *pipeline* eseguita in *foreground* (primo piano).
- **$-**
  Il trattino, restituisce i *flag* di opzione.
- **$$**
  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.

30.2.2 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 30.1 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 30.2 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 (**:**). Il valore predefinito dipende dalla configurazione di **bash** 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.

- **MAILPATH**
  Rappresenta il vari *pathname* dei file che possono contenere posta. Oltre al percorso di ricerca può essere fornito anche il messaggio che si vuole vedere in caso di presenza di posta. Il simbolo **?** viene utilizzato per indicare l'inizio del messaggio; il parametro **$_** viene utilizzato come sostituto del file di posta corrente. Segue un esempio, in cui si verificano due file: `/var/mail/tizio` e ~`/posta`. A seconda del file che ne contiene, viene visualizzato un messaggio differente.

	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 precedentemente visitata.
"	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 computer.
"	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 di **`bash`**.
"	PPID	Il PID del processo genitore della *shell* attuale.
"	SHLVL	Il livello di annidamento di **`bash`**.

Tabella 30.1: Elenco delle variabili di **`bash`** più importanti, 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 *home* dell'utente.	
"	CDPATH	Il percorso di ricerca per il comando **`cd`**.	
"	MAILPATH	La directory dei file della posta.	
"	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 30.2: Elenco di alcune delle altre variabili utilizzate da **`bash`**. L'elenco è classificato in base all'origine storica.

```
MAILPATH='/var/mail/tizio?"Posta":~/posta?"$_ ha della posta"'
```

- **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 i doppi apici, si deve di fatto trasformare in `"\\u@\\h:\\w\\$ "`.

30.2.2.1 Esportazione

Quando si creano e/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.

30.3 Espansione

Con questo termine si intende la traduzione di parametri e 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 *pathname* può cambiare il numero delle parole di una 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.

30.3.1 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.

30.3.2 Espansione delle parentesi graffe

L'espansione delle parentesi graffe deriva da **`csh`**.

⟨*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/bash/{old,new,dist,bugs}***

Crea quattro directory: **old**, **new**, **dist** e **bugs** a partire da `/usr/local/src/bash/`.

\# ***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.

30.3.3 Espansione della tilde

L'espansione della tilde deriva da **csh**.

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 *home* dell'utente stesso. Se invece, dopo il carattere ~ c'è subito la barra, o nessun altro carattere, si intende il contenuto della variabile **HOME**, ovvero la directory *home* dell'utente attuale.

Esempi

$ ***cd ~***

Corrisponde a uno spostamento nella directory *home* dell'utente.

$ ***cd ~tizio***

Corrisponde a uno spostamento nella directory *home* 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**.

30.3.4 Espansione di parametri e variabili

Il modo normale in cui si fa riferimento a un parametro o 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.

I modi in cui con **bash** è possibile ottenere la sostituzione di parametri e variabili sono numerosi. Questi sono generalmente derivati da **ksh** e sono descritti nella sezione 32.4.

30.3.5 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 da **ksh** e la seconda originaria di **sh**.

$ (⟨*comando*⟩)

$ **`**⟨*comando*⟩ **`**

Nel secondo caso dove si utilizzano gli apici inversi, la barra inclinata rovescia (\) eventualmente contenuta 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 inclinata rovescia.
Se la sostituzione appare tra doppi apici, la suddivisione in parole e l'espansione di *pathname* 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.

30.3.6 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 trattato in modo speciale. Tutti gli elementi all'interno della 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, **bash** segnala l'errore e non esegue la 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**.

30.3.7 Suddivisione di parola

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 doppi apici.
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 della 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, la suddivisione in parole di quanto contenuto tra la coppia di doppi apici, non avviene e di conseguenza, non può avvenire la successiva espansione di *pathname*.

```
b* d*
```

/$ ***echo '$Pippo'***[*Invio*]

Se si utilizzano gli apici semplici, non avviene alcuna sostituzione della variabile **Pippo**.

```
$Pippo
```

30.3.8 Espansione di pathname

Dopo la suddivisione in parole, **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 *pathname* corrispondenti al modello. Se non si ottiene alcuna corrispondenza, il comportamento predefinito di **bash** è tale per cui la parola resta immutata, consentendo quindi l'utilizzo dei caratteri jolly per il *globbing* per identificare un *pathname*.[2]

Per convenzione, si considerano nascosti i file e le directory che iniziano con un punto. Di conseguenza, normalmente, i caratteri jolly non permettono di includere i nomi che iniziano con tale punto. Perciò, se necessario, questo punto deve essere indicato espressamente.

La barra inclinata di separazione dei *pathname* non viene mai generata automaticamente da un *globbing*.

Caratteri jolly

`*`

Corrisponde a qualsiasi stringa, compresa la stringa vuota.

`?`

Corrisponde a un qualsiasi carattere (uno solo).

`[...]`

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.

30.4 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 inclinata rovescia o di virgolette di qualche tipo, vengono rimossi.

---

[2] In generale, sarebbe meglio essere precisi quando si vuole indicare espressamente un nome che effettivamente contiene un asterisco o un punto interrogativo: si deve usare la barra inclinata rovescia che funge da carattere di *escape*.

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 degli 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.

31.1 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 stato 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 una esecuzione di un comando viene utilizzato in una 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 **`bash`** ad assegnare 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, **`bash`** 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*).

31.2 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*).

31.3 Lista di comandi

La lista di comandi è una sequenza di una o più *pipeline* separate da **`;`**, **`&`**, **`&&`** o ||, e terminata da **`;`**, **`&`** o *newline*. 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.
Segue l'elenco degli operatori con la descrizione del loro significato.

- **`;`**
  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, e/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 *newline*.

Esempi

`# ./config ; make ; make install`

Avvia in sequenza una serie di comandi per la compilazione e installazione di un ipotetico programma.

`$ 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 non comporta alcuna differenza di comportamento.

Di seguito si vedono due spezzoni di *script* equivalenti: nel secondo si sostituisce il punto e virgola con un codice *newline*, dato che il contesto lo consente.

```
ls ; echo "Ciao a tutti"
```

```
ls
echo "Ciao a tutti"
```

- **&&**

  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.

  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.

- ||

  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.

  Esempi

  `$ mkdir ./prova || mkdir ./prova1`

  Si tenta di creare la directory `prova/`, se il comando fallisce si tenta di creare `prova1/` al suo posto.

- **&**

  I comandi seguiti dal simbolo **&** vengono messi in esecuzione in *background*. La descrizione del meccanismo con cui i programmi possono essere messi e gestiti in *background* viene fatta nella sezione 31.6. 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 in *background* 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** in *background*.

  `# gpm -t ms &`

  Avvia il programma **gpm** di gestione del mouse in *background*.

- **( )**

  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.

- `{ }`

  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 *shell*. Gli esempi seguenti sono equivalenti.

```
#!/bin/bash
{  mkdir ./prova ; cd ./prova ; ls ; }
```

```
#!/bin/bash
{  mkdir ./prova
   cd ./prova
   ls
}
```

31.4 Alias

La gestione degli alias deriva da **`ksh`**.

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, **`bash`** cerca la prima parola di questo comando (quello che lo identifica) all'interno dell'elenco degli alias e, 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* **`csh`**, 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 di **`bash`**, 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`** 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.

31.5 Ridirezione

Prima che un comando sia eseguito, si può ridirigere il suo input e/o 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, **bash** emette un errore.

31.5.1 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*.

31.5.2 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 esplicitamente indicato il descrittore dello *standard input*.

31.5.3 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** 33.1.36). 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.

31.5.4 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.

31.5.5 Ridirezione simultanea di standard output e standard error

`&>` ⟨*parola*⟩

`>&` ⟨*parola*⟩

`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.

31.5.6 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 indicata 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 verrà espanso. Altrimenti, il testo viene espanso come di consueto.

Per maggiori dettagli conviene consultare la documentazione interna: **`info bash`** oppure **`bash(1)`**.

31.5.7 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*.

31.5.8 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.

31.5.9 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*.

31.5.10 Ridirezione e script

Lo *standard input* di uno *script* è quello del 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 ad un testo stampabile che viene inviato alla stampante predefinita.

```
#!/bin/bash
#======================================================================
# /var/spool/text/input-filter
#======================================================================
cat | /usr/bin/unix2dos | lpr
```

31.5.11 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 nello *standard output*.

```
#!/bin/bash
function ordina() {
        sort
}
ordina < ./elenco > ./elenco_ordinato
```

31.6 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* attualmente in esecuzione che può essere letta attraverso il comando interno **`jobs`**. Quando **`bash`** avvia un processo in modo asincrono (ovvero in *background*), 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*:

- *foreground* o in primo piano;
- *background* o asincroni, o in sottofondo.

Un *job* è in *foreground* quando utilizza la tastiera e il video del terminale che si sta utilizzando. Un *job* è in *background* quando lavora in modo indipendente e asincrono rispetto all'attività dal terminale.

Un *job* in esecuzione in *foreground* 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* di **`bash`**. In alternativa si può sospendere un *job* in esecuzione in *foreground*, 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* di **`bash`**, 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 l'esecuzione del *job* sospeso in *background*, mentre **`fg`** consente di fare riprendere l'esecuzione del *job* sospeso in *foreground*. **`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, **`bash`** segnala un 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, **`bash`** segnala un errore.

`%%` | `%+`

Le notazioni **`%%`** o **`%+`** fanno riferimento al *job* corrente dal punto di vista della *shell*, che corrisponde all'ultimo *job* sospeso quando era in *foreground*.

`%-`

La notazione **`%-`** fa riferimento al penultimo *job* sospeso.

Il controllo dei *job* è descritto anche nella sezione 22.1.

Esempi

`$ fg %1`

Porta in *foreground* il *job* numero 1.

`$ %1`

Porta in *foreground* il *job* numero 1.

`$ bg %1`

Porta in *background* il *job* numero 1.

`$ %1 &`

Porta in *background* il *job* numero 1.

31.7 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 (**`/`**), ci si trova di fronte a un *pathname* e, di conseguenza, il comando si riferisce precisamente a un file eseguibile e 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, **`bash`** emette un messaggio di errore e restituisce un valore di uscita diverso da zero.

Quando **`bash`** ha determinato che si tratta di un eseguibile esterno ed è riuscita a trovarlo, vengono eseguite le seguenti azioni.

- 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 eventualmente forniti 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* ed eventuali altri argomenti ricevuti nella riga di comando originale.

31.8 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* che quindi, ereditano 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 **`mioprogramma`** in *background* con un diverso percorso di ricerca, senza però influenzare lo stato generale della configurazione di ambiente della *shell*.

```
$ PATH=/bin:/sbin mioprogramma &
```

Bash: programmazione

La programmazione con **`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.

32.1 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 **`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* di **`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.

32.1.1 Commenti

Come appena accennato, il simbolo **`#`** introduce un commento, fino alla fine della riga, cioè qualcosa che non ha alcun valore per l'interprete.

32.2 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 *newline*.

32.2.1 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`

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 -c -f ${DA_ARCHIVIARE}.tgz -v -z -p $DA_ARCHIVIARE
done
```

32.2.2 select

Il comando **`select`** permette all'utente di effettuare una scelta inserendo un valore attraverso la tastiera. **`select`** è stato ereditato da **`ksh`**.

`select` ⟨*variabile*⟩ `[ in` ⟨*parola*⟩`... ]` **`;`** `do` ⟨*lista-di-comandi*⟩ **`;`** `done`

L'elenco di parole che segue **`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
```

32.2.3 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`

La parola che segue **`case`** viene espansa e quindi confrontata con ognuno dei modelli, usando le stesse regole della espansione di *pathname*. 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 viene controllato nessuno 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.

32.2.4 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
```

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 emette 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
```

32.2.5 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`

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`**.

32.2.6 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`

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).

32.2.7 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 normalmente impiegate all'interno di file *script*.

`[ function ]` ⟨*nome*⟩ `()` { ⟨*lista*⟩`...` **`;`** }

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 la 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 Letter."
}
```

Nell'esempio seguente, si utilizza il comando **`return`** per fare in modo che l'esecuzione della funzione termini in un determinato punto 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
```

32.3 Espressioni aritmetiche

La *shell* consente di risolvere delle espressioni aritmetiche in certe circostanze. Il calcolo avviene su interi di tipo *long* 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 32.1 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 32.1: Operatori aritmetici di **`bash`**.

La tabella 32.2 riporta l'elenco degli operatori di confronto.

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 32.2: Operatori di confronto.

La tabella 32.3 riporta l'elenco degli operatori logici.

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 32.3: Operatori logici.

La tabella 32.4 riporta l'elenco degli operatori binari.

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*⟩	Inversione binaria.
⟨*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 32.4: Operatori binari.

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 una 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.

32.4 Riferimenti particolari alle variabili

L'espansione normale delle variabili è già stata vista nella sezione 30.3.4, ma **`bash`** offre in particolare dei modi alternativi, derivati da **`ksh`**, 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.

32.4.1 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
```

32.4.2 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
```

32.4.3 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
```

32.4.4 Lunghezza del contenuto

Questo tipo di sostituzione riguarda solo **`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
```

32.4.5 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.

32.5 Array

Oltre alle normali variabili scalari, 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* di **`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`** e/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 della 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.

32.5.1 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 della istruzione).
${⟨*array*⟩[⟨*indice*⟩]}

Se si legge un *array* come se fosse una normale variabile scalare, 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*.

32.5.2 Lettura della dimensione

Per ottenere la dimensione di un *array* si utilizza una delle due notazioni seguenti (le parentesi quadre fanno parte della istruzione).

`${#`⟨*array*⟩`[*]}`
`${#`⟨*array*⟩`[@]}`

32.5.3 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*.

Bash: comandi interni

I comandi interni sono quelli eseguiti direttamente dalla *shell*, come se si trattasse di funzioni. La tabella 33.1 mostra l'elenco 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 *pathname* 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 una espressione condizionale.
"	times	Emette i tempi di utilizzo accumulati.
"	trap	Specifica i comandi da eseguire quando la *shell* riceve segnali.
"	umask	Determina la maschera di creazione dei file.
"	unset	Elimina le variabili.
"	wait	Attende la conclusione dei processi figli.
csh	pushd	Salva la directory corrente.
"	popd	Estrae una directory precedentemente salvata.
"	dirs	Elenca le directory memorizzate.
"	history	Emette l'elenco dei comandi memorizzati nello storico.
"	logout	Termina l'esecuzione di una *shell* di *login*
"	source	Esegue la stessa funzione del singolo punto (`.`).
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 output*.
"	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.
"	shopt	Attiva o disattiva alcune opzioni di *shell*.

Tabella 33.1: Comandi interni di **`bash`**.

33.1 Comandi interni in ordine alfabetico

Nelle sezioni seguenti sono descritti i comandi interni di **`bash`** in ordine alfabetico.

33.1.1 :

`:` [⟨*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.

33.1.2 . | 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 all'interno della 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 dal suo interno;
- se non vengono eseguiti comandi, il valore è zero;
- se il file non è stato trovato, è *Falso*.

33.1.3 alias

`alias [`⟨*nome*⟩`[=`⟨*valore*⟩`]... ]`

Il comando **`alias`** senza argomenti, emette attraverso lo *standard output* la lista degli alias nella forma ⟨*nome*⟩**=**⟨*valore*⟩. Se non viene indicato il valore dell'alias, ne viene emesso il nome e il contenuto. La coppia ⟨*nome*⟩**=**⟨*valore*⟩ deve essere scritta senza lasciare spazi prima e dopo del segno **=**. 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*.

33.1.4 bg

`bg [`⟨*specificazione-del-job*⟩`]`

Mette in *background* il *job* indicato, come se fosse stato avviato con un **&** finale. Se non viene indicato il *job*, viene messo in *background* 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 31.6.

33.1.5 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 di quella utilizzata per il file di configurazione `~/.inputrc`.

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 non siano fornite opzioni sconosciute.

33.1.6 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.

33.1.7 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.

33.1.8 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 contenere 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 semplicemente riferimento alla directory corrente.

33.1.9 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.

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 che 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, viene restituito il valore restituito dal programma stesso.

33.1.10 continue

`continue [`*n*`]`

Riprende, a partire dalla successiva iterazione, 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.

33.1.11 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 quelli esistenti con i loro valori.

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 attuale sessione di funzionamento della *shell*. Si dice che le variabili vengono esportate.

`-i`

La variabile viene trattata come un intero e 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.

33.1.12 dirs

`dirs [-l] [+/-`*n*`]`
Visualizza l'elenco delle directory memorizzate nello *stack* apposito. Per salvare una directory nello *stack* si utilizza il comando **`pushd`**, mentre per tornare alla directory precedentemente salvata si utilizza il comando **`popd`**.

Opzioni

`+`*n*
Mostra solo l'*n*-esimo elemento dello *stack* a partire dall'alto. L'ultimo elemento inserito si raggiunge con **`+0`**.

`-`*n*
Mostra solo l'*n*-esimo elemento dello *stack* a partire dal basso. Il primo elemento inserito si raggiunge con **`-`**.

`-l`
Mostra un elenco lungo senza abbreviare i percorsi con il simbolo tilde (`~`).

`dirs` restituisce zero se non sono stati commessi degli errori.

33.1.13 echo

`echo [-neE] [`⟨*argomento*⟩`... ]`
Emette gli argomenti separati da uno spazio. Restituisce sempre il valore zero.

Opzioni

`-n`
Sopprime il codice *newline* 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 *newline* finali;
`\`**`f`** *form feed* o <*FF*>;
`\`**`n`** *newline* o <*LF*>;
`\`**`r`** *carriage return* o <*CR*>;
`\`**`t`** tabulazione (orizzontale) o <*HT*>;
`\`**`v`** tabulazione verticale o <*VT*>;
`\\` barra inclinata rovescia;
`\`***`ooo`*** il carattere il cui codice ottale corrisponde al numero indicato da *ooo*.

33.1.14 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.

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 se non è stato fornito un nome di comando interno inesistente.

33.1.15 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*.

33.1.16 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*.

33.1.17 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.

33.1.18 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.

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.

33.1.19 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.

33.1.20 fg

`fg [`⟨*job*⟩`]`

Pone il *job* indicato in *foreground*, ovvero in primo piano. Se non viene specificato il *job*, si intende quello attuale, ovvero, l'ultimo a essere stato messo in *background*.
Il controllo sui *job* è descritto nella sezione 31.6.

33.1.21 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 un `:`, 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 una 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 una 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 `:`.

Se viene incontrata una 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 un semplice *script* 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 varie 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
```

33.1.22 hash

`hash [-r] [`⟨*comando*⟩`... ]`

Per ciascun comando indicato, viene determinato e memorizzato il percorso completo.

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.

33.1.23 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 emessa una lista dei comandi interni.

Il valore restituito è zero a meno che il modello fornito non combaci con alcun comando.

33.1.24 history

`history [`*n*`]`

`history -rwan [`⟨*file-storico*⟩`]`

Se non viene indicata alcuna opzione, emette la lista dello storico dei comandi con i numeri di riga. Le righe segnate con un asterisco sono state modificate. Con argomento *n*, vengono elencate solo le ultime *n* righe. Se viene fornito un argomento che non sia un'opzione, questo viene usato come nome del file storico, altrimenti viene usato il valore di **`HISTFILE`**. Le opzioni, se fornite, hanno i seguenti significati:

Opzioni

`-a`

Aggiunge le righe di storico nuove, ovvero quelle inserite dall'inizio della sessione corrente di **`bash`**, al file storico.

`-n`

Inserisce le righe dello storico non ancora lette dal file storico, nella lista dello storico corrente. Queste sono le linee aggiunte al file storico dall'inizio della sessione corrente di **`bash`**.

`-r`

Legge il contenuto del file storico e lo usa come storico corrente.

`-w`

Scrive lo storico corrente sul file storico sovrascrivendo il contenuto del file stesso.

Restituisce il valore zero a meno che siano stati commessi degli errori oppure ci siano stati dei problemi di lettura o scrittura con il file storico.

33.1.25 jobs

`jobs [-lnp] [`⟨*job*⟩`... ]`

`jobs [-x` ⟨*comando*⟩ `[`⟨*argomenti*⟩`... ]`

La prima forma elenca i *job* attivi.

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 una opzione non ammessa o sia stata fornita l'indicazione di un *job* impossibile.
Il controllo sui *job* è descritto nella sezione 31.6.

33.1.26 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.

33.1.27 let

`let` ⟨*argomento*⟩ [⟨*argomento*⟩...]

Permette di eseguire operazioni aritmetiche con l'utilizzo di variabili. Ogni argomento è una 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 la 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 contentuo della **PIPPO1** con quello di **PIPPO2** e assegna il risultato alla **PIPPO1** stessa.

$ ***echo $PIPPO1***[*Invio*]

```
478
```

33.1.28 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 **=**. 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 nello *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.

33.1.29 logout

`logout`

Termina l'esecuzione di una *shell* di *login*.

33.1.30 popd

`popd [+/-`*n*`]`

Elimina degli elementi dallo *stack* delle directory. Se non vengono forniti argomenti, elimina la directory in cima allo *stack* (l'ultima a essere stata inserita), ed esegue un **cd** verso quella directory.

Opzioni

+*n*

Preleva solo l'*n*-esimo elemento dello *stack* a partire dall'alto. L'ultimo elemento inserito si raggiunge con **+0**.

-*n*

Preleva solo l'*n*-esimo elemento dello *stack* a partire dal basso. Il primo elemento inserito si raggiunge con **-0**.

Se il comando **popd** viene eseguito con successo, viene anche effettuato un **dirs**, e il valore restituito è zero. **popd** restituisce *Falso* se viene commesso un errore di qualunque tipo.

33.1.31 pushd

`pushd [⟨directory⟩] | pushd [+/-n]`

Aggiunge una directory in cima allo *stack* delle directory, oppure sposta gli elementi e alla fine esegue un **`cd`** verso quella directory che si trova in cima. Senza argomenti, scambia le due directory in cima e restituisce zero, a meno che lo *stack* sia vuoto.

Opzioni

`+n`

Ruota lo *stack* in modo che l'*n*-esima directory a partire dall'altro (cioè a partire dall'ultima) vada in cima.

`-n`

Ruota lo *stack* in modo che l'*n*-esima directory a partire dal basso (cioè a partire dalla prima) vada in cima.

⟨*directory*⟩

L'indicazione esplicita di una directory, aggiunge questa directory in cima allo *stack* rendendola anche la nuova directory corrente.

Se il comando ha successo, viene anche effettuato un **`dirs`**. Se viene usata la prima forma, **`pushd`** restituisce zero a meno che il **`cd`** fallisca. Con la seconda forma, **`pushd`** restituisce zero purché non sia stato specificato un elemento non esistente nello *stack*, o il cambiamento di directory fallisca.

33.1.32 pwd

`pwd`

Emette il percorso assoluto della directory corrente. Se è stato usato il comando interno **`set -P`**, i percorsi che utilizzano *link* simbolici vengono tradotti in percorsi reali. Restituisce zero se non si verifica alcun errore mentre si legge il percorso della directory corrente.

33.1.33 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 è stata diversamente distribuita. 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*.

Opzioni

`-r`

Utilizzando questa opzione, la barra inclinata rovescia (\) seguita dal codice *newline*, 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.

33.1.34 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.

33.1.35 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*.

33.1.36 set

`set [`⟨*flag*⟩`] [`⟨*argomento*⟩`... ]`

Questo comando viene usato per modificare una serie molto lunga di modalità di funzionamento. Segue l'elenco dei *flag*.

Flag

`-a`

Le variabili che vengono modificate o create, sono marcate automaticamente per l'esportazione verso l'ambiente per i comandi avviati dalla *shell*.

`-b`

Fa in modo che lo stato di un *job* in *background* terminato, venga riportato immediatamente. Altrimenti, questa informazione viene emessa subito prima del successivo *prompt* primario.

`-e`

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 una istruzione *if*, di una lista **`&&`** o ||, o se il valore restituito dal comando è stato invertito per mezzo di **`!`**.

`-f`

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 nuovo avvio successivo.

`-k`

Tutti gli argomenti in forma di istruzione di assegnamento sono collocati nell'ambiente per il comando, non soltanto quelli che precedono il nome del comando stesso.

`-m`

Il controllo dei *job* è attivato. Questo *flag* è attivo in modo predefinito per le *shell* interattive.

`-n`

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.

`-o` ⟨*nome-di-opzione*⟩

Permette di indicare una serie di opzioni in modo verbale (espressivo). Queste opzioni vengono descritte nella sezione seguente.

Se non viene fornito alcun nome di opzione, vengono emessi i valori delle opzioni correnti.

`-p`

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 lo UID e il GID efficace non è uguale all'UID e GID reale. Disattivare questa modalità fa sì che lo UID e GID efficaci diventino uguali a quelli reali.

`-t`

Termina l'esecuzione dopo aver letto ed eseguito un comando.

`-u`

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`

Emette le righe inserite nella *shell* appena queste vengono lette.

`-x`

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`

Viene attivata l'espansione delle parentesi graffe (predefinito).

`-C`

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 > (31.5).

> In generale sarebbe meglio evitare di intervenire in questo modo, dal momento che ciò non è conforme allo standard di utilizzo normale.

`-H`

Abilita la sostituzione dallo storico utilizzando lo stile **!**. Questo *flag* è attivo in modo predefinito quando la *shell* è interattiva.

`-P`

Se attivato, non segue i *link* simbolici quando esegue i comandi che, come **cd**, cambiano la directory corrente. Vengono usate invece le directory reali. L'azione di seguire i *link* simbolici non è così ovvia come sembra.

- Esempi

  Se `/usr/sys` è un *link* 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
  ```

`--`

Se nessun argomento segue questo *flag*, allora i parametri posizionali (**$1**, **$2**, ecc.) non sono impostati. Altrimenti, i parametri posizionali sono posti al valore degli argomenti, persino se alcuni di loro iniziano con un trattino.

`-`

Segnala la fine delle opzioni, e fa sì che tutti gli argomenti rimanenti siano assegnati ai parametri posizionali. Le opzioni **-x** e **-v** sono disattivate. Se non vi è alcun argomento, i parametri posizionali rimangono inalterati.

I *flag* sono disattivati in modo predefinito, salvo se specificato diversamente. Usando il segno **+** piuttosto di **-** fa in modo di disattivare (*off*) questi *flag*. I *flag* possono anche essere specificati come opzioni in una chiamata della *shell*. L'insieme corrente di *flag* può essere trovato nel parametro **$-**. Dopo le opzioni, gli argomenti rimanenti sono trattati come valori per i parametri posizionali e sono assegnati in ordine a **$1**, **$2**,... Se non viene fornita alcuna opzione o argomento, sono emesse tutte le variabili di *shell*.

set restituisce *Vero* se non viene incontrata una opzione errata.

33.1.36.1 Opzioni del flag -o

Il *flag* **`-o`** permette di definire le opzioni elencate di seguito.

`allexport`
Lo stesso del *flag* **`-a`**.
`braceexpand`
La *shell* effettua l'espansione delle parentesi graffe. Si tratta della modalità predefinita.
`emacs`
Usa l'*editing* della riga di comando in stile Emacs. Si tratta della modalità predefinita quando la *shell* è interattiva, a meno che non sia stata avviata con l'opzione **`-nolineediting`**.
`errexit`
Lo stesso di **`-e`**.
`histexpand`
Lo stesso di **`-H`**.
`ignoreeof`
L'effetto è lo stesso di quello ottenibile eseguendo **`IGNOREEOF=10`**.
`interactive-comments`
Permette di ignorare i commenti (**`#`** e il resto della riga) anche durante l'esecuzione di una *shell* interattiva
`monitor`
Lo stesso di **`-m`**.
`noclobber`
Lo stesso di **`-C`**.
`noexec`
Lo stesso di **`-n`**.
`noglob`
Lo stesso di **`-f`**.
`nohash`
Lo stesso di **`-d`**.
`notify`
Lo stesso di **`-b`**.
`nounset`
Lo stesso di **`-u`**.
`physical`
Lo stesso di **`-P`**.
`posix`
Cambia il comportamento di **`bash`** dove le operazioni predefinite differiscono dallo standard POSIX 1003.2 per farlo combaciare con lo standard.
`privileged`
Lo stesso di **`-p`**.
`verbose`
Lo stesso di **`-v`**.
`vi`
Usa un *editing* della riga di comando in stile **`vi`**.
`xtrace`
Lo stesso di **`-x`**.

33.1.37 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.

33.1.38 shopt

`shopt [`⟨*opzioni*⟩`] [`⟨*opzione-di-shell*⟩`... ]`
Questo comando permette di cambiare alcune opzioni di *shell*, attivandole o disattivandole. **`shopt`** si affianca al comando **`set`**, e permette anche di intervenire sulle opzioni gestite da quest'ultimo.

Opzioni

`-s`
Attiva le opzioni di *shell* elencate.

`-u`
Disattiva le opzioni di *shell* elencate.

`-o [`⟨*opzione-set*⟩`...]`
Permette di intervenire sulle opzioni gestite dal comando **`set`**.

Se si utilizza l'opzione **`-s`** o **`-u`** senza l'indicazione di nomi di opzioni di *shell*, vengono elencati rispettivamente quelle attive e quelle disattivate.
L'elenco di queste opzioni di *shell* non viene riportato. Vedere **`info bash`** o **`bash(1)`**.

33.1.39 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.

33.1.40 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. Se i file vengono espressi nella forma `/dev/fd/`*n*, allora si fa riferimento al descrittore di file *n*.

33.1.40.1 test sul tipo di file

`-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 *link* 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 [`⟨*file-descriptor*⟩`]`
Vero se il descrittore di file indicato è aperto su un terminale. Se non viene indicato l'argomento, si intende **`1`** che corrisponde allo *standard output*.

33.1.40.2 Test sui permessi di accesso

`-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 suo bit *sticky* impostato.
`-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

33.1.40.3 Test sulle caratteristiche dei file

`-s` ⟨*file*⟩
Vero se il file esiste e ha una dimensione maggiore di zero.
⟨*file1*⟩ `-nt` ⟨*file2*⟩
Vero se il primo file è più recente (in base alla data di modifica) del secondo.
⟨*file1*⟩ `-ot` ⟨*file2*⟩
Vero se il primo file è più vecchio (in base alla data di modifica) del secondo.
⟨*file1*⟩ `-et` ⟨*file2*⟩
Vero se i due file sono sullo stesso dispositivo e hanno lo stesso numero di *inode*. Ciò accade quando si creano degli *hard link*.

33.1.40.4 Test su stringhe

`-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 precede la seconda in un ipotetico ordinamento.
⟨*stringa1*⟩ > ⟨*stringa2*⟩
Vero se la seconda stringa precede la prima in un ipotetico ordinamento.

33.1.40.5 Test di confronto numerico

⟨*argomento1*⟩ ⟨*operando*⟩ ⟨*argomento2*⟩
Il risultato dipende dall'operando utilizzato.

Operandi

- **`-eq`** uguale a;
- **`-ne`** diverso da;
- **`-lt`** minore di;
- **`-le`** minore o uguale a;
- **`-gt`** maggiore di;
- **`-ge`** maggiore o uguale a.

Gli argomenti da confrontare possono essere interi positivi, interi negativi, o l'espressione speciale **`-l`** ⟨*stringa*⟩ che calcola la lunghezza della stringa stessa.

33.1.40.6 Operatori booleani

`!` ⟨*espressione*⟩
Vero se il risultato dell'espressione è *Falso*.
⟨*espressione1*⟩ `-a` ⟨*espressione2*⟩
Vero se entrambe le espressioni danno un risultato *Vero*.
⟨*espressione1*⟩ `-o` ⟨*espressione2*⟩
Vero se almeno una delle due espressioni dà un risultato *Vero*.

33.1.41 times

`times`
Emette i tempi di utilizzo accumulati.

33.1.42 trap

`trap [-l] [⟨argomento⟩] [⟨segnale⟩]`

Il comando espresso nell'argomento dovrà essere letto ed eseguito quando la *shell* riceverà il/i segnali indicati. Se non viene fornito l'argomento, o viene fornito 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. Il segnale indicato deve essere uno di quelli definiti in **`signal.h`**, o un numero di segnale. 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.

33.1.43 type

`type [-all] [-type | -path] ⟨nome⟩ [⟨nome⟩... ]`

Determina le caratteristiche di uno o più comandi indicati come argomento.

Opzioni

⟨*nessuna opzione*⟩

Se viene usato senza opzioni, indica come verrebbe interpretato ciascun nome indicato negli argomenti, se 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);
- **`file`**.

Se il nome non viene trovato, allora non viene emesso nulla e viene restituito il valore *Falso*.

`-path | -p`

Viene emesso o il nome del file che verrebbe eseguito se il nome indicato negli argomenti fosse utilizzato come un nome di comando, o nulla se non esiste un tale file. Se un comando si trova nella tabella *hash*, viene emesso il valore lì contenuto e non necessariamente il file che appare per primo nell'elenco dei percorsi di ricerca della variabile **`PATH`**.

`-all | -a`

Emette tutti i ritrovamenti di elementi con i nomi indicati. Inclusi alias e funzioni, se, e solo se non viene usato anche il *flag* **`-path`**. La tavola *hash* dei comandi non viene consultata quando si usa **`-all`**.

Restituisce *Vero* se uno qualsiasi degli argomenti viene trovato, *Falso* se non ne viene trovato alcuno.

33.1.44 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 il limite viene omesso, viene emesso il valore corrente del limite *soft* o *hard* (a seconda delle opzioni) della risorsa. Quando viene specificata più di una risorsa, il nome del limite e l'unità vengono emessi prima del valore.

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.

33.1.45 umask

`umask [-S] [`⟨*modalità*⟩`]`

La maschera di 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.

33.1.46 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.

33.1.47 unset

`unset [-fv] [`⟨*nome-di-variabile-o-di-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 essa viene successivamente ripristinata.

Restituisce *Vero* se non vengono riscontrati errori.

33.1.48 wait

`wait [n]`

Attende la conclusione del processo specificato e restituisce il suo stato 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.

Parte IX

Memoria di massa, dischi e filesystem

Appunti Linux

Via Turati, 15 - I-31100 Treviso - daniele@calion.com

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.

<*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/*>

La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.

<*http://www.pluto.linux.it/ildp/AppuntiLinux/*>

<*http://ildp.psy.unipd.it/AppuntiLinux/*>

<*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

<*http://linux.calion.com/pluto/ildp/AppuntiLinux/*>

Memoria di massa

Con il termine ***memoria di massa*** ci si riferisce alla parte di memoria non volatile di un computer, che consente l'immagazzinamento di grandi quantità di dati. Il tipo di supporto (volume, media) più utilizzato è a disco, che permette un accesso rapido ai dati memorizzati, anche se ci sono ancora ancora i nastri magnetici, graditi per la loro economicità.

34.1 Nastro

Il sistema di memorizzazione a nastro è stato il primo (a parte le schede perforate) a essere utilizzato con i computer. 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 una 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.

34.2 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, magnetoottica 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 magnetoottici 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.

34.2.1 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.

34.2.2 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), che 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 vendono 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.
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

[1] Bisogna ricordare che siamo immersi in una grande quantità di fonti magnetiche, a cominciare dal campo terrestre, a cui si aggiungono tutti quelli generati dall'attività umana: un telefono cellulare acceso, appoggiato a un dischetto magnetico provoca la sua cancellazione.

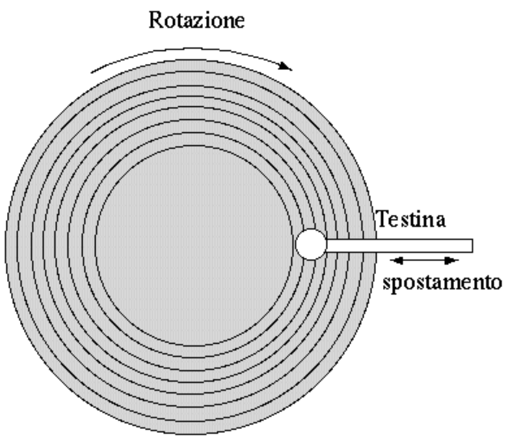

Figura 34.1: Visione dall'alto di un piatto sul quale scorre una testina per la lettura e scrittura dei dati. Le tracce magnetiche, concentriche, non sono visibili, ma vengono individuate dalla testina magnetica.

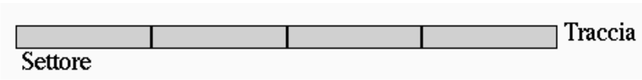

Figura 34.2: Le tracce concentriche dei dischi magnetici sono suddivise in settori di uguale dimensione.

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 una unità di memorizzazione è composta da più dischi, questi sono posti assieme sullo stesso asse e 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.

34.2.3 Geometria

Come già accennato, il settore è la minima unità di memorizzazione accessibile in un disco, di qualunque tipo sia. Nel caso di dischi organizzati a tracce concentriche, si utilizzano coordinate composte da cilindro, testina, settore.

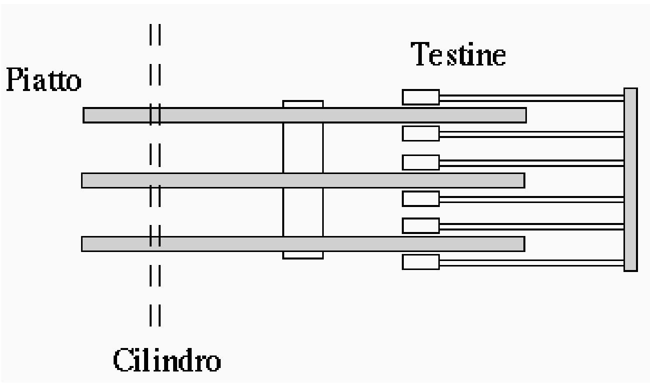

Figura 34.3: Schema di un disco fisso visto lateralmente.

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.
In questo modo, 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.

34.2.4 Dischi magnetoottici

I dischi magnetoottici (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 magnetoottico 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.

34.2.5 Dischi ottici

Mentre i dischi magnetoottici si comportano in maniera analoga a quelli magnetici, i dischi ottici richiedono una registrazione sequenziale, ed eventualmente consentono una 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.

34.3 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 una 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]

34.4 Partizioni

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 più o meno seguita. I sistemi operativi principali si conformano a questa convenzione, altri la interpretano a loro modo.[3]

34.4.1 MBR

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.

Lo spazio riservato nell'MBR per annotare i dati delle partizioni è limitato e consente la suddivisione in un massimo di 4 partizioni principali.

34.4.2 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.

[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.

[3] Per esempio, Minix utilizza una partizione che poi suddivide per conto suo in una maniera particolare non riconoscibile da un altro sistema operativo.

La partizione primaria è quella normale e a essa si attribuisce un codice per riconoscere il tipo di *filesystem* che contiene o dovrebbe contenere. La partizione estesa viene definita con il codice **5** (in esadecimale) ed è il contenitore di altre partizioni più piccole, dette partizioni logiche.

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.

34.5 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.

34.6 Memoria transitoria (cache)

L'accesso ai dati dei dischi è una 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 tampone, 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 del computer, può comportare una perdita parziale dei dati, se le operazioni di scrittura accodate nella *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 il computer dopo aver completato la procedura di spegnimento in base al tipo di sistema operativo utilizzato.

34.7 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 *flag* particolare quando il suo *filesystem* viene montato. Nel momento in cui viene smontato, l'ultima cosa a essere modificata nel *filesystem* è il *flag* 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.

34.7.1 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 *link*.
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 *link* (indicati più precisamente come *link hard*) 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 (*link*).

Directory

La directory contiene una serie di voci, e per ognuna di queste:

- il nome;
- il riferimento all'*inode* (cioè il *link*).

34.7.2 Link

Come si è visto nella sezione precedente, le voci di una directory contengono ognuna un riferimento (detto comunemente *link*) 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 *link* simbolici, che sono solo file contenenti un riferimento a un altro file.
Per distinguere questi due tipi di *link*, quando si parla di *link* normali, si usa l'aggettivo *hard*, o *hardware*.

34.8 Riferimenti

- Stein Gjoen, *Disk HOWTO*
- Lars Wirzenius, *Linux System Administrators' Guide*

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 (*root*) e gli altri secondari, collegati fra loro in modo da formare una unica struttura.
La tabella 35.1 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* Ext2.
mkdosfs | mkfs.msdos	Creazione di un *filesystem* Dos-FAT.
mkfs	*Front-end* per l'utilizzo dei programmi di creazione dei *filesystem*.
e2fsck | fsck.ext2	Controllo di un *filesystem* Ext2.
dosfsck	Controllo di un *filesystem* Dos-FAT.
fsck	*Front-end* per l'utilizzo dei programmi di controllo dei *filesystem*.
mount	Collegamento di un *filesystem* in quello globale.
umount	Distacco di un *filesystem* da quello globale.
/etc/fstab	Elenco di *filesystem* e *mount point* predefiniti.
/etc/mtab	Elenco dei *mount* 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	Disattiva il controllo delle quote.
edquota	Modifica le quote assegnate.
repquota	Restituisce una tabella delle quote assegnate e della situazione effettiva.
quota	Verifica della quota personale.
update (bdflush)	*Daemon* per lo scarico periodico della memoria *cache* dei dischi.
sync	Scarico manuale della memoria *cache* dei dischi.
mkisofs	Creazione di un *filesystem* ISO-9660.
cdrecord	Masterizzazione di CD-R.

Tabella 35.1: Riepilogo dei programmi e dei file per la gestione dei dischi e dei *filesystem*.

35.1 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. la eventuale suddivisione in partizioni e
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, in particolare, delle unità rimovibili come i dischetti. In quest'ultimo caso, a parte i dischetti, si deve quasi sempre utilizzare quanto fornito insieme ai *drive*, 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`**, 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.

35.1.1 # fdformat

`fdformat [-n]` ⟨*dispositivo*⟩

`fdformat` esegue una 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/fd0H1440` e `/dev/fd1H1440` che si riferiscono al formato da 1440KB 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 normali 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/fd0H1440`***

Inizializza un dischetto da 1440KB nella prima unità a dischetti.

`#` ***`fdformat /dev/fd1H1440`***

Inizializza un dischetto da 1440KB nella seconda unità a dischetti.

35.1.2 # 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 di rado.

> 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/fd0H1440 1440`***

Esegue il controllo del dischetto, in sola lettura, per tutta la sua estensione: 1440 blocchi di 1Kbyte. Trattandosi di un controllo in sola lettura, **`badblocks`** può essere eseguito da un utente normale (sempre che questo abbia i permessi in lettura per il dispositivo che si va a leggere).

35.1.3 # 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
   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
```

[1] Vale la pena di ricordare che i nomi di dispositivo relativi ai dischetti possono cambiare leggermente da una distribuzione Linux a un'altra. Spesso, il formato dei dischetti da 1440KB corrisponde al file `/dev/fd0u1440`.

```
   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 automaticamente definita del tipo **`Linux native`**, ma in certi casi può essere necessario modificare il tipo di partizione creato attraverso il comando **`t`**. Ogni tipo di partizione ha un codice 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, è necessario riavviare il sistema prima di poter 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`, ...) 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.

35.1.4 # 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`**.

35.1.5 # 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 una unità di memorizzazione. Questa viene indicata nel modo consueto, attraverso l'indicazione del nome del file di dispositivo corrispondente (`/dev/...`).

La dimensione è espressa in blocchi. Se viene omessa l'indicazione di questo valore, **`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 4K byte (4096) e non può essere inferiore a 1K (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 *backup* 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`** (35.2.1), ma non c'è alcuna garanzia che il recupero funzioni.

35.1.6 # 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 nel primo *drive*, dopo aver controllato la sua superficie e determinando automaticamente la dimensione in blocchi.

35.1.7 # 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 parla anche di *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 nel primo *drive*, 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 nel primo *drive*, dopo aver controllato la sua superficie. La dimensione in blocchi viene indicata in modo esplicito.

35.2 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.

35.2.1 # 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 venir poste.

`-p`

Ripara automaticamente il *filesystem* senza porre alcuna domanda.

`-y`

Risponde automaticamente **`y`**, *yes*, a tutte le domande che potrebbero venir poste, in modo da permetterne l'utilizzo non interattivo attraverso uno *script*.

Exit status

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.

35.2.2 # 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.

Per conoscere maggiori dettagli conviene consultare **`dosfsck(8)`**.

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.

Exit status

- **`0`** Non sono stati riscontrati errori nel *filesystem*.
- **`1`** Sono stati riscontrati errori risolvibili.
- **`2`** Errore nell'utilizzo o nella sintassi.

35.2.3 # 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 *front-end*.

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)`**.

35.3 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 computer, occorre eseguire l'operazione opposta di distacco. Occorre cioè smontarla (*unmount*).
In un sistema Linux devono essere necessariamente collegati il *filesystem root*, ovvero quello principale, e il *filesystem* virtuale `/proc/` che però non fa capo ad alcuna unità fisica.
Se si utilizzano partizioni di *swap* per la gestione della memoria virtuale, queste devono essere collegate con una operazione concettualmente simile al *mount*, anche se poi non appaiono nella struttura generale del *filesystem* globale.

35.3.1 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 35.2, sono elencati i nomi che identificano i tipi di *filesystem* riconoscibili da Linux.

tipo di filesystem	descrizione
minix	Minix
ext2	Second extended
hpfs	
umsdos	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 swap

Tabella 35.2: Elenco dei nomi di filesystem utilizzabili.

35.3.2 Implicazioni legate al mount

Il *mount* 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 *mount*.

- Il *filesystem* estraneo potrebbe non essere sicuro, di conseguenza si pone il problema di:
  - decidere se permettere l'esecuzione 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 normali possono eseguirne il *mount*.
- 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.
- 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 da parte di un utente e gruppo particolare.

35.3.3 Opzioni

In occasione del *mount* 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.
Vedere **`mount(8)`** e **`nfs(5)`**.

Opzioni valide per ogni tipo di filesystem

`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 *mount* 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 attivare 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 dei *filesystem* **`umsdos`**, **`msdos`** e **`vfat`**.

`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 *mount*.

`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 *mount*.

`umask=`⟨*maschera*⟩

Permette di stabilire quali permessi inibire nel *filesystem*. Si tratta di tre cifre numeriche dove la prima 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.

`case=lower | case=asis`

Converte in minuscolo, o lascia come sono, i caratteri dei nomi dei file. Il valore predefinito è **`case=lower`**.

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.

35.3.4 # 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* precedentemente montato. La forma normale e più semplice di utilizzo di **mount** è la seguente.

mount -t ⟨*tipo-di-filesystem*⟩ ⟨*dispositivo*⟩ ⟨*mount-point*⟩

In questo modo si richiede al *kernel* di montare il *filesystem* del dispositivo specificato nella directory indicata (*mount point*).

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 *mount*, 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 avvio del sistema (/etc/rc.d/rc*). La presenza del file di configurazione /etc/fstab è utile anche per semplificare il *mount* (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 ⟨*mount-point*⟩

In linea di principio, solo l'utente *root* può montare un *filesystem*. Per permettere agli utenti normali di montare e smontare una 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 *mount* 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* ad esclusione di quelli segnati come **noauto**.

-t [no]⟨*tipo-di-filesystem*⟩[**,...**]

Specifica il tipo di *filesystem*. Sono riconosciuti i nomi indicati nella tabella 35.2. 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 e/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 *mount* 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 35.3.3.

Esempi

`# mount -t ext2 /dev/hda2 /mnt`

Monta il *filesystem* di tipo Ext2 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 condiviso dal computer **`roggen.brot.dg`**, corrispondente alla sua directory `/pubblica/` (e discendenti), nella directory locale `/roggen/`.

35.3.5 # 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* precedentemente montato, riferito al dispositivo `/dev/hda2`.

`# umount /mnt`

Smonta il *filesystem* precedentemente montato nella directory `/mnt`.

35.3.6 /etc/fstab

Il file `/etc/fstab` viene utilizzato per definire le caratteristiche e le directory di collegamento (*mount point*) 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 **`#`** sono ignorate e trattate 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 *mount point* per il *filesystem*.
3. Il terzo campo definisce il tipo di *filesystem* e ne viene indicato il nome in base alla tabella 35.2.

   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.
4. Il quarto campo descrive le opzioni speciali per il tipo di *mount* che si intende eseguire. Si tratta delle stesse opzioni speciali descritte in **`mount(8)`** e anche nella sezione 35.3.4 in occasione della spiegazione dell'uso dell'opzione **`-o`**.
5. Il quinto campo viene utilizzato per determinare quali *filesystem* possono essere utilizzati per il *dump*.
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 root*, cioè quello principale, dovrebbe avere il numero **`1`** in questo campo, mentre gli altri, il numero **`2`**. Se questo campo contiene il valore **`0`**, significa che quel *filesystem* non deve essere controllato.

Esempi

Nell'esempio seguente, tutte le unità che non sono stabilmente collegate al corpo fisico del computer, hanno l'opzione **`noauto`** che impedisce il *mount* automatico all'avviamento del sistema. Queste vengono attivate quando si esegue **`mount`** con il vantaggio di poter semplicemente indicare la directory di collegamento (*mount point*) o il nome del dispositivo.

```
# nome              Mount point  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 root* 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 correttamente inserito 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. Viene in particolare 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 *mount* 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 normali 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.
- Il computer **`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.

35.3.7 /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.

35.3.8 $ 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.

35.4 Quota

Generalmente, l'utilizzo dello spazio nel *filesystem* non è controllato, per cui gli utenti possono teoricamente utilizzare 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 le 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, Linux può gestire esclusivamente le quote di utilizzo delle partizioni di tipo Ext2, cioè il suo tipo nativo.

35.4.1 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.

35.4.2 Kernel

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* (15.5.11) **Y**

35.4.3 Configurazione con /etc/fstab

La gestione delle quote delle partizioni deve essere attivata espressamente nel momento del *mount*. 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               Mount point  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
```

35.4.4 Registrazione delle quote

I livelli di quota dei vari utenti e/o 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 inizialmente creare tali file, vuoti.

L'esempio seguente mostra il caso in cui si voglia controllare la partizione di *root*, ovvero quella che contiene l'origine del *filesystem*.

```
# touch /quota.user
# touch /quota.group
# chmod 0600 /quota.user
# chmod 0600 /quota.group
```

35.4.5 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.

Detto questo, 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 Linux non fornisce degli *script* già pronti, si può 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
```

Se tra le partizioni da controllare c'è anche quella di *root*, l'unico modo per fare sì che il controllo delle quote sia operativo, è quello di riavviare il sistema.

35.4.6 # quotacheck

`quotacheck [`⟨*opzioni*⟩`] [{`⟨*partizione*⟩|⟨*mount-point*⟩`}...]`

`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 *root*.

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 *mount* **`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 `/mnt/disco2` come soggette al controllo delle quote, sia degli utenti che dei gruppi.

35.4.7 # quotaon

`quotaon [`⟨*opzioni*⟩`] [{`⟨*partizione*⟩|⟨*mount-point*⟩`}...]`

`quotaon` attiva la gestione delle quote da parte del *kernel*. Non si tratta quindi di un *daemon*, 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 *mount* **`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 `/mnt/disco2` come soggette al controllo delle quote, sia degli utenti che dei gruppi.

35.4.8 # quotaoff

`quotaoff [`⟨*opzioni*⟩`] [{`⟨*partizione*⟩|⟨*mount-point*⟩`}...]`

`quotaoff` disattiva la gestione delle quote da parte del *kernel*. Le opzioni e la sintassi sono le stesse di **`quotaon`**.

35.4.9 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 limiti di utilizzo di *inode*, in pratica, il numero massimo di file. Dal momento che si ha a che fare con *filesystem* Ext2 che normalmente possono avere un *inode* ogni 4KB, si può facilmente stabilire 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 implicitamente consentire 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 **`quota`**.

Con il programma **`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 10Mbyte, con una tolleranza del 10% (11Mbyte 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
```

L'esempio mostra 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.

35.4.10 # edquota

edquota [⟨*opzioni*⟩] [⟨*utente*⟩...]

edquota è il programma per 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**.

35.4.11 # repquota

`repquota [`⟨*opzioni*⟩`] [`{⟨*partizione*⟩|⟨*mount-point*⟩}`...]`

`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).

35.4.12 $ 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`**.

35.5 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 tampone, è il motivo principale per cui si deve fare attenzione a chiudere l'attività del sistema prima di spegnere il computer.

La memoria *cache* viene gestita automaticamente dal *kernel*, ma è un *daemon* quello che si occupa di richiedere lo scarico periodico.

35.5.1 # update (bdflush)

`update [⟨opzioni⟩]`

Con questo nome, **`update`**, viene avviato il *daemon* che si occupa di richiedere periodicamente al *kernel* lo scarico della memoria *cache*. Deve essere avviato, 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 oppure l'inserimento di un *record* apposito all'interno di `/etc/inittab`.

Esempi

L'esempio seguente mostra una configurazione della procedura di inizializzazione attraverso la quale **`update`** viene avviato attraverso lo *script* iniziale. Segue il *record* del file `/etc/inittab` con cui si fa riferimento allo *script* e lo spezzone di *script*.

```
si::sysinit:/etc/rd.d/rc.sysinit
...
```

```
# Attivazione della memoria virtuale.
/sbin/swapon -a

# Attivazione di update.
/sbin/update

# Controllo della partizione del filesystem root.
/sbin/fsck -A -a
...
```

L'esempio seguente mostra una configurazione della procedura di inizializzazione attraverso la quale **`update`** viene avviato direttamente attraverso `/etc/inittab`.

```
ud::once:/sbin/update
...
```

35.5.2 $ sync

`sync [⟨opzioni⟩]`

`sync` permette di scaricare nei dischi i dati contenuti nella memoria *cache*. Viene usato solitamente dalla procedura di spegnimento per garantire che tutti i dati siano correttamente registrati su disco prima dello spegnimento fisico del computer.

Può essere utilizzato in caso di emergenza, quando per qualche ragione non si può attendere il completamento della procedura di spegnimento, o per qualunque altro motivo.

Di solito non si usano opzioni.

35.6 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 utilizzati per la creazione dei dischetti di avvio utilizzati per installare Linux la prima volta.

35.6.1 Gestione da parte del kernel

Per utilizzare 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* (15.5.5).

35.6.2 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 1440KB.

`# 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** chiede conferma. Basta inserire la lettera **y** per proseguire.

y[*Invio*]

Nello stesso modo si poteva creare un *filesystem* differente.

35.6.3 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 successivamente inizializzato, 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.

35.6.4 trasferimento di un file-immagine in un disco

Se il file-immagine su cui si ha lavorato corrisponde esattamente a un disco o 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.

35.7 Filesystem ISO-9660 e creazione di CD-ROM

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 spirare che parte dalla parte 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.

35.7.1 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 una estensione di un massimo di tre caratteri.[2] 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.

[2] 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.

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.

35.7.2 $ mkisofs

`mkisofs [`⟨*opzioni*⟩`]` ⟨*directory*⟩

`mkisofs` è un programma in grado di generare un *filesystem* ISO-9660, eventualmente utilizzando le estensioni Rock Ridge, 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.

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 una 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 e/o cancelletto (**`~`** e **`#`**) non vengono inclusi, trattandosi normalmente di copie di *backup*.

`-f`

Utilizzando questa opzione, si fa in modo che i *link* 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 *link* 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 una 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*, vengono 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 riconosca le estensioni Rock Ridge.

`-v`

Durante il funzionamento, genera più informazioni.

`-x` ⟨*directory*⟩

Permette di escludere la directory indicata.

Esempi

`#` ***mkisofs -r -T -v -o prova.img /home/tizio/elio***

Crea una 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 **`#`**.

35.7.3 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 *home*.

`$` ***mkisofs -r -T -v -o ~/prova.img ~/prova***[*Invio*]

In questo modo è stato creato il file `prova.img` nella directory *home* dell'utente.

Per verificare il risultato si può eseguire il *mount* dell'immagine appena creata, ma per questo occorre avere i privilegi dell'utente *root*.

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

`Password:` ***ameba***[*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, una 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 una 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*]

35.7.4 Masterizzazione

Per giungere alla realizzazione di un proprio CD, occorre una unità per la masterizzazione e il software adatto per trasferire una 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.

35.7.4.1 Masterizzatore SCSI

Il masterizzatore più comune è di tipo SCSI e per questo è necessario che il *kernel* sia stato predisposto opportunamente.

- *SCSI support* (15.5.7) **Y**
- *SCSI disk support* (15.5.7) **Y**
- *SCSI CD ROM support* (15.5.7) **Y**
- *SCSI generic support* (15.5.7) **Y**

Oltre a questo occorre avere indicato il tipo di controller SCSI.

La registrazione di un CD-R attraverso una 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 controller SCSI si tratta (probabilmente è l'unico installato), il numero dell'unità SCSI e l'eventuale LUN (se il LUN non viene utilizzato, questo corrisponderà semplicemente a zero).

35.7.4.2 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 il computer 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 una 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.

35.7.5 # 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=`⟨*controller-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 (**-**).[3]

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 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 KB/s. Per fare un esempio, 4 corrisponde a quello che di solito si esprime come x4 o 4x.

---

[3] 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.

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.

`-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 una 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 -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.

35.7.6 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 può 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*.

Concludendo, 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.

Come ultima considerazione, 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`

35.8 Dischi senza filesystem

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 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`**.

35.9 Riferimenti

- Stein Gjoen, *Disk HOWTO*
- Jeff Tranter, *CDROM HOWTO*
- Winfried Trümper, *CD-writing HOWTO*

Memoria virtuale

La memoria virtuale è una estensione virtuale 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 *swap* deriva da questa continua operazione di scambio.

Utilizzando Linux, se non si dispone di una quantità di memoria RAM molto grande (anche molto superiore a 16MB) è praticamente necessario attivare il meccanismo della memoria virtuale

La tabella 36.1 elenca i programmi e i file a cui si accenna in questo capitolo.

Nome	Descrizione
mkswap	Inizializza un'area di *swap*.
swapon	Attiva un'area di *swap*.
swapoff	Disattiva un'area di *swap*.
/etc/fstab	Elenco di *filesystem* e di aree di *swap* gestiti automaticamente.

Tabella 36.1: Riepilogo dei programmi e dei file per la gestione della memoria virtuale.

36.1 Partizione o file di swap

Con Linux è possibile attivare la gestione della memoria virtuale utilizzando due possibili aree nel disco fisso: una partizione dedicata o un file. Dal momento che possono essere gestite diverse aree di *swap* conviene attivarne almeno una in una partizione dedicata. In pratica, la partizione di *swap* dovrebbe consentire almeno la gestione normale del sistema, mentre i file di *swap* potrebbero servire come un mezzo eccezionale per estenderne la dimensione.

In questo senso, la scelta della dimensione della partizione di *swap* è importante perché una volta deciso, questa normalmente non può più essere cambiata facilmente. La dimensione massima di un'area di *swap* è di 128MB e possono esserne definite un massimo di 16. In generale, per la partizione di *swap* è conveniente utilizzare una dimensione pari ad almeno la stessa quantità della memoria RAM effettiva, con un minimo di circa 10MB.

36.2 Creazione

Prima di poter attivare la gestione dello *swap* è necessario creare lo spazio in cui questo potrà risiedere. Questo vale anche nel caso in cui si utilizzi lo *swap* su un file.

36.2.1 Partizione di swap

La creazione di una partizione di *swap* procede nello stesso modo con cui si crea una qualunque altra partizione. In questo caso non c'è la necessità di eseguire un avviamento 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`** (35.1.3); occorre ricordare in particolare di assegnare alla partizione il tipo corretto di identificatore: **`82`** (*Linux swap*).

36.2.2 File di swap

La caratteristica necessaria di un file destinato a fungere da area di *swap* 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`** (40.1.7) 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 4KB, pari a quanto stabilito con l'opzione **`bs=4k`**. In effetti, la dimensione ottimale di un file del genere è un multiplo di 4KB perché le pagine di memoria, utilizzate durante lo scambio della stessa, sono di questa dimensione.

Per esempio, volendo creare il file di *swap* `/swap1` di 8MB si può procedere come segue.

`#` ***`dd if=/dev/zero of=/swap1 bs=4k count=2096`***

Se tutto procede come desiderato si ottiene una risposta del tipo seguente.

```
2096+0 records in
2096+0 records out
```

36.3 Inizializzazione

Un'area di *swap* 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 *swap* mentre questa è in uso. Ciò dovrebbe essere intuitivo, ma alle volte si dimentica di fare attenzione a questo particolare.

36.3.1 # mkswap

`mkswap [-c]` ⟨*dispositivo*⟩ `[`⟨*dimensione-in-blocchi*⟩`]`

mkswap permette di predisporre una partizione o un file per lo *swap*, 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 della 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 *swap* `/dev/hda3` specificando una dimensione di 33264 blocchi, pari a circa 32MB.

`#` ***mkswap -c /swap1 8192***

Inizializza il file di *swap* `/swap1` precedentemente creato con una dimensione di 8MB.

36.4 Attivazione e disattivazione della memoria virtuale

Per fare in modo che un'area di *swap* 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 *mount* dei dischi o delle partizioni. Per questo motivo, l'attivazione delle aree di *swap* può essere gestita automaticamente attraverso la configurazione del file `/etc/fstab`.

36.4.1 # swapon

`swapon [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩`]`

swapon attiva l'utilizzo di un dispositivo per la gestione della memoria virtuale (*swap*). Di solito si tratta di una partizione o un file di *swap* 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 *swap* previste all'interno del file `/etc/fstab`.

> Le partizioni o i file di *swap* attivati manualmente e non quindi con l'ausilio della configurazione del file `/etc/fstab` devono essere disattivati manualmente prima della conclusione dell'attività di Linux.

Alcune opzioni

`-a`

Viene attivato lo *swap* 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***

Avvia la gestione dello *swap* con la partizione `/dev/hda3`.

`#` ***swapon /swap1***

Avvia la gestione dello *swap* con il file `/swap1`.

`#` ***swapon -a***

Avvia la gestione dello *swap* con tutte le partizioni indicate così nel file `/etc/fstab`.

36.4.2 # swapoff

`swapoff [`⟨*opzioni*⟩`] [`⟨*dispositivo*⟩`]`

swapoff è l'opposto di **swapon** e si occupa di disattivare lo *swap* su un particolare dispositivo o file per lo *swap*, 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 *link* al programma **swapon** che si comporta così quando viene avviato con questo nome: **swapoff**.

Le partizioni o i file di *swap* attivati manualmente e non quindi con l'ausilio della configurazione del file `/etc/fstab` devono essere disattivati manualmente prima della conclusione dell'attività di Linux.

Esempi

`#` ***`swapoff /dev/hda3`***

Termina la gestione dello *swap* con la partizione `/dev/hda3`.

`#` ***`swapoff /swap1`***

Termina la gestione dello *swap* con il file `/swap1`.

`#` ***`swapoff -a`***

Termina la gestione dello *swap* con tutte le partizioni indicate così nel file `/etc/fstab`.

36.4.3 Gestione automatica attraverso /etc/fstab

Per quanto riguarda la gestione dello *swap*, il file `/etc/fstab` permette di definire quali partizioni e file debbano essere utilizzati automaticamente per questo scopo.
La configurazione di `fstab` per lo *swap* è praticamente obbligatoria, a meno di non voler ogni volta provvedere ad attivare e disattivare la memoria virtuale attraverso l'uso diretto di **`swapon`** e **`swapoff`**.
L'esempio seguente mostra due ipotetici *record* di `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
```

36.4.4 Procedura di inizializzazione e di spegnimento

Durante l'esecuzione della procedura di inizializzazione del sistema, si distinguono due fasi per l'attivazione della memoria di *swap*: prima dell'attivazione dei *filesystem* vengono attivate le partizioni di *swap*; dopo anche i file di *swap*.
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 *swap*, perché l'unico *filesystem* in funzione dovrebbe essere quello di *root* 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 il file di *swap* 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 *swap*, la chiusura del sistema deve avvenire nel modo corretto: prima si devono disattivare i file di *swap*, quindi si possono smontare i *filesystem* (riportando quello di *root* in sola lettura), e infine si possono disattivare le partizioni di *swap*.
In pratica, spesso si disattiva tutto lo *swap* subito, 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 *swap*.

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 *link* 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 *link* 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.

37.1 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 necessariamente essere accessibile anche in scrittura. Ciò che è condivisibile può essere utilizzato da più computer contemporaneamente, il resto no. Ciò che è indispensabile per l'avviamento del computer, non può, o non dovrebbe, essere collocato in *filesystem* remoti.
Purtroppo non è sempre detto che la distinzione sia sempre netta.

37.2 Filesystem standard

Nelle sezioni seguenti viene descritta la struttura essenziale di un *filesystem* Linux, secondo la struttura stabilita dal FHS (*Filesystem Hierarchy Standard*). Per maggiori dettagli si può consultare il documento originale all'indirizzo <*http://www.pathname.com/fhs*>.

37.2.1 / - 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/` *mount point* 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.

37.2.2 /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 normali (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.

37.2.3 /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.

Nei sistemi 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.

37.2.4 /dev/ - file di dispositivo

La directory `/dev/` contiene una lunga serie di file di dispositivo. Perché i vari componenti fisici del computer possano funzionare, occorre che per ognuno di essi sia stato previsto il relativo file di dispositivo, 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 *link* simbolico.

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.

37.2.5 /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. In particolare:

- `X11/` contiene la configurazione per X Window System;
- `opt/` contiene la configurazione specifica di programmi inseriti all'interno della gerarchia `/opt/`;
- `skel/` contiene le configurazioni di partenza per i nuovi utenti;
- `rc.d/` questa directory, o altra simile, contiene i file utilizzati dalla procedura di inizializzazione del sistema (stile System V).

37.2.5.1 /etc/X11/ - configurazione di X Window System

La directory `/etc/X11/` è il contenitore dei file di configurazione, riferiti a un particolare sistema, di X Window System. Qui dovrebbero trovarsi anche i file di configurazione dei gestori di finestre.

37.2.5.2 /etc/opt/ - configurazione dei programmi collocati in /opt/

I file di configurazione specifici di un sistema, riferiti a programmi collocati nella gerarchia `/opt/`, si trovano nella directory `/etc/opt/`. Più precisamente, da questa directory si articolano altre directory con lo stesso nome dell'applicativo a cui si riferiscono: `/etc/opt/`⟨*applicativo*⟩`/`.

37.2.6 /home/ - directory home degli utenti

La directory `/home/` è normalmente il punto di partenza per tutte le directory *home* 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.

37.2.7 /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, che contiene i moduli del *kernel*:

- `modules/`.

37.2.8 /mnt/ - mount point 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 relativi dischi e accedevi.

37.2.9 /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*⟩`/`.

37.2.10 /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.

37.2.11 /root/ - directory home dell'utente root

La directory `/root/` è la directory *home* dell'utente *root*. Ci sono molti validi motivi per evitare di mescolarla insieme a quelle degli utenti normali. 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 root*, 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.

37.2.12 /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 (*RAM disk*).

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 un'istruzione di eliminazione di tutti i file contenuti in questa directory, in modo che a ogni avviamento 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`**.

37.2.13 /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. X Window System che utilizza una propria directory discendente da `/usr/` fa eccezione.

37.2.13.1 /usr/X11R6/ - X Window System, 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 X Window System.

Il nome dipende dalla versione e dal rilascio, e per questo sono normalmente necessari (e utili) alcuni *link* 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 Window System, legati al sistema, devono essere collocati in `/etc/X11/`.

37.2.13.2 /usr/bin/ e /usr/sbin/ - binari non essenziali

La directory `/usr/bin/` contiene gli eseguibili di uso comune meno importanti. I file al suo interno sono generalmente accessibili in esecuzione a tutti gli utenti.
La directory `/usr/sbin/` contiene eseguibili non indispensabili, il cui utilizzo è generalmente 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 gestire funzionalità di rete essenziali per poter montare eventuali *filesystem* remoti. Tutto il resto, compresi i *daemon* 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 *link* 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
...
```

37.2.13.3 /usr/games/ - giochi e programmi educativi

La directory `/usr/games/` serve per contenere programmi meno importanti destinati al passatempo o all'educazione.
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/`.

37.2.13.4 /usr/include/ - file include

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 da qui, almeno attraverso *link* simbolici.

37.2.13.5 /usr/lib/ - librerie per la programmazione e 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/`.

37.2.13.6 /usr/local/ - programmi locali

La directory `/usr/local/` è il punto di inizio per l'installazione di programmi localmente, senza che questi siano interessati da normali procedure di aggiornamento del software 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 computer, mentre `/usr/local/` potrebbe essere la particolarità di ogni computer, o di un gruppo più piccolo.
In generale, questa directory dovrebbe apparire vuota subito dopo l'installazione di 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.

37.2.13.7 /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 *database* **`terminfo`**;
- `zoneinfo/` informazioni sull'ora legale.

Oltre a queste directory, potrebbero esserne aggiunte altre, specifiche di particolari applicazioni o gruppi di queste.

37.2.13.7.1 /usr/share/man/ - pagine di manuale

La directory `/usr/share/man/` contiene file 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 13.2.1.
Questa non è l'unica posizione in cui si collocano i file di manuale, ma questi, in particolare, riguardano il sistema in generale e i programmi collocati a partire dalla radice e da `/usr/`. Sono esclusi quelli che riguardano X Window System, collocati in `/usr/X11R6/man/`, quelli relativi ai programmi installati localmente che si trovano in `/usr/local/man/` e quelli specifici degli applicativi aggiunti che dovrebbero trovarsi in `/opt/`⟨*applicativo*⟩`/man/`.

37.2.13.7.2 /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`.

37.2.13.8 /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 *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 *link* 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/`.

37.2.14 /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/`.

37.2.14.1 /var/cache/ - directory di cache

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 archiviato in una procedura di *backup*.
La struttura essenziale di questa directory è la seguente:

- `fonts/` *font* generati localmente;

- `man/` pagine di manuale formattate;
- `www/` *proxy* WWW o dati transitori;
- ⟨*applicativo*⟩`/` dati transitori specifici di un determinato programma.

37.2.14.2 /var/lock/ - file di lock

I file *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.

37.2.14.3 /var/log/ - file di log

La directory `/var/log/` contiene i file di *log*: sia quelli utilizzati dal *log* di sistema, sia quelli di altri programmi.

37.2.14.4 /var/mail/ - caselle postali degli utenti

La directory `/var/mail/` viene usata per contenere i file delle caselle postali degli utenti.

37.2.14.5 /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*⟩`/`

37.2.14.6 /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 attualmente connessi 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.

37.2.14.7 /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, della posta inviata e di altri gestori di servizi.
In particolare vanno ricordate le sottodirectory seguenti:

- `lpd/` code di stampa;
- `mqueue/` posta in uscita;
- `rwho/` file di **`rwhod`**.

37.2.14.8 /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.

37.2.14.9 /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 dovrebbero essere cancellato al riavvio del sistema.

Parte X

File e directory

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.

<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>

<ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/>

<ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>

<ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/>

La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.

<http://www.pluto.linux.it/ildp/AppuntiLinux/>

<http://ildp.psy.unipd.it/AppuntiLinux/>

<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>

<http://linux.calion.com/pluto/ildp/AppuntiLinux/>

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 38.1 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 *pathname*.
dirname	Emette il nome della directory estraendolo da un *pathname*.
namei	Scompone un percorso alla ricerca di *link* 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 38.1: Riepilogo dei programmi per la gestione delle directory, dei percorsi e del loro contenuto.

38.1 Directory

La directory è un tipo speciale di file, il cui scopo è quello di contenere riferimenti ad altri file e/o directory. Detto in altri termini, la directory è un indice di file e/o di altri sottoindici.
I permessi attribuiti a una directory vanno interpretati in maniera particolare:

- in lettura permette di conoscere il contenuto di una directory attraverso un programma **`ls`** o simile, senza questo permesso, la directory può ugualmente essere attraversata;
- in scrittura permette di modificarne il contenuto, cioè di aggiungere o eliminare file e altre directory;
- in esecuzione permette il suo attraversamento, ovvero permette di raggiungere il suo contenuto o quello di altre directory discendenti.

38.1.1 $ 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 *umask*. Il valore tipico di *umask* 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 | --mode=`⟨*modo*⟩

Permette di definire esplicitamente i permessi attribuiti alla o 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`** (39.2.5).

`-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 la 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`**.

38.1.2 $ 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.

38.2 Percorsi

Il ***percorso*** o *path* è il modo con cui si identifica la posizione di un certo file o 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. In tal caso si usa spesso la dizione ***nome di percorso*** o *pathname*.

38.2.1 $ 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 *link* 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.

38.2.2 $ basename

`basename` ⟨*pathname*⟩ `[`⟨*suffisso*⟩`]`

Estrae il nome di un file da un *pathname*, cioè da un nome contenente l'informazione del percorso e del file. In pratica: rimuove da *pathname* 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
```

38.2.3 $ dirname

`dirname` ⟨*pathname*⟩

Estrae il nome della directory da un *pathname*, cioè da un nome contenente l'informazione del percorso e del file. In pratica: rimuove da *pathname* la parte finale a partire dall'ultima barra obliqua (**`/`**) di divisione tra l'informazione della directory e il nome del file. Se il *pathname* contiene solo un nome di file, il risultato è un singolo punto (`.`), cioè la directory corrente. Il risultato viene emesso attraverso lo *standard output*.

Esempi

$ ***dirname "/idrogeno/ossigeno"***[*Invio*]

```
/idrogeno
```

38.2.4 $ namei

`namei [`⟨*opzioni*⟩`]` ⟨*pathname*⟩`...`

Scompone un *pathname* finché raggiunge un punto terminale. In pratica vengono analizzati i *pathname* forniti, ne viene scomposto e descritto il contenuto nelle varie (eventuali) sottodirectory, e infine, se tra gli elementi contenuti nei *pathname* richiesti esistono dei *link* simbolici, viene visualizzato anche l'elemento di destinazione. Questo programma è particolarmente utile per seguire i *link* simbolici, soprattutto quando questi hanno troppi livelli, cioè quando un *link* punta a un altro *link* ecc. I vari elementi visualizzati sono preceduti da una lettera che ne descrive le caratteristiche:

- **`f:`** il *pathname* che si sta analizzando;

- **d** directory;
- **l** *link* simbolico;
- **s** *socket*;
- **b** *block device* (dispositivo a blocchi);
- **c** *character device* (dispositivo a caratteri);
- **-** file normale;
- **?** errore.

Esempi

$ ***namei /usr/bin/X11***

Genera il seguente risultato:

```
f: /usr/bin/X11
 d /
 d usr
 d bin
 l X11 -> ../X11R6/bin
   d ..
   d X11R6
   d bin
```

dal quale si intende che la directory `/usr/bin/X11` in realtà non esiste, e che si tratta di un *link* simbolico alla vera directory `/usr/X11R6/bin/`.

38.2.5 $ pathchk

`pathchk [`⟨*opzioni*⟩`] [`⟨*pathname*⟩`... ]`

Per ogni *pathname* 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 *pathname*, non ha il permesso di esecuzione necessario per essere attraversata;
- la lunghezza totale di un *pathname* è maggiore di quella gestibile con quel tipo di *filesystem*;
- la sola lunghezza di uno degli elementi di un *pathname* 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.

Exit status

- **0** se tutti i *pathname* 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'
```

38.3 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) effettivamente esistenti è compito della *shell*: è cioè qualcosa a cui gli altri programmi sono normalmente estranei. Nella sezione 30.3.8 viene trattato il modo con cui la *shell* **bash** si comporta al riguardo.

Il contenuto di una directory viene normalmente analizzato 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.

38.3.1 $ 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 una 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, il numero di *hard link*, il nome del 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 eventualmente contenuti 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). Se viene usato il formato lungo di visualizzazione (**`-l`**), viene visualizzata questa data, se viene usato insieme a **`-t`**, l'elenco viene ordinato in base a questa data.

`-u` | `--time=atime` | `--time=access` | `--time=use`

Utilizza la data di accesso ai file. Se viene usato il formato lungo di visualizzazione (**`-l`**), viene visualizzata questa data, se viene usato 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 cosiddette copie di *backup* dei file.

`-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;
- **`@`** *link* simbolico;

[1] È importante ricordare che se vengono indicati dei nomi di file e/o directory nella riga di comando, è compito della *shell* espandere eventuali caratteri jolly. Di conseguenza, in questo caso, è la *shell* a non fornire a **`ls`** i nomi che iniziano con un punto.

- | *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. Questa è la modalità normalmente predefinita.

`-w` *n* | `--width` *n*

Definisce la larghezza a disposizione per l'elenco. Il valore *n* 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 espressamente indicati nella riga di comando.

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`**.

38.3.2 $ 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 consta 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)`**.

38.3.3 $ 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.

38.3.4 $ du

`du [⟨opzioni⟩] ⟨file⟩...`

du (*Disk Usage*) emette una sorta di statistica dell'utilizzo dello spazio da parte di un elenco di file e/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** l'eventuale presenza 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.

38.4 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
```

38.4.1 $ 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 determinato comando, sia quale programma viene effettivamente scelto nel caso ne esistano diversi con lo stesso nome collocati in posizioni diverse 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** (33.1.43).

[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**.

38.4.2 $ 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 39.1 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* Ext2.
lsattr	Elenca gli attributi particolari di un *filesystem* Ext2.
touch	Cambia la data e l'orario di accesso e/o di modifica.

Tabella 39.1: Riepilogo dei programmi e dei comandi per la gestione delle proprietà, dei permessi e degli attributi di file e directory.

39.1 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 meglio definito.

39.1.1 $ 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 (`:`) 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 miofile`***

L'utente *root* cambia l'utente proprietario del file `miofile`, facendo in modo che diventi **`tizio`**.

`#` ***`chown tizio.users miofile`***

L'utente *root* cambia l'utente e il gruppo proprietario del file `miofile`, facendo in modo che diventino **`tizio`** e **`users`**.

`$` ***`chown .users miofile`***

L'utente proprietario del file `miofile` cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

39.1.2 $ 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 miofile`***

L'utente proprietario del file `miofile` cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

39.2 Permessi

I permessi, attribuiti a un file o a una directory, definiscono le operazioni che con questo possono essere compiute a seconda dell'utente. La gestione di questi è già stata introdotta nella sezione 3.7.5.
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 accedere ai file in essa 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 bit* (*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.

39.2.1 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 *home*.

```
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 a utilizzare l'*account* normale.

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 normale. Si può comprendere quindi, quanto sia pericoloso utilizzare questi permessi, SUID e SGID, senza oculatezza.

39.2.2 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 una ipotesi astratta: l'idea di attribuire i permessi SUID e SGID a una *shell* è semplicemente terribile.

39.2.3 S-bit e le directory

I tre bit 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 generalmente attribuito 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.

39.2.4 umask

Quando viene creato un file, questo appartiene automaticamente all'utente che lo crea e al gruppo a cui appartiene l'utente. I permessi gli vengono attribuiti in base a una maschera, detta *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** (33.1.45).

39.2.5 $ chmod

`chmod [`⟨*opzioni*⟩`]` ⟨*modo*⟩ ⟨*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 **ugoa** 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 **rwxXstugo** 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 predefinita (*umask*) che può essere modificata attraverso il comando di *shell* **`umask`** (33.1.45). In pratica, la variazione riguarda tutti i tipi di utente, a esclusione dei bit attivati nella maschera predefinita.

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 SUID e/o 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 lo *sticky bit*.

`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 *link* simbolici: se si interviene su un *link* 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 *home* e anche nelle sottodirectory successive.

39.3 Attributi

Le caratteristiche standard di un file in un sistema Unix sono le proprietà e i permessi. In alcuni casi è possibili attribuire altri attributi come quando si utilizza il *filesystem* Ext2. È naturalmente compito del *kernel* fare in modo che questi attributi siano gestiti in modo corretto.

39.3.1 $ chattr

`chattr [`⟨*opzioni*⟩`] [`⟨*modo*⟩`]` ⟨*file*⟩`...`

Cambia gli attributi su un *filesystem* di tipo Ext2 (*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 *hard link* verso di esso (i *link* 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).

39.3.2 $ lsattr

`lsattr [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`

Elenca gli attributi dei file su un *filesystem* Ext2 (*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.

39.4 Data

Tutti i file riportano tre indicazioni data-orario:

- **`ctime`** la data e l'ora di creazione: 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.

39.4.1 $ 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.

Copia, link, 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 40.1 elenca i programmi a cui si accenna in questo capitolo.

Programma	Descrizione
cp	Copia.
ln	Crea dei *link*.
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.

Tabella 40.1: Riepilogo dei programmi per la copia, il *link*, lo spostamento, la cancellazione e l'archiviazione di file e directory.

40.1 Copia e link

La copia genera un altro file o un'altra directory, il *link* 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 *link* sono molto importanti e vengono usati di frequente. Si distinguono due tipi di questi: *link* simbolici e *link hard*. Attraverso il *link hard* si creano dei riferimenti a dati esistenti in modo non distinguibile da quelli originali; i *link* simbolici sono dei file speciali e per questo distinguibili dai file originali.
A fianco del problema della copia di file (o 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).

40.1.1 Link simbolici

Si è detto che i *link* 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 *link*, 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 *link* simbolico. Alla fine, viene indicato anche a chi punta il *link*: **`-> /bin/sh`**.
Si può osservare inoltre che i permessi di un *link* 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 *link* simbolico.

> L'esistenza dei *link* simbolici altera la logica normale della copia: ha senso copiare i file a cui i *link* puntano o ha senso copiare i *link*? Solitamente si considera che la gestione dei *link* 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 *link*.

40.1.2 Link hard

La gestione dei *link hard* è più seria, nel senso che deve essere riservata a situazioni di particolare necessità.
Si è detto che attraverso il *link hard* 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 altri file. In altre parole, se si utilizzano i *link hard*, un file viene effettivamente cancellato quando sono stati eliminati tutti i riferimenti a questo.

Per comprendere in pratica cosa accade, si può provare a eseguire gli esempi seguenti.

$ ***touch miofile***[*Invio*]

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

```
-rw-rw-r--   1 tizio    tizio           0 Mar  2 10:48 miofile
```

$ ***ln miofile tuofile***[*Invio*]

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

```
-rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 miofile
-rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 tuofile
```

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 *link*.

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 miofile tuofile***[*Invio*]

```
270385 -rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 miofile
270385 -rw-rw-r--   2 tizio    tizio           0 Mar  2 10:48 tuofile
```

Come si vede, i due file hanno lo stesso *inode* (il numero che appare prima dei permessi), quindi **sono lo stesso file**.

40.1.3 Directory e link hard

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 singolo punto (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 (*link hard*).

La directory precedente, rappresentata dai due punti in sequenza (`..`), ha invece tre riferimenti totali, perché esiste questa directory (in pratica: i due riferimenti naturali, più questo, perché esiste questa directory).

40.1.4 $ 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.

Alcune opzioni

`-a | --archive`

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

`-b | --backup`

Mantiene delle copie di *backup* per i file che vengono sovrascritti con la copia.

`-d | --no-dereference`

Copia i *link* simbolici mantenendoli come tali, invece di copiare il file a cui i *link* 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 *hard link* invece di copiare i file (non vale per le directory).

`-s | --symbolic-link`

Crea un *link* 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 backup. Se non viene specificato con questa opzione, si utilizza il simbolo contenuto nella variabile di ambiente **`SIMPLE_BACKUP_SUFFIX`** e 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 *backup*, quando si usa anche l'opzione **`-b`**. Il tipo di *backup* viene definito in base ai termini seguenti.

Tipo di backup

`t | numbered`

Esegue un *backup* numerato.

`nil | existing`

Esegue un *backup* numerato per i soli file che hanno già almeno una versione di *backup*.

`never | simple`

Esegue un *backup* semplice.

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 *backup* (ovvero 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 di *backup*.

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.

Attenzione

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 **--force**.

40.1.5 $ ln

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

ln [⟨*opzioni*⟩] ⟨*origine*⟩... ⟨*directory*⟩

Crea un collegamento (*link*) tra file e/o directory. Se viene specificata una origine e una destinazione, il nome indicato come destinazione sarà il nuovo *link* 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 all'interno della directory di destinazione verranno creati tanti *link* quanti sono i nomi indicati come origine. Se non viene specificato diversamente attraverso le opzioni, vengono creati degli *hard link* e non dei *link* simbolici.

Alcune opzioni

`-b | --backup`

`-f | --force`

`-i | --interactive`

`-S | --suffix`

`-V | --version-control`

Le opzioni sopra indicate funzionano nello stesso modo di **cp** (40.1.4).

`-s | --symbolic-link`

Crea un *link* simbolico invece di crearne uno *hard*.

`-d | -F | --directory`

Permette all'utente *root* di creare un *hard link* per una directory.

`-n | --no-dereference`

Quando la destinazione corrisponde a un *link* simbolico preesistente che punta verso una directory, il funzionamento normale prevederebbe la creazione del *link* in quella directory. Usando questa opzione si intende evitare questo, rimpiazzando quel *link* simbolico. Per poter attuare ciò, occorre però utilizzare anche l'opzione **--force**.

Variabili

ln utilizza le variabili di ambiente **VERSION_CONTROL** e **SIMPLE_BACKUP_SUFFIX** nello stesso modo di **cp** (40.1.4).

Esempi

`$ ln -s /bin/ls ~/elenco`

Crea il *link* simbolico `elenco` all'interno della directory *home* che punta al comando **ls**. Eseguendo `~/elenco` si ottiene in pratica di eseguire il comando **ls**.

`$ ln /bin/ls ~/elenco`

Crea un *hard link* `elenco` all'interno della directory *home* che punta al comando **ls**. Eseguendo **~/elenco** si ottiene in pratica di eseguire il comando **ls**.

`$ ln -s /bin/* ~/`

Crea una serie di *link* simbolici all'interno della directory *home* per tutti i file contenuti in `/bin`.

`$ ln -s /bin ~/binari`

Crea il *link* simbolico ~`/binari` alla directory `/bin/`. Eseguendo **`cd ~/binari`** ci si ritroverà in `/bin/`.

40.1.6 $ install

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

`install [`⟨*opzioni*⟩`] -d` ⟨*directory*⟩`...`

Copia di file con attribuzione dei permessi e delle proprietà. Il comando 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 tipicamente usato per l'installazione di programmi.

Alcune opzioni

`-b` | `--backup`

`-S` | `--suffix`

`-V` | `--version-control`

Le opzioni sopra indicate funzionano nello stesso modo di **`cp`** (40.1.4).

`-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` ⟨*modo*⟩ | `--mode=`⟨*modo*⟩

Definisce i permessi in modo analogo alla sintassi di **`chmod`** (39.2.5).

`-o` ⟨*proprietario*⟩ | `--owner=`⟨*proprietario*⟩

Definisce l'utente proprietario dei file installati o delle directory.

40.1.7 $ 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 e/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 attraverso un 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 *newline*. Questi *record* vengono troncati o allungati in modo da corrispondere alla dimensione indicata attraverso l'opzione **`cbs`**. Alla fine, i caratteri *newline* 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 il codice *newline*.

`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 un unico carattere. <*LF*>.

`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 *short* (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 normalmente usato 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 (precedentemente inizializzato). 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 1440KB (e naturalmente che il file di 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 512KB: 2*18*512 = 18KB. Specificando l'opzione **obs=18k** si intende fare in modo che **dd** fornisca al dispositivo `/dev/fd0` blocchi di quella dimensione in modo 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.

40.2 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 la 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* Ext2 (*Linux native*) 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 40.2.3 viene mostrato uno *script* in grado di gestire agevolmente una sorta di cestino del genere.

40.2.1 $ mv

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

Sposta i file e/o le directory. Se vengono specificati solo i nomi di due elementi (file o directory), il primo viene spostato e/o 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 diversi *filesystem*, vengono spostati solo i cosiddetti file regolari (quindi: no *link*, no directory).

Alcune opzioni

`-b | --backup`

`-f | --force`

`-i | --interactive`

`-S | --suffix`

`-V | --version-control`

Le opzioni sopra indicate funzionano nello stesso modo di **cp** (40.1.4).

Attenzione

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 **--force**.

40.2.2 $ 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 eventuali altre directory contenute non vengono rimossi prima, ma 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 una 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 `{\ttfamily{}-\/}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 *home*, 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 un 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**.

40.2.3 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 *home*.

```
#!/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
    # utilizzata come 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 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 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 diversi filesystem.
    #-----------------------------------------------------------------
    if ! mv $* $CESTINO/$RICICLO 2> /dev/null
    then
        #-------------------------------------------------------------
        # Essendo fallito lo spostamento, almeno in parte, si procede
        # con la copia.
        # La copia viene fatta senza dereferenziare i link.
        #-------------------------------------------------------------
        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.
#=====================================================================
```

40.3 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 attraverso 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.

Questo argomento viene ripreso anche nel capitolo dedicato alle copie di sicurezza (121).

40.3.1 $ 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*

  Dallo *standard input* viene letto un elenco di nomi di file (uno per riga) e questi file (con il loro contenuto) vengono copiati nello *standard output*.

- *copy-in*

  Dallo *standard input* viene letto il contenuto di un archivio dal quale si possono estrarre i file in esso contenuti.

- *copy-pass*

  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 **`info cpio`** o **`cpio(1)`**.

Alcune opzioni

`-o | --create`

copy-out

`-A | --append`

Aggiunge dati a un archivio esistente che deve essere specificato con l'opzione **`-O`**.

`-L | --dereference`

Quando incontra dei *link* simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i *link*.

`-O` ⟨*nome-archivio*⟩

Specifica il nome dell'archivio da creare o incrementare, invece di utilizzare lo *standard output*.

`-i | --extract`

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.

`-p | --pass-through`

copy-pass

`-d | --make-directories`

Crea le directory necessarie.

`-l | --link`

Se possibile, crea dei *link* invece di copiare i file.

`-L | --dereference`

Quando incontra dei *link* simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i *link*.

40.3.2 $ 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*.

Per la descrizione completa di questo programma, conviene consultare **`tar(1)`**.

Alcune opzioni

opzioni di funzionamento

`-A | --catenate | --concatenate`

Aggiunge dei file **`tar`** a un archivio già esistente.

`-c | --create`

Crea un nuovo archivio.

`-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 | --append`
Aggiunge dati a un archivio già esistente.

`-t | --list`
Elenca il contenuto di un archivio.

`-u | --update`
Aggiunge solo i file più recenti rispetto a quanto già contenuto nell'archivio.

`-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 *link* 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` ⟨*Kilobyte*⟩ | `--tape-length=`⟨*Kilobyte*⟩
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 nello *standard output*.

`-p | --same-permissions | --preserve-permissions`
Estrae tutti i permessi associati ai file.

`-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 le proprietà 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`**.

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 1440KB).

> Con la copia multivolume, come in questo caso, non è possibile utilizzare la compressione automatica attraverso l'opzione **`-z`** o **`-Z`**.

$ ***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 le opzioni **-p** e **--same-owner** siano già predefinite a causa di una opportuna modifica fatta da chi ha realizzato la distribuzione Linux. In ogni caso vale la pena di ricordarsene.

$ ***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**.

40.3.3 $ 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 di dati (*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 **info gzip** 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.

40.3.4 $ gunzip

gunzip come programma autonomo non esiste: è un *link* a **gzip**. Se **gzip** viene avviato con il nome **gunzip** si comporta come se fosse stata utilizzata l'opzione **-d**.

Il *link* verso **gzip** può essere sia simbolico che non.

40.3.5 $ zcat

zcat come programma autonomo non esiste: è un *link* a **gzip**. Se **gzip** viene avviato con il nome **zcat** si comporta come se fosse stata utilizzata l'opzione **-c**.

Il *link* verso **gzip** può essere sia simbolico che non.

40.3.6 Copie di sicurezza

Quello che segue è un 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>
#
# 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
        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 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 falso.
            #-------------------------------------------------------
            exit 1
        fi
    fi
    #---------------------------------------------------------------
    # Giunti a questo punto, dovrebbe esistere la directory
    # di destinazione.
    # Inizia il loop 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'' e in particolare:
        # ``-z'' permette di comprimere automaticamente l'archivio
        #        attraverso ``gzip'';
        #-----------------------------------------------------------
        tar -c -f $DESTINAZIONE/$BASE_NAME.tgz -v -z \
--same-permissions --same-owner --same-order $DA_ARCHIVIARE
    done
    #---------------------------------------------------------------
    # L'operazione di archiviazione e' terminata.
    #---------------------------------------------------------------
    echo "L'archiviazione e' terminata."

#===================================================================
# Fine.
#===================================================================
```

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 e/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.

41.1 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 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.

41.1.1 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 41.1 elenca le opzioni principali.

Opzione	Descrizione
–G	Utilizza una espressione regolare elementare (comportamento predefinito).
–E	Utilizza una 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 il nome dei file per i quali la ricerca ha avuto successo.
–L	Emette solo il nome dei file per i quali la ricerca non ha avuto successo.
–v	Inverte il senso della ricerca: valgono le righe che non corrispondono.

Tabella 41.1: Opzioni principali di **`grep`**.

41.1.2 Espressioni regolari

Una 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 normalmente 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.

41.1.3 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 inclinata rovescia (\).

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 una 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.

Una 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 41.2 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 41.2: Operatori di ripetizione nelle espressioni regolari gestite dal programma **`grep`**.

Due espressioni regolari possono essere concatenate (in sequenza) generando una 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* rappresenta 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.

41.1.4 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 inclinata rovescia: **`?`**, **`+`**, \\{, \\|, **`(`** e **`)`**.

41.1.5 egrep

Come già accennato, alcune distribuzioni di Linux forniscono un programma **`egrep`** alternativo, invece di utilizzare il solito *link* 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 `\{`.

41.1.6 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 **bash** 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 **bash** fornisce a **grep** la stringa `\'.*\'` che fortunatamente viene ugualmente interpretata da **grep** nel modo corretto.

41.2 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 **info find(1)** oppure **find(1)**.

41.2.1 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.

41.2.2 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.

41.2.3 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 all'interno delle 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.

41.2.4 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 *link* simbolico e il suo contenuto corrisponde al modello che può essere espresso utilizzando anche i caratteri jolly. Un *link* 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 *link*.

`-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 (Kilobyte);
- **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 35.2).

`-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** block;
- **c** carattere;
- **d** directory;
- **p** *pipe*;
- **f** file normale (*regular file*);
- **l** *link* simbolico;
- **s** *socket*.

41.2.5 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 ad 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 una operazione AND è *Falso* basta sapere che almeno una delle due condizioni in ingresso ha un valore *Falso*; per sapere che il risultato di una 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 una 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.

41.2.6 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 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 directory) che avverano l'insieme delle condizioni. È l'azione predefinita, se non ne vengono indicate delle altre.[1]

41.2.7 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 doppie virgolette.

\# ***find / -xdev -nouser -print***

Esegue una ricerca nel *filesystem root* a partire dalla directory radice, escludendo gli altri *filesystem*, per i file e le directory appartenenti a utenti non registrati (privi di *account*).

$ ***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 root*, 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 root*, 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.

---

[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 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 *link*, sono abbastanza comprensibili, mentre tutto il resto viene generalmente indicato 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 Linux.
La tabella 42.1 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 42.1: 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.

42.1 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.

42.1.1 $ mkfifo

`mkfifo [`⟨*opzioni*⟩`]` ⟨*file*⟩`...`
`mkfifo` crea uno o più file FIFO (*pipe* con nome).

Alcune opzioni

`-m [`⟨*modo*⟩ `| --mode=[`⟨*modo*⟩

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`** (39.2.5). Il valore predefinito di questi permessi è 0666, meno il valore della maschera *umask*.

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 miofile | sort | lpr`**.

`$` ***`mkfifo fifo1 fifo2`***

Crea due file FIFO: `fifo1` e `fifo2`.

`$` ***`cat miofile >> fifo1 &`***

Invia `miofile` 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`**.

42.2 File di dispositivo

I file di dispositivo sono riferimenti a *driver* contenuti nel *kernel*. Nei sistemi Unix, questi file di dispositivo devono indicare due numeri, detti ***maggiore*** e ***minore***, 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 Linux.
Dal momento che questi file servono solo in quanto contengono i numeri maggiore e minore di un certo dispositivo, potrebbero funzionare anche collocati al di fuori della loro directory tradizionale, utilizzando anche nomi differenti. Questa possibilità viene sfruttata da alcune distribuzioni 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.

Dal momento 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 una distribuzione Linux e un'altra. Inoltre, l'esistenza di utenti e gruppi fittizi, con nomi come **`floppy`**, **`sys`**, **`daemon`** e altri, dipende proprio da questa esigenza di controllo dell'accesso ai dispositivi.

42.2.1 # mknod

`mknod [`⟨*opzioni*⟩`]` ⟨*file*⟩ ⟨*tipo*⟩ `[`⟨*maggiore*⟩ ⟨*minore*⟩`]`

`mknod` permette di creare un file FIFO oppure un file di dispositivo. Il tipo di file viene indicato attraverso una lettera, e i numeri maggiore e minore 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 [`⟨*modo*⟩ `| --mode=[`⟨*modo*⟩

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`** (39.2.5). Il valore predefinito di questi permessi è 0666, meno il valore della maschera *umask*.

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 `hda19`, nella directory corrente, utilizzando dei permessi opportuni.

42.2.2 /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 minore e maggiore; occorre anche stabilire i permessi corretti, l'utente e il gruppo proprietari. Trascurando questi particolari, si rischierebbe di aprire gravi buchi nella sicurezza del sistema.

In questo senso, questo *script* è diverso da un sistema 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.

42.3 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`** *link* simbolico;
- **`p`** *pipe* con nome, o FIFO;
- **`c`** dispositivo a caratteri;
- **`b`** dispositivo a blocchi.

Parte XI

Programmi di utilità vari

Appunti Linux

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.

<*ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/*>

<*ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/*>

<*ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/*>

La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.

<*http://www.pluto.linux.it/ildp/AppuntiLinux/*>

<*http://ildp.psy.unipd.it/AppuntiLinux/*>

<*http://www.panservice.it/mirror/ILDP/AppuntiLinux/*>

<*http://linux.calion.com/pluto/ildp/AppuntiLinux/*>

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, convenzionalmente separata in righe di lunghezze diseguali, e per questo terminate attraverso un codice particolare.
La tabella 43.1 elenca i programmi a cui si accenna in questo capitolo.

Programma	Descrizione
cat	Emissione di file.
tac	Emissione inversa delle righe.
nl	Emissione con numerazione delle righe.
od	Conversione in ottale.
rev	Emissione 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	Conteggio delle parole.
sum	Calcola un *checksum* (obsoleto).
cksum	Calcola un *checksum*.
md5sum	Calcola un *checksum* 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 43.1: Riepilogo dei programmi per la gestione dei file di testo.

43.1 Newline

Questo codice di conclusione delle righe 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 conclusione di riga in modo astratto, cioè senza volere restare legati a una particolare architettura di sistema, si parla di *newline*.
La maggior parte dei programmi di utilità generale utilizzati all'interno di Linux è di provenienza GNU. Gli scopi del progetto GNU fanno sì che l'intento principale nella realizzazione di questi programmi sia la loro massima portabilità. In questo senso, nella documentazione di questi si fa sempre riferimento a codici *newline*, lasciando all'utente l'onere di sapere cosa significhi questo nel proprio sistema operativo.
Per quanto riguarda Linux, *newline*=*<LF>*.

43.2 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 *newline* (**`0x0A`**).

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.
Newline 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 *newline*. Ma una riga vuota, *blank line*, può essere veramente vuota e contenere soltanto il codice *newline*.

43.3 Programmi di utilità per i file di testo

I programmi descritti nelle sezioni seguenti, che fanno parte delle *GNU text utilities* (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*.

43.4 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*.

43.4.1 $ 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. Sono esclusi <*Tab*> e *newline*.

`-E | --show-ends`

Visualizza la fine di ogni riga aggiungendo il simbolo **`$`**.

`-T | --show-tabs`

Mostra esplicitamente il carattere <*Tab*> utilizzando il simbolo **`^I`**.

`-A | --show-all`

Mostra tutti i simboli non stampabili. È equivalente a **`-vET`**.

43.4.2 $ tac

`tac [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

`tac` emette attraverso lo *standard output* i file forniti come argomento, invertendo l'ordine dei *record*. In pratica, **`tac`** è l'inverso di **`cat`**.
In linea di massima si può dire che i *record* corrispondano alle righe, e ciò finché non viene definito un tipo di separatore dei *record* diverso dal solito *newline*.

Alcune opzioni

`-s` ⟨*separatore*⟩ `| --separator=`⟨*separatore*⟩

Permette di definire il separatore da usare per distinguere i *record*.

43.4.3 $ 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: testata, corpo e piede.
Vedere **`info nl`** oppure **`nl(1)`**.

43.4.4 $ od

`od` converte i file forniti come input in ottale o in altri formati.
Vedere **`info od`** oppure **`od(1)`**.

43.4.5 $ rev

`rev [⟨file⟩... ]`

Emette attraverso lo *standard output* il file fornito come argomento, rovesciando l'ordine dei caratteri di ogni riga.

43.5 Rimpaginazione

Alcuni programmi si occupano di modificare l'impaginazione del testo, cambiandone la larghezza o aggiungendo delle intestazioni.

43.5.1 $ fmt

`fmt [⟨opzioni⟩] [⟨file⟩... ]`

Formatta il testo contenuto nei file forniti come argomento, eliminando e aggiungendo codici *newline*, in modo che il testo risulti al massimo di una data larghezza. Il risultato viene emesso attraverso lo *standard output*. Se non viene specificata l'ampiezza massima della riga, questa si intende essere di 75 caratteri.

Alcune opzioni

`-w ⟨ampiezza⟩ | --width=⟨ampiezza⟩`
: Determina esplicitamente l'ampiezza massima del testo.

Vedere **`info fmt`** oppure **`fmt(1)`**.

43.5.2 $ 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 una testata di 5 righe, dove la terza di queste contiene l'indicazione della data e dell'ora di stampa e 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*> eventualmente contenuto nei file di input, genera un salto pagina in uscita.

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⟩]`
: Stampa 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`

Stampa raddoppiando 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 **`info pr`** oppure **`pr(1)`**.

43.5.3 $ fold

`fold [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

Formatta il testo contenuto nei file forniti come argomento, suddividendo le linee troppo lunghe inserendo un codice *newline* 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 <*Tab*>, <*BS*> e <*CR*> come gli altri.

`-s | --spaces`

Cerca di interrompere le righe subito dopo uno di spazio vuoto, in modo da evitare di spezzare delle parole.

`-w` ⟨*larghezza*⟩ `| --width=`⟨*larghezza*⟩

Definisce la larghezza massima della colonna di testo finale.

43.5.4 $ 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.

43.5.5 $ 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.

43.6 Formattazione

Un tipo particolare 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 viene normalmente interpretato come arretramento di una posizione. Gli effetti normali che si possono ottenere in questo modo sono il sottolineato e il grassetto.

Per esempio, per ottenere la parola **GRA** in grassetto, si utilizza una sequenza del tipo: **G** <***BS***> **G R** <***BS***> **R A** <***BS***> **A**.

43.6.1 $ 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 e/o 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 non vengono quindi eliminati.

`-x`

I caratteri di tabulazione vengono sostituiti con spazi.

43.6.2 $ 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 particolarmente orientato alla 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.

43.7 Estrazione parziale

Alcuni programmi si occupano di estrarre dall'input solo alcune porzioni o di suddividere l'input in parti più piccole.

43.7.1 $ head

`head [⟨opzioni⟩] [⟨file⟩... ]`

Emette attraverso lo *standard output* la prima parte (le prime 10 righe se non viene specificato diversamente nelle 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 1MB.

`-n ⟨dimensione⟩ | --lines=⟨dimensione⟩`

Emette la quantità di righe indicata dalla dimensione.

43.7.2 $ tail

`tail [⟨opzioni⟩] [⟨file⟩... ]`

Emette attraverso lo *standard output* la parte finale (le ultime 10 righe se non viene specificato diversamente nelle 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 1MB.

`-n` ⟨*dimensione*⟩ | `--lines=`⟨*dimensione*⟩
Emette la quantità di righe indicata dalla dimensione.

43.7.3 $ 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 ordine 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 1MB.

`-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**.

43.7.4 $ 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 come argomento.
Vedere **info csplit** oppure **csplit(1)**.

43.8 Controlli sommari

Alcuni programmi si occupano di calcolare somme di vario genere in base al contenuto dell'input. Può trattarsi del conteggio di determinati elementi o di *checksum*.

43.8.1 $ wc

`wc [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

Emette attraverso lo *standard output* la statistica sul conteggio dei caratteri *newline* (in pratica il numero delle righe), delle parole e dei byte. Se attraverso le opzioni vengono specificati uno o due tipi di conteggio, quelli non espressamente indicati, 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 di caratteri *newline*).

43.8.2 $ sum

`sum [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

sum calcola un *checksum* 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.

43.8.3 $ cksum

`cksum [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

`cksum` calcola un *checksum* CRC (*Cyclic Redundancy Check*) per ogni file fornito negli argomenti.
Non utilizza opzioni, tranne quelle standard.

43.8.4 $ md5sum

`md5sum [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

`md5sum` calcola un *checksum* MD5 (*Message Digest*) a 128 bit per ogni file fornito negli argomenti.
Vedere **`info md5sum`** oppure **`md5sum(1)`**.

43.9 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.

43.9.1 $ 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.

Alcune opzioni

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 dovevano essere già ordinati in precedenza. Nel caso non lo fossero, si può sempre usare la modalità di ordinamento normale. L'utilizzo di questa opzione fa risparmiare tempo quando la situazione lo consente.

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 del set ASCII puro (da 0x30 a 0x7E compresi).

`-n`

Esegue una comparazione, o un ordinamento, 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-di-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.

Vedere **`info sort`** oppure **`sort(1)`**.

43.9.2 $ 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 **`info uniq`** oppure **`uniq(1)`**.

43.9.3 $ comm

`comm [`⟨*opzioni*⟩`] [`⟨*file1*⟩`]` ⟨*file2*⟩

Confronta due file ordinati e emette nello *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.

43.9.4 $ 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)`**.

43.10 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.

43.10.1 $ 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 -m ":" -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.

43.10.2 $ 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 carattere *newline*.

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.

43.10.3 $ 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 **`info join`** oppure **`join(1)`**.

43.11 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.

43.11.1 $ tr

`tr [`⟨*opzioni*⟩`]` ⟨*stringa1*⟩ `[`⟨*stringa2*⟩`]`

Copia lo *standard input* nello *standard output* trasformandolo in vari modi possibili. 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 **`info tr`** oppure **`tr(1)`**.

43.11.2 $ expand

`expand [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

Emette sullo *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*⟩`...]`

Premette 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.

`-i` | `--initial`

Converte solo i caratteri di tabulazione iniziali, cioè quelli che precedono caratteri diversi da spazio, o che precedono altri caratteri di tabulazione.

43.11.3 $ 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*⟩`...]`

Premette 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.

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 detto *patch*, e queste *patch* si applicano a un file, o a una serie di file, per ottenere altrettanti file aggiornati in base alle modifiche contenute nelle *patch* stesse.

Esistono tanti modi di costruire un file di *patch*. 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. Le *patch* più frequenti riguardano file di testo normale, ed è di questo che si vuole trattare in questo capitolo.

44.1 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*.

44.1.1 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 *newline*.

44.1.2 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.

44.1.3 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`**.

Esempi

Per verificare in pratica il funzionamento di **`diff`** per 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.

Annotazioni

Quando si vogliono distribuire file di *patch* 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.

44.1.4 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 *patch*. 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.

44.1.5 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.

44.1.6 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`**. Quello 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 testate 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 eventualmente contenute.

44.2 patch

Il programma più adatto per applicare delle modifiche è **patch**, che 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 *email*, senza doverli estrapolare.

patch ⟨*opzioni*⟩ < ⟨*file-diff*⟩

patch utilizza quindi lo *standard input* per ricevere i file di modifiche. Questi devono contenere l'indicazione del file da modificare, e per questo si possono utilizzare solo i formati *diff* contestuali.

Il file originale viene rinominato in modo da ottenere 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 delle modifiche. Se qualche blocco di modifiche non può essere applicato, questi rigetti vengono indicati in un file terminante con l'estensione `.rej`.

44.2.1 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 tali modifiche cono state costruite. 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 *email* o per una modifica attraverso un programma per l'*editing* 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 seguente.

`-l | --ignore-white-space`

In questo modo, una qualunque sequenza di spazi equivale a un'altra qualunque sequenza di spazi, indipendentemente dal fatto che siano stati usati caratteri di tabulazione, o caratteri spazio veri e propri, e indipendentemente dalla loro quantità.

44.2.2 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`.

44.2.3 Directory corrente

L'applicazione di una modifica richiede che il programma **patch** venga avviato quando ci si trova nella directory corrente adatta. Per esempio, se si vuole utilizzare un file di modifiche in cui l'intestazione è come quella seguente,

```
--- primo       Tue Mar  3 08:12:30 1998
+++ secondo     Wed Mar  4 11:16:32 1998
...
```

si intende che le modifiche si applichino al file `primo` contenuto nella directory corrente. Nello stesso modo, quando si ha a che fare con modifiche che intervengno su più file di una directory, come nell'esempio seguente,

```
--- uno/primo   Thu Mar  5 09:48:10 1998
+++ due/primo   Fri Mar  6 08:30:07 1998
...
```

si intende che tale directory (`uno/` in questo caso) sia una sottodirectory di quella corrente.

44.3 Riferimenti

La documentazione **info** riguardo alla creazione, distribuzione e applicazione di *patch*, è molto dettagliata: **info diff**.

Programmi di utilità vari

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 45.1 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 una espressione condizionale e ne restituisce il risultato.
expr	Valuta una 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 45.1: Elenco di alcuni programmi di utilità per uso vario.

45.1 Emissione di testo

Alcuni programmi si occupano di emettere del testo, più o meno formattato, attraverso lo *standard output*. In pratica, queste informazioni vengono normalmente visualizzate, quando non si fa una ridirezione.

45.1.1 $ echo

`echo` [⟨*opzioni*⟩] [⟨*stringa*⟩`...`]

Emette le stringhe indicate come argomento, separate da uno spazio e con l'aggiunta di un codice *newline* finale.

Opzioni

`-n`

Sopprime il codice *newline* 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*;
- **\c** eliminazione del carattere di *newline* finale;
- **\f** *form feed* o <*FF*>;
- **\n** *newline* o <*LF*>;
- **\r** *carriage return* o <*CR*>;
- **\t** tabulazione (orizzontale) o <*HT*>;
- **\v** tabulazione verticale;
- \\ barra inclinata rovescia;
- ***nnn*** il carattere il cui codice ottale corrisponde al numero indicato da *nnn*.

45.1.2 $ 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).

45.1.3 $ yes

`yes [`⟨*stringa*⟩`... ]`

`yes` emette senza fine, attraverso lo *standard output*, le stringhe indicate come argomento (separate da uno spazio l'una dall'altra), seguite dal codice *newline*. 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 da *newline*.

`$` ***`yes n`***[*Invio*]

```
n
n
n
n
n
...
```

Se vengono specificate delle stringhe come argomento, queste stringhe vengono emesse ripetitivamente.

`$` ***`yes`*** | ***`mioprog`***

Si inviano una serie di lettere **`y`**, seguite da *newline*, al programma ipotetico **`mioprog`** che probabilmente tende a fare una serie di domande alle quali si vuole rispondere sempre con una lettera **`y`**.

45.2 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*).

45.2.1 $ false

`false`

`false` è un programma banale che restituisce semplicemente il valore 1, corrispondente in pratica a *Falso*.

45.2.2 $ true

`true`

`true` è un programma banale che restituisce semplicemente il valore 0, corrispondente in pratica a *Vero*.

45.2.3 $ 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. Se i file vengono espressi nella forma `/dev/fd/`*n*, allora si fa riferimento al descrittore di file *n*.

Test sul tipo di file

`-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.

`-h | -L` ⟨*file*⟩
Vero se il file esiste ed è un *link* 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 [`⟨*file-descriptor*⟩`]`
Vero se il descrittore di file indicato è aperto su un terminale. Se non viene indicato l'argomento, si intende **1** che corrisponde allo *standard output*.

Test sui permessi di accesso

`-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 suo bit *sticky* impostato.

`-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

Test sulle caratteristiche dei file

`-e` ⟨*file*⟩
Vero se il file esiste ed è di qualunque tipo.

`-s` ⟨*file*⟩
Vero se il file esiste e ha una dimensione maggiore di zero.

⟨*file1*⟩ `-nt` ⟨*file2*⟩
Vero se il primo file è più recente del secondo, in base alla data di modifica.

⟨*file1*⟩ `-ot` ⟨*file2*⟩
Vero se il primo file è più vecchio del secondo, in base alla data di modifica.

⟨*file1*⟩ `-et` ⟨*file2*⟩
Vero se i due file sono sullo stesso dispositivo e hanno lo stesso numero di *inode*. Ciò accade quando si creano degli *hard link*.

Test su stringhe

`-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.

Test di confronto numerico

⟨*argomento1*⟩ ⟨*operando*⟩ ⟨*argomento2*⟩

Il risultato dipende dall'operando utilizzato.

Operandi

`-eq` uguale a;
`-ne` diverso da;
`-lt` minore di;
`-le` minore o uguale a;
`-gt` maggiore di;
`-ge` maggiore o uguale a;

Gli argomenti da confrontare possono essere interi positivi, interi negativi, o la espressione speciale **`-l`** ⟨*stringa*⟩ che calcola la lunghezza della stringa stessa.

Operatori booleani

`!` ⟨*espressione*⟩

Vero se il risultato dell'espressione è *Falso*.

⟨*espressione1*⟩ `-a` ⟨*espressione2*⟩

Vero se entrambe le espressioni danno un risultato *Vero*.

⟨*espressione1*⟩ `-o` ⟨*espressione2*⟩

Vero se almeno una delle due espressioni da un risultato *Vero*.

Esempi

`$` ***`test 1 -gt 2 && echo "ok"`***[*Invio*]

```
ok
```

`$` ***`test -d /bin && echo "/bin è una directory"`***[*Invio*]

```
/bin è una directory
```

45.2.4 $ 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*.

Exit status

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 fare confusione 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.

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 una 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 parte di 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*⟩ ⟨*set-di-caratteri*⟩

Restituisce la prima posizione all'interno della stringa, a cui corrisponde uno dei caratteri che compone il set di caratteri.

`length` ⟨*stringa*⟩

Restituisce la lunghezza della stringa.

Esempi

$ ***export miavar=2***[*Invio*]

$ ***expr $miavar + 1***[*Invio*]

```
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*]

[1] Questo significa che l'espressione regolare deve corrispondere a partire dell'inizio della stringa.

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**.

45.3 Ridirezione

La ridirezione dei flussi di input e di output dei programmi viene svolto 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.

45.3.1 $ tee

tee [⟨*opzioni*⟩] [⟨*file*⟩...]

Copia lo *standard input* nello *standard output* e 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.

45.4 Contesto operativo

Alcuni programmi si occupano di mostrare o modificare il contesto in cui si opera.

45.4.1 $ stty

stty [⟨*opzioni*⟩]
stty [⟨*configurazione*⟩]

stty permette di modificare o di conoscere la configurazione del proprio terminale.
Vedere **info stty** oppure **stty(1)**.

45.4.2 $ 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.

45.4.3 $ 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.

Exit status

- **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**.

45.5 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.

45.5.1 $ 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.

Capitolo 46

Editing: creazione e modifica di file di testo

In informatica, con il termine *editing* ci si riferisce spesso alla elaborazione di file di testo semplici.

46.1 vi

Nei sistemi Unix, il programma 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 nulla 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.

Di fatto, per Linux, non esiste in circolazione una versione originale di **`vi`**, ma tanti cloni di questo, con potenzialità più o meno ampliate. Per questo, di solito **`vi`** sarà un *link* (simbolico o meno) al clone che si utilizza effettivamente.

> Il clone **`vi`** più importante è **`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 con i classici 640KB.

46.1.1 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 per *editing* a utilizzare tutto lo schermo per visualizzare le righe del testo su cui si opera. Da qui deriva lo stesso nome **`vi`**: *VIsual*.

A sua volta, **`vi`** è la derivazione di un programma precedente, **`ex`** di tipo a singola riga. Ma questa eredità è servita per incorporare i comandi tipici di **`ex`**.

46.1.2 Avvio di vi

`vi` [⟨*opzioni*⟩] [⟨*file*⟩...]

`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 46.1 in cui le righe dello schermo contrassegnate dal simbolo tilde (~) rappresentano lo spazio non utilizzato dal file.

Figura 46.1: Avvio del programma **`vi`**.

Normalmente, **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 *link* al clone **vi** che si ha a disposizione. Di norma, quando **vi** viene avviato con questo nome, si comporta come se gli fosse stata data l'opzione **-R**: sola lettura.

`ex [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

Come già accennato, **ex** è il programma da cui è derivato **vi**. Generalmente, nelle distribuzioni Linux, si trova un *link* (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.

46.1.3 Modalità di funzionamento

vi distingue diverse modalità di funzionamento, altrimenti definibili come stati o contesti. Quando si avvia, **vi** 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 dal programma **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.

46.1.4 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.

46.1.5 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**.

46.1.6 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. Nel caso del programma **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.

Nel caso di cloni più sofisticati, è normalmente concesso 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.

I comandi a disposizione per passare alla modalità di inserimento sono molti e non si limitano quindi ai due modi appena descritti. La tabella 46.1 ne elenca alcuni.

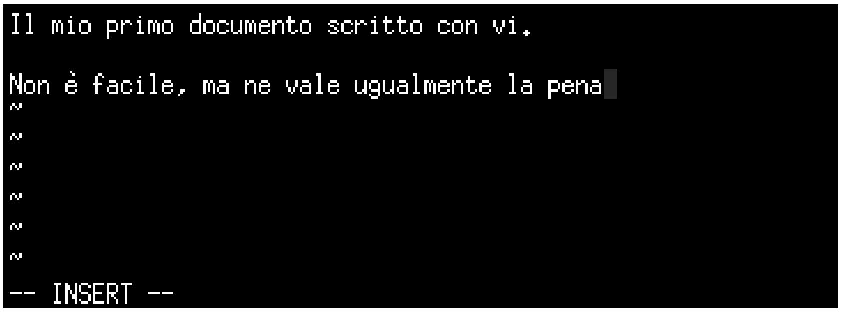

Figura 46.2: **vi** durante la fase di inserimento di testo.

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 46.1: Elenco dei comandi utilizzabili per passare alla modalità di inserimento.

46.1.7 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 determinata riga 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 46.2 ne elenca alcuni.

Esempi

5w

Sposta il cursore all'inizio della quinta parola successiva.

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.

46.1.8 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/o seguire da un comando di spostamento. Il comando di

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 46.2: Elenco dei comandi utilizzabili per la navigazione all'interno del testo.

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.
Per poter applicare questo concetto, è però 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.

46.1.9 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 (*newline*), 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 46.3: 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.

46.1.10 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 46.4: 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.

46.1.11 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 memorie di transito, o *buffer*, alle quali si possono accodare diverse parti di testo.

L'operazione con la quale si copia una porzione di testo in un *buffer* viene detta *yanking* (estrarre) e questo giustifica l'uso della lettera **y** nei comandi che compiono questa funzione.

I *buffer* per lo spostamento o la copia di testo possono essere 27: uno per ogni lettera dell'alfabeto e uno aggiuntivo senza nome.

Il modo più semplice di gestire questo meccanismo è quello di usare il *buffer* 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 *buffer*, 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 nel *buffer*.

Comando	Descrizione
yy	Copia la riga attiva nel *buffer*.
y⟨*mod*⟩	Copia nel *buffer* il testo fino all'estensione indicata dal modificatore.
dd	Trasferisce la riga attiva nel *buffer*.
d⟨*mod*⟩	Trasferisce nel *buffer* il testo fino all'indicazione dal modificatore.
p	Incolla prima della posizione del cursore.
P	Incolla dopo la posizione del cursore.

Tabella 46.5: Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso del *buffer* senza nome.

Esempi

5yy
Copia nel *buffer* 5 righe a partire da quella attiva.

yw
Copia nel *buffer* il testo che parte dalla posizione attiva fino all'inizio della prossima parola.

y$
Copia nel *buffer* il testo che parte dalla posizione attiva fino alla fine della riga.

3dd
Sposta nel *buffer* tre righe a partire da quella attiva.

2P
Incolla due copie del testo contenuto nel *buffer* a partire dalla posizione a sinistra del cursore.

46.1.12 Copia e spostamento con nome

Quando si vogliono utilizzare dei *buffer* specifici, cioè identificati attraverso le lettere dell'alfabeto, si procede nei modi già visti nel caso dell'uso del *buffer* 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 *buffer*: basta indicare il *buffer* attraverso una lettera maiuscola.

Comando	Descrizione
"*x*yy	Copia la riga attiva nel *buffer x*
"*x*y⟨*mod*⟩	Copia nel *buffer x* il testo fino all'indicazione dal modificatore.
"*x*dd	Trasferisce la riga attiva nel *buffer x*.
"*x*d⟨*mod*⟩	Trasferisce nel *buffer x* il testo fino all'indicazione dal modificatore.
"*x*p	Incolla il contenuto del *buffer x* prima del cursore.
"*x*P	Incolla il contenuto del *buffer x* dopo il cursore.

Tabella 46.6: Elenco dei comandi per le operazioni di copia e spostamento di testo che fanno uso dei *buffer* con nome.

Esempi

"*adw*
Sposta il testo nel *buffer* **a** a partire dalla posizione attiva fino all'inizio della prossima parola.

"*a5yy*
Copia 5 righe nel *buffer* **a** a partire dalla posizione attiva (inclusa).

"*A3yy*
Aggiunge 3 righe nel *buffer* **a** a partire dalla posizione attiva (inclusa).

"*a2P*
Incolla due copie del contenuto del *buffer* **a** a partire dalla posizione precedente a quella su cui si trova il cursore.

46.1.13 Ricerche

Per effettuare delle ricerche all'interno del documento aperto con **vi** si possono utilizzare le espressioni regolari (appendice A) 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 46.7: 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 46.8.

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 46.8: Le espressioni regolari di **vi**.

46.1.14 Ricerche e sostituzioni

La ricerca e sostituzione sistematica avviene attraverso un comando *colon* particolare. La sua sintassi è la seguente.

:⟨*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 espressamente confermata.

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.

46.1.15 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 precedentemente annullate 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 46.9: Annullamento di un comando.

46.1.16 Caricamento, salvataggio e conclusione

Il file o i file utilizzati per l'*editing* vengono normalmente aperti attraverso l'indicazione nella riga di comando, al momento dell'avvio di **vi**.

Il salvataggio di un file può essere fatto per mezzo del comando **:w** (*Write*) eventualmente seguito 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 la 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 sul 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 iniziare l'*editing*.
: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 46.10: I comandi per il caricamento dei file e il loro salvataggio.

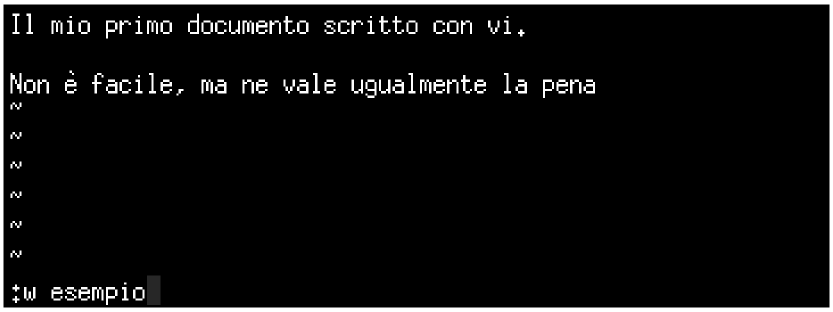

Figura 46.3: **vi** durante l'esecuzione del comando **:w**, prima della pressione conclusiva del tasto [*Invio*]

46.1.17 Variabili

vi ha ereditato dal programma **ex** delle variabili di configurazione. 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 46.11 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 46.11: 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.

46.1.18 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 possono dare fastidio alle operazioni di *editing*, questo comando permette di farli scomparire mostrando il testo effettivo del file sul quale si sta lavorando.

`:!`⟨*comando*⟩

Esegue il comando di *shell* indicato.

`:ab` ⟨*abbreviazione*⟩ ⟨*testo-da-sostituire*⟩

Permette di stabilire una abbreviazione che verrà sistematicamente sostituita con tutto quello che segue il comando. Se si usa **:ab** da solo, si ottiene un elenco delle abbreviazioni disponibili.

46.1.19 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
```

46.1.20 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 una ottima versione Dos in grado di funzionare molto bene anche con i vecchi computer dotati di poca memoria.

Quando si trasferiscono testi da un sistema Linux a Dos e viceversa si pone il problema del set di caratteri a disposizione: su 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 è possibile 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*⟩

46.2 joe

Il programma **joe** è solo uno tra i tanti adatti per l'*editing* 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.

46.2.1 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 *home*;
- in `/usr/lib/joe/`.

La distribuzione normale di **joe** fornisce diversi file di configurazione, che vengono utilizzati a seconda del nome utilizzato per avviare il programma (per esempio attraverso un *link* simbolico).

Le indicazioni sul funzionamento di **joe** 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 *home* e rinominarlo, in modo da ottenere `~/.joerc`.

46.2.2 Avvio di joe

`joe [`⟨*opzioni-globali*⟩`] [[`⟨*opzioni-particolari*⟩`]`⟨*file*⟩`]...`

joe può essere avviato o meno con l'indicazione di uno o più file sul quale 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 *link*, oppure modificando direttamente il nome dell'eseguibile. A seconda del nome utilizzato, 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`

[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.

La conclusione normale dell'attività con un file coincide con il salvataggio silenzioso 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 una operazione di taglia/copia-incolla.

`-lightoff`

Con questa opzione si fa in modo che al termine di una 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 46.4. 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.

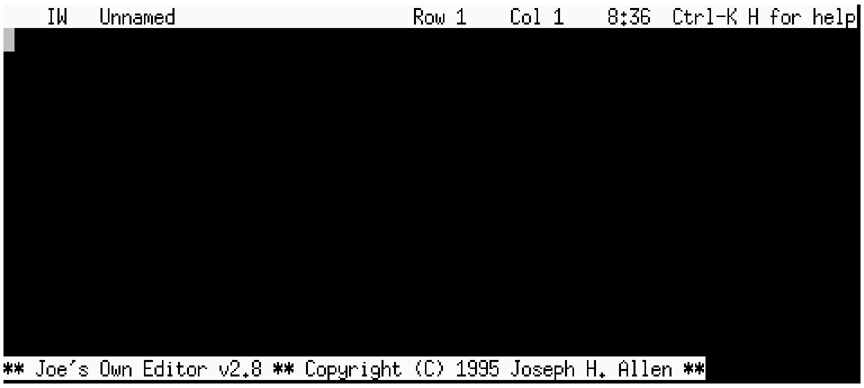

Figura 46.4: Avvio di **`joe`**.

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 (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 una 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.
La riga inferiore dello schermo viene normalmente utilizzata per l'inserimento di informazioni che completano un comando, come il nome di un file da salvare o caricare. Quando questa parte non viene utilizzata, la finestra contenente il testo del file su cui si lavora si riappropria di quello spazio.

46.2.3 Comandi

I comandi che si possono impartire a **`joe`** sono generalmente in forma di sequenza di tasti iniziata con 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 in linea, 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 una ulteriore combinazione [*Ctrl+c*].

46.2.4 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 46.5 mostra un esempio di questa guida.

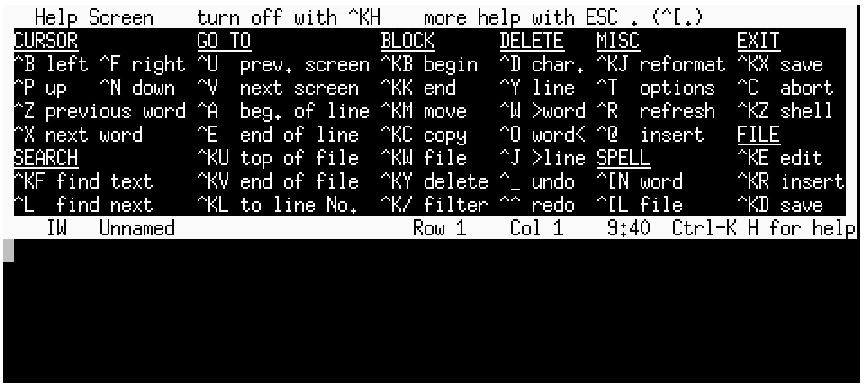

Figura 46.5: La guida interna di **`joe`**.

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).

46.2.5 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 46.12 riassume brevemente le possibilità esistenti.

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 una 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+ˆ	Ctrl+ì	*Redo*: riesegue l'ultimo comando, compresa la digitazione.
Ctrl+r		Ripristina il testo sullo schermo.

Tabella 46.12: Riepilogo delle funzionalità di inserimento e modifica di testo con **`joe`**.

È importante ricordare che i caratteri che utilizzano più di 7 bit non vengono visualizzati se non si utilizza l'opzione **`-asis`** o se non si configura il file `~/.joerc`.

46.2.5.1 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 il set 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 46.13 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 46.13: Lettere accentate e simboli corrispondenti nei terminali a 7 bit, quando la codifica a 8 bit di partenza è ISO 8859-1.

46.2.6 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 46.14 elenca i tasti e le combinazioni utili per la navigazione del cursore nel testo.

46.2.7 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.

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 46.15 elenca i comandi utili per la gestione di file e finestre.

La figura 46.6 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;

Tasto o combinazione	Risultato
tasti freccia	Spostano il cursore di una posizione nella direzione particolare 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 46.14: Comandi utili per la navigazione con **`joe`** quando è configurato nel modo predefinito.

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 46.15: Comandi per la gestione di file e finestre.

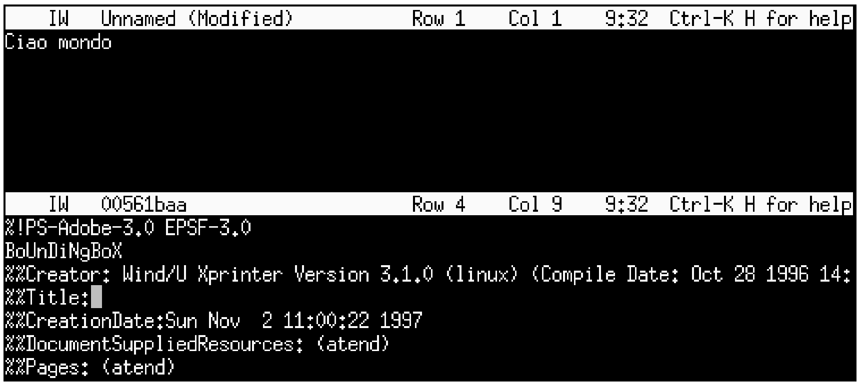

Figura 46.6: Due finestre che si dividono lo spazio dello schermo.

- 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.

46.2.8 Taglia, copia, incolla e blocchi di testo

Una della caratteristiche più importanti di **`joe`** è quella di poter intervenire attraverso le consuete operazioni di 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.
La tabella 46.16 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 46.16: Comandi per la gestione dei blocchi di testo.

46.2.9 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 46.17 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 46.17: Comandi per la formattazione del testo.

46.2.10 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 *newline*.
[*Ctrl+l*] consente di ripetere una ricerca.

46.2.11 Completamento e storico

Quando il programma propone un *prompt*, ovvero 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 una informazione inserita in precedenza nella stessa situazione. Si ottiene cioè lo scorrimento dello storico dei dati inseriti.

46.2.12 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)`**.

Capitolo 47

File manager

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, o **`mc`**.

47.1 Midnight Commander

Midnight Commander, o **`mc`**, è un ottimo *file manager* per i terminali a carattere. Il nome richiama chiaramente la sua origine: si tratta di un programma molto simile al noto Norton Commander.

Non si può dire che si tratti di un programma essenziale, dal momento che gli strumenti a disposizione con Linux, e in generale con Unix, sono più che sufficienti. Ma uno strumento del genere, può facilitare di molto l'attività dell'utente normale o dell'amministratore del sistema.

Lo scopo principale di questo programma è quello di permettere la visione e la gestione simultanea di due directory di lavoro.

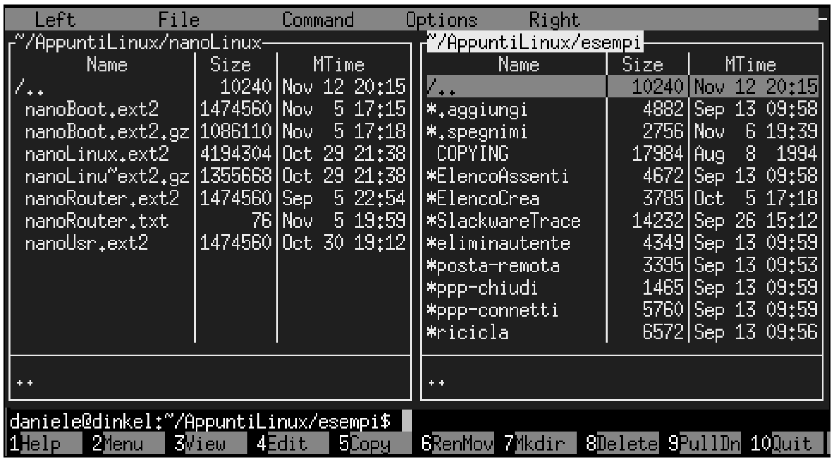

Figura 47.1: Midnight Commander.

A questo proposito, si può suddividere lo schermo gestito da **`mc`** in quattro parti:

1. la superficie più grande è utilizzata, nella parte centrale, 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 eventualmente restare 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*].

47.1.1 Avvio di Midnight Commander

`mc` [⟨*opzioni*⟩] [⟨*directory1*⟩ [⟨*directory2*⟩]]

`mc` viene normalmente avviato 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.

47.1.2 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 una 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.

47.1.3 Tastiera

`mc` potrebbe essere utilizzato anche in situazioni non ottimali come lo è invece una console di 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 permettono di utilizzarlo, in sua sostituzione si può utilizzare la sequenza [*Esc*] [*tasto*]. Si tratta quindi di premere [*Esc*], rilasciarlo e premere successivamente il tasto seguente.[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 una 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 evidenziato o predefinito.[2]

Il funzionamento di altri tasti e combinazioni di tasti viene riassunto nella tabella 47.1. Quello che è importante tenere presente è che non sono sempre disponibili tutti in ogni situazione.

47.1.4 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 relativa funzione.

47.1.5 Configurazione dei pannelli

`mc` 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.

- **`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;

[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 una 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
Ctrl+l		Ridisegna l'immagine dello schermo o della finestra.
Ctrl+r		Rilegge il contenuto delle due directory aggiornando i pannelli.
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 caratteri jolly o espressioni regolari.
\		Deseleziona un gruppo di file, attraverso caratteri jolly o espressioni regolari.
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 che venga interpretato diversamente.
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 *editing*.
freccia destra	Ctrl+f	Sposta il cursore a destra in una qualunque riga di *editing*.
Backspace	Ctrl+h	Cancella il carattere precedente.
Canc	Ctrl+d	Cancella il carattere successivo.

Tabella 47.1: Alcuni tasti e combinazioni di tasti utili per il controllo di **`mc`**.

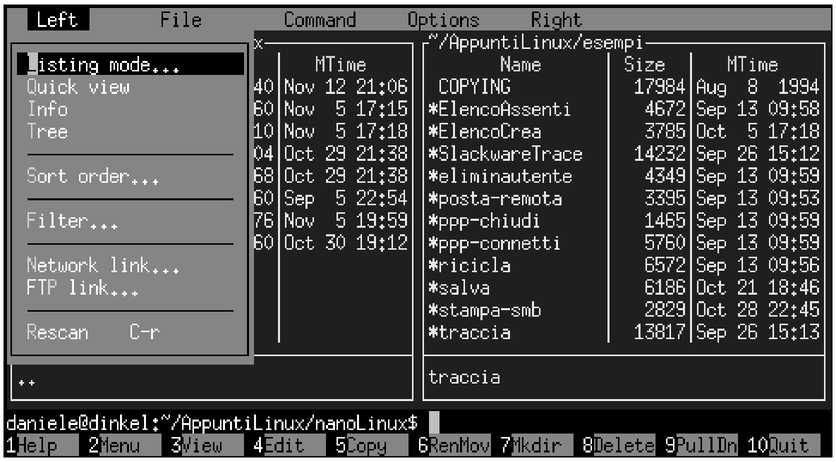

Figura 47.2: Il menu **`Left`** di **`mc`**.

- **`Info`**
  mostra le informazioni complete del file selezionato nell'altro pannello;
- **`Tree`**
  mostra la struttura del *filesystem*;
- **`Sort order`**
  permette di definire l'ordinamento;
- **`Filter`**
  permette di definire un filtro, espresso attraverso caratteri jolly o espressioni regolari, dei nomi degli elementi contenuti nel pannello.

A livello globale, è possibile definire la configurazione di **`mc`** 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`.

47.1.5.1 Listing mode

La funzione **`Listing mode`** è molto utile per configurare l'aspetto dell'elenco di file del pannello sinistro e destro. La figura 47.3 mostra un esempio della maschera che viene ottenuta attraverso la sua selezione.

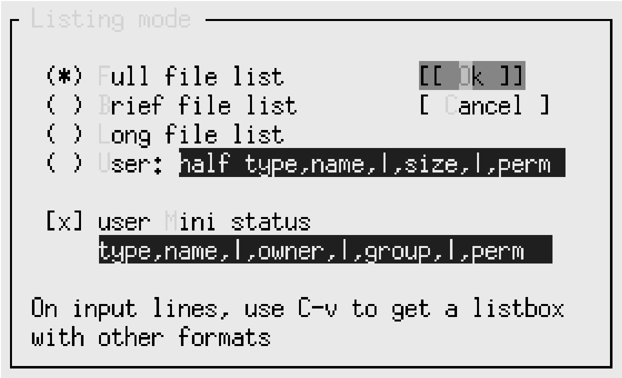

Figura 47.3: La maschera **`Listing mode`**.

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:`**. 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:

- **`|`**
  la barra verticale 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 da cursore. Gli elementi da visualizzare sono indicati attraverso una sintassi analoga a quanto appena descritto sopra.

47.1.6 Opzioni

Il menu opzioni (**`Options`**) è quello che permette di configurare il funzionamento di **`mc`** in modo globale, indipendentemente dalle particolarità di una particolare finestra. 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 **`mc`**, sia per ciò che riguarda questo menu, che per quanto accessibile attraverso i menu **`Left`** e **`Right`**, viene memorizzata nel file `.mc.ini`, nella directory personale dell'utente a cui si riferisce. Per ottenere questo, occorre però che tale configurazione sia salvata attraverso la funzione **`Save setup`**, alla fine del menu delle opzioni.

47.1.6.1 Configurazione

La funzione **`Configuration`** permette di modificare alcuni comportamenti importanti di **`mc`**. Per esempio è possibile includere o escludere i file "nascosti" e quelli di *backup*. Inoltre, le versioni recenti di **`mc`** 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 47.4 mostra un esempio della maschera di configurazione generale delle opzioni.

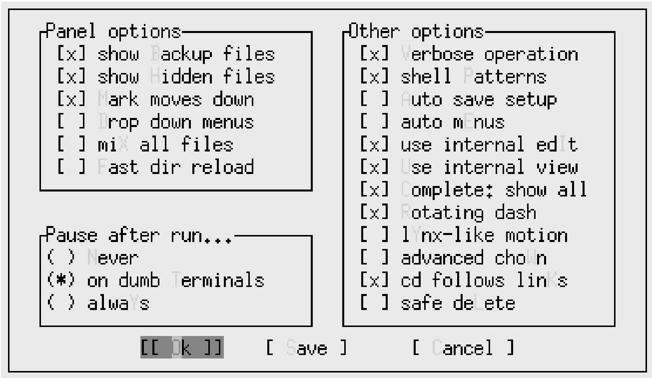

Figura 47.4: Menu opzioni: configurazione.

47.1.7 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.

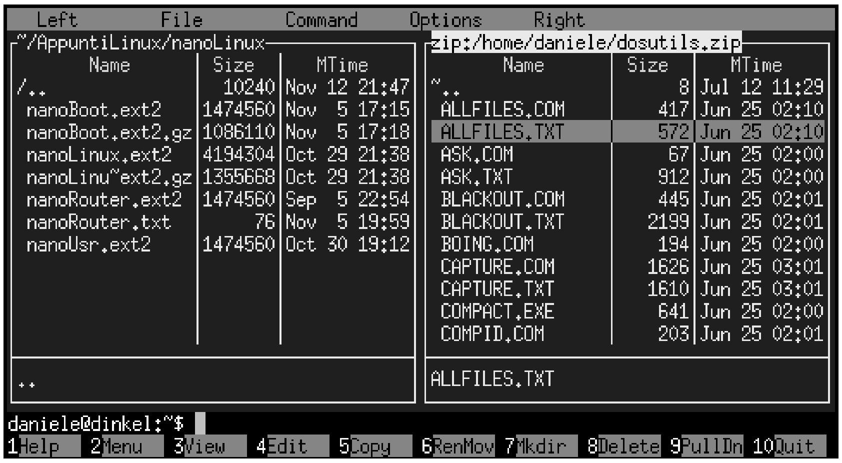

Figura 47.5: **`mc`** permette di accedere facilmente anche al contenuto di archivi compressi.

Nello stesso modo è possibile accedere a un servizio FTP, come se si trattasse di un normale *filesystem*.

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.

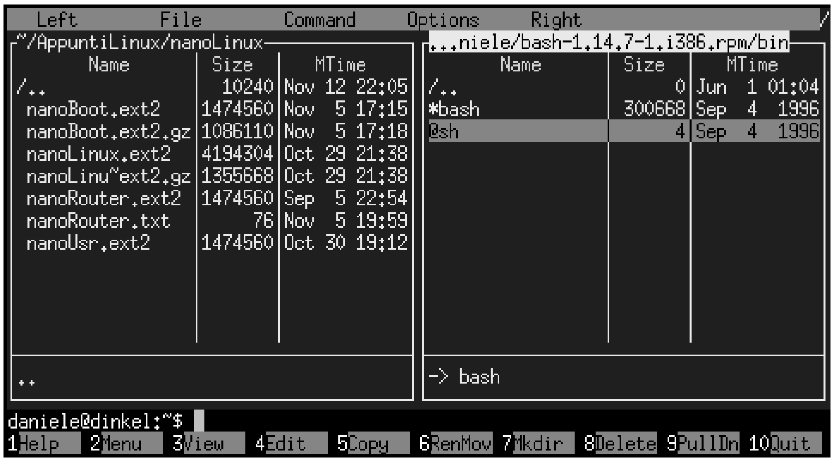

Figura 47.6: Gli archivi RPM vengono gestiti come normali archivi compressi.

`ftp://[`⟨*utente*⟩`@]computer[`⟨*directory-remota*⟩`]`

Per esempio, per accedere alla directory `/pub/` dell'FTP **`dinkel.brot.dg`**, si può utilizzare il comando seguente.

`$` ***`cd ftp://dinkel.brot.dg/pub`***[*Invio*]

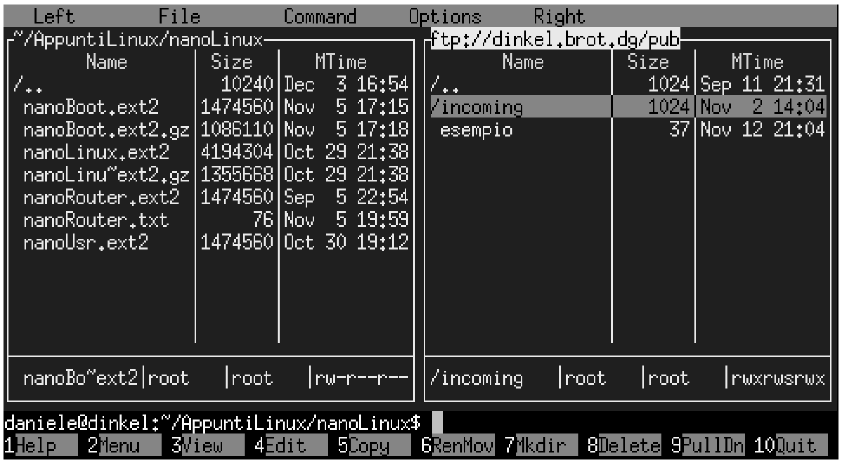

Figura 47.7: **`mc`** incorpora le funzionalità di un *client* FTP.

Per **`mc`**, nel concetto di *filesystem* virtuale, rientra anche l'analisi di un *filesystem* Ext2 allo scopo di recuperare i file cancellati. A questa funzione si accede attraverso il menu **`command`** e la voce **`Undelete files`**. La figura 47.8 mostra l'elenco di **`inode`** recuperabili nella quarta partizione del secondo disco.

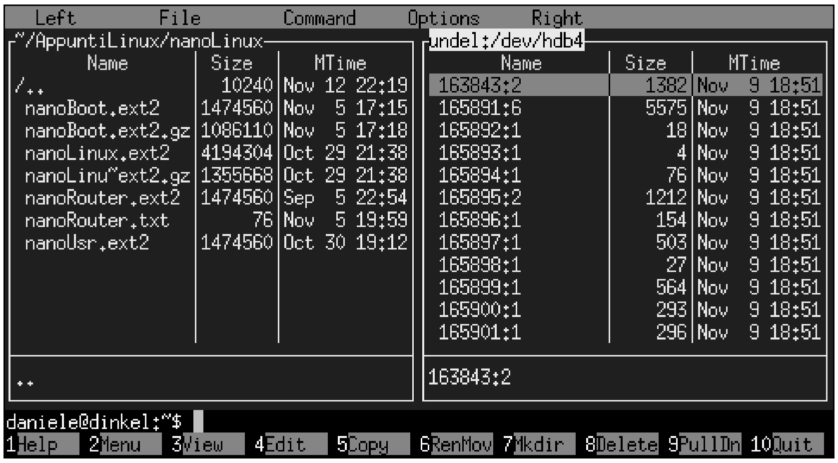

Figura 47.8: **mc** permette di recuperare facilmente i file cancellati in un *filesystem* Ext2.

Parte XII

Stampare

Appunti Linux

Via Turati, 15 - I-31100 Treviso - daniele@calion.com

Questo documento può essere ottenuto nella sua versione integrale da uno dei siti seguenti.

<ftp://ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://pluto.psy.unipd.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://ftp.linux.it/pub/mirrors/pluto/ildp/AppuntiLinux/>
<ftp://linux.keycomm.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://ftp.unina.it/pub/linux/pluto/ildp/AppuntiLinux/>
<ftp://ftp.media.dsi.unimi.it/pub/linux/ftp.pluto.linux.it/pub/pluto/ildp/AppuntiLinux/>
<ftp://linux.calion.com/pub/pluto/ildp/AppuntiLinux/>

La versione HTML di questo documento è disponibile per la consultazione diretta, attraverso la rete, nei siti seguenti.

<http://www.pluto.linux.it/ildp/AppuntiLinux/>
<http://ildp.psy.unipd.it/AppuntiLinux/>
<http://www.panservice.it/mirror/ILDP/AppuntiLinux/>
<http://linux.calion.com/pluto/ildp/AppuntiLinux/>

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 normalmente utilizzato nelle distribuzioni di 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 molto più avanti, ma di sicuro l'esigenza di stampare si avverte molto presto, e per questo viene anticipato questo argomento.

48.1 Kernel

Per poter utilizzare la stampante, occorre avere compilato il *kernel* inserendo la gestione della stampa. All'interno del gruppo **`Character devices`** occorre rispondere affermativamente alla domanda seguente se si utilizza la porta parallela.

- CONFIG_PRINTER - Paralel printer support **`Y`**

Nel caso in cui si debba condividere la porta parallela per la stampa e per una connessione PLIP con un altro computer, occorre compilare il supporto alla stampa come modulo, **`M`**, per poterlo disattivare quando si vuole fare una connessione PLIP.

48.2 Dispositivi di stampa

I dispositivi di stampa sono le porte parallele e/o seriali utilizzate per la connessione con le stampanti. Il nome assegnato alle porte parallele destinate alla stampa, dipende dell'indirizzo I/O di queste. Segue un elenco degli indirizzi possibili per le porte parallele.

- 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 `/dev/lp1`. Ma questo ragionamento non vale più per i *kernel* 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 alla seguente.

```
lp1 at 0x0378, (polling)
```

48.2.1 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. Una azione del genere corrisponde a quello che in ambiente Dos si poteva fare nel modo seguente.

`C:>` ***`copy /b stampa.prn lpt1`***

48.3 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 computer remoti.

[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 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 Linux.

48.3.1 Coda di stampa

La coda di stampa è gestita schematicamente attraverso i componenti seguenti:

- una directory utilizzata come *spool*, cioè come deposito delle stampe da eseguire: `/var/spool/lpd/`⟨*stampante*⟩;
- un file di configurazione contenente le informazioni sui tipi di stampante a disposizione, le loro caratteristiche e le posizioni delle relative directory di *spool*: `/etc/printcap`;
- un programma (*daemon*) che si occupa di gestire la stampa di quanto contenuto nella directory di *spool*: **`lpd`**;
- un programma per inserire nuove stampe nella coda, cioè nella directory di *spool* della stampante che dovrà stamparle: **`lpr`**.

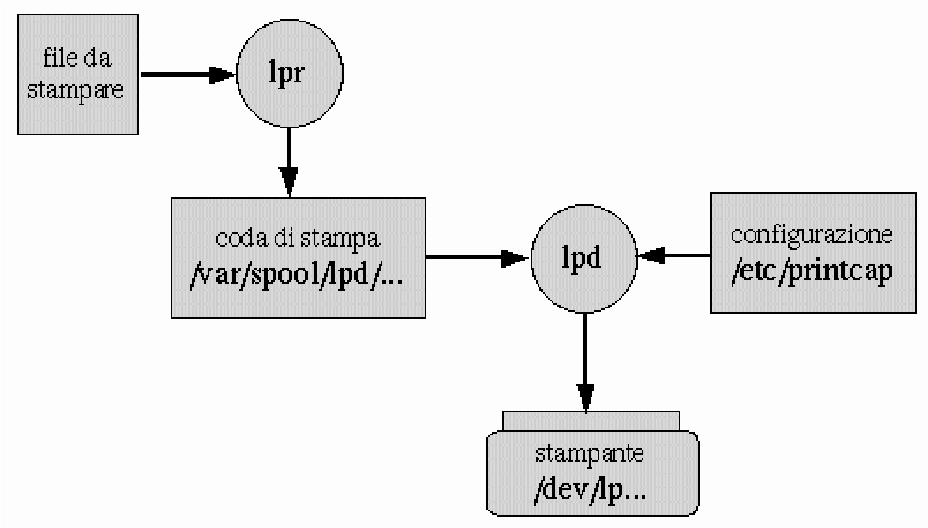

Figura 48.1: Schema semplificato della gestione della coda di stampa.

Le stampe che si accodano all'interno di una directory di *spool* ricevono un numero di *job* che serve per distinguerle. Le stampe accodate sono quindi dei processi di stampa accodati.
Il *daemon* **`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.

48.3.2 Server di stampa

La prima copia del *daemon* **`lpd`** si occupa anche di stare in ascolto della porta **`printer`** in modo da poter ricevere richieste di stampa da altri computer collegati in rete. L'utilizzo del servizio non è concesso a tutti indiscriminatamente, è necessario che il nome, o l'indirizzo IP del computer che ne fa richiesta sia presente nel file `/etc/hosts.equiv` (66.1.1), 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.

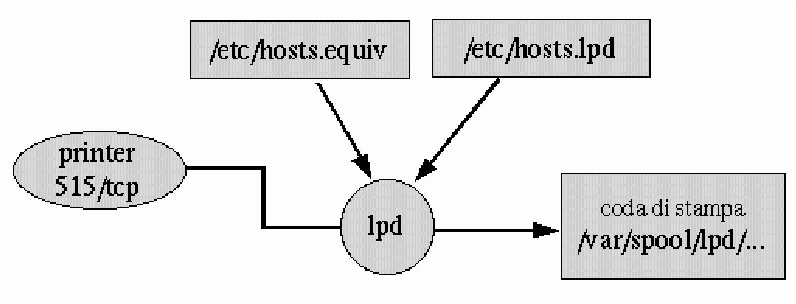

Figura 48.2: Lo stesso *daemon* **`lpd`** è il responsabile dell'inserimento in coda dei file da stampare, provenienti da computer remoti.

Il *daemon* **`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.

48.3.3 # lpd

`lpd [`⟨*opzioni*⟩`]`

È il *daemon* che si occupa di gestire la coda di stampa, ovvero lo *spool* 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 normalmente utilizzata 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`

  I computer elencati all'interno di questo file sono ammessi a utilizzare il servizio di stampa locale.
- `/etc/hosts.lpd`

  I computer elencati all'interno di questo file sono ammessi a utilizzare il servizio di stampa locale.

48.3.4 /etc/hosts.lpd

Questo file serve per elencare i nomi dei computer cui è consentito collegarsi e utilizzare le stampanti del computer locale.

Anche i computer elencati nel file `/etc/hosts.equiv` sono ammessi a utilizzare le stampanti locali. Ma questo file riguarda piuttosto il concetto generale della equivalenza, e viene utilizzato soprattutto per consentire l'accesso a programmi come **`rsh`** (66.3.2).

In generale quindi, se quello che si vuole è solo concedere l'utilizzo della stampante, è opportuno inserire i nomi di quei computer 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
```

48.3.5 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`**.

48.3.6 $ lpr

`lpr [`⟨*opzioni*⟩`] [`⟨*file*⟩`... ]`

Si occupa di accodare la stampa dei file indicati come argomento oppure dello *standard input*.

Alcune opzioni

`-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 allo *spool*, elimina il file. Funziona in combinazione con il parametro **`-s`**.

`-s`

Invece di copiare il file da stampare nella directory di *spool*, si limita a creare un *link* simbolico. Se si utilizza questa opzione, il file originale non può essere rimosso o modificato fino a che la stampa non è terminata.

`-#`*n*

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*.

48.3.7 $ lpq

`lpq [`⟨*opzioni*⟩`] [`⟨*numero-di-job*⟩`... ] [`⟨*utente*⟩`... ]`

`lpq` esamina lo *spool* 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 *job* di stampa.

48.3.8 $ lprm

`lprm [`⟨*opzioni*⟩`] [`⟨*utente*⟩`... ]`

Permette di rimuovere uno o più *job* di stampa precedentemente accodati. 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 *job* 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-di-job*⟩`...`

Se tra gli argomenti vengono indicati uno o più numeri, questi si intendono riferiti ai processi di stampa (*job*) che si vogliono eliminare.

48.3.9 # 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 *daemon* 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.

Comandi

`lpc` può essere eseguito anche da un utente normale, ma in tal caso potranno essere eseguite solo alcune funzioni. Segue l'elenco dei 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 *daemon* di stampa. Ciò può essere utile quando per qualche motivo inspiegabile il *daemon* ha interrotto la sua attività lasciando dei processi di stampa nella coda.

`status { all |` ⟨*stampante*⟩`}`

Visualizza lo stato della stampante locale indicata.

I comandi restanti sono riservati all'uso da parte dell'utente *root*.

`abort { all |` ⟨*stampante*⟩`}`

Termina l'esecuzione del *daemon* attivo che si occupa della stampa nel computer locale e quindi disabilita la stampa, prevenendo l'avvio di altri *daemon* da parte di **`lpr`**. Quando verrà riavviata la stampa, verrà ripresa la stampa del processo di stampa attivo nel momento dell'interruzione.

`clean { all |` ⟨*stampante*⟩`}`

Elimina i file temporanei, di dati e di controllo che non possono essere stampati, cioè quelli che non fanno parte di un *job* 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 *daemon* necessari.

`stop { all |` ⟨*stampante*⟩`}`

Interrompe il *daemon* di *spool* 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-di-job*⟩`... ] [`⟨*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 *daemon* di stampa se necessario, annullando l'effetto di un precedente comando **`down`**.

48.3.10 /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. Si tratta di un *database* nel quale ogni *record* contiene le informazioni per una stampante. I campi di questi *record* sono separati dal simbolo due punti (`:`) e possono essere spezzati su più righe utilizzando la barra inclinata rovescia (\) seguita immediatamente da *newline*. All'interno di questo file si possono trovare le indicazioni di diverse stampanti che possono fare capo a una 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 del computer 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 eventuale valore.

> La compilazione di questo file è molto delicata: in particolare, occorre fare bene attenzione a non lasciare spazi di qualunque tipo dopo i simboli \.

Quando si modifica il file `/etc/printcap`, occorre ricordare di inviare un segnale **`SIGHUP`** al *daemon* **`lpd`**, oppure lo si deve riavviare. Diversamente, le modifiche non avranno effetto.

`kill -HUP` ⟨*pid-di-lpd*⟩

Vedere **`printcap(5)`**.

48.3.10.1 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 un segno di **=** seguito dalla stringa racchiusa tra doppi apici.

- Campi numerici

  Dopo il nome del campo, viene aggiunto un segno di **#** 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 computer remoto da utilizzare per la stampa.
`rp`
Remote Printer. Il nome di una stampante remota, cioè il nome utilizzato nel computer remoto indicato nel campo **`rm`**.
`sd`
Spool Directory. Il nome della directory di *spool* 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.

48.3.10.2 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*.

48.3.10.3 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 di *spool* 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 del computer 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 una 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`** del computer remoto non ha un filtro.

48.3.11 /var/spool/lpd/*/

Le directory di *spool* sono definite attraverso il campo **`sd`** dei *record* contenuti in `/etc/printcap`. Normalmente, nelle distribuzioni 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 e/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 (*job*) 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 semplicemente eliminare 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` in ogni directory di *spool* da controllare. Questo file deve contenere un numero corrispondente alla quantità di blocchi che si vuole restino comunque liberi.

48.4 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.

48.4.1 File di testo

Negli ambienti Unix i file di testo (o file ASCII) seguono la convenzione della terminazione di riga attraverso il codice ASCII <*LF*>.
Con il sistema operativo Dos è stato introdotto un codice di fine riga differente, corrispondente a <*CR*><*LF*>. La maggior parte delle stampanti in circolazione è adatta a quest'ultimo tipo di terminazione 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 la terminazione 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 *newline* in modo che comprendano sia <*CR*> che <*LF*>.
Il tipico programma in grado di eseguire questa conversione è **unix2dos**, il quale si comporta come programma filtro, e per questo è perfettamente adatto all'utilizzo come filtro di stampa. 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***

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 interna: **unix2dos(1)**.

48.4.2 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.

48.5 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 un scopo particolare. Di fatto, è preferibile limitarsi a utilizzarne uno solo, e precisamente quello del campo **if**, o *Input Filter*.

48.5.1 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 il computer dal quale parte la stampa;

- ⟨*file-di-accounting*⟩

  rappresenta il nome del file eventualmente indicato 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 *log*: il nome dell'utente e il nome del computer.

48.5.2 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 alcun *flag*. Un'altra cosa molto importante è la directory corrente: corrisponde sempre alla directory di *spool*.

48.5.3 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.

[2] Se si fanno modifiche al file `/etc/printcap` non bisogna poi dimenticare di inviare un segnale di aggancio al *daemon* **lpd** per fare in modo che venga riletto: **kill -s SIGHUP** ⟨*pid-di-lpd*⟩.

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 ottiene solo una segnalazione di errore: **`unix2dos`** non è 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 e 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.

48.5.4 Filtri PostScript

Tutti i filtri di stampa in grado di convertire file PostScript in qualcosa di stampabile senza 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=- -
```

48.5.5 Problemi

In passato è capitato che una particolare versione del sistema di stampa BSD per Linux avesse un difetto che non le permetteva di utilizzare il flusso di dati proveniente dal filtro di stampa.
In questo caso, si può utilizzare un trucco: il filtro di stampa riceve i dati dallo *standard input* nel modo solito, li trasforma e, invece di emetterli attraverso lo *standard output* li invia a un'altra coda di stampa.
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
```

48.5.6 Filtri sofisticati

Non è necessario complicarsi troppo la vita. Spesso la distribuzione 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 una 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 di *spool* e i filtri necessari.
Infine, è il caso di ricordare le **`psutils`** che sono in grado di rielaborare file PostScript, cosa utile per esempio quando su un solo foglio si vogliono stampare più pagine ridotte.

48.6 Accessori di stampa

Alcuni programmi sono utili come strumenti per la diagnosi o la configurazione della stampa.

48.6.1 # lptest

`lptest [`⟨*larghezza*⟩`] [`⟨*altezza*⟩`]`

Emette attraverso lo *standard output* una serie di caratteri utilizzabile per un test 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 un test di stampa utile solo per le stampanti a impatto come quelle ad aghi, a catena, a margherita...

48.7 Stampare attraverso X

Quando si inizia a utilizzare X Window System, 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`**.

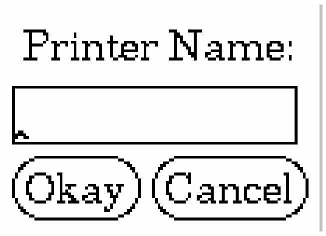

Figura 48.3: Questo è un esempio di un programma che è in grado di stampare solo attraverso **`lpr`**. Se non viene indicato alcun nome di stampante, si fa riferimento a **`lp`**, o comunque alla stampante predefinita.

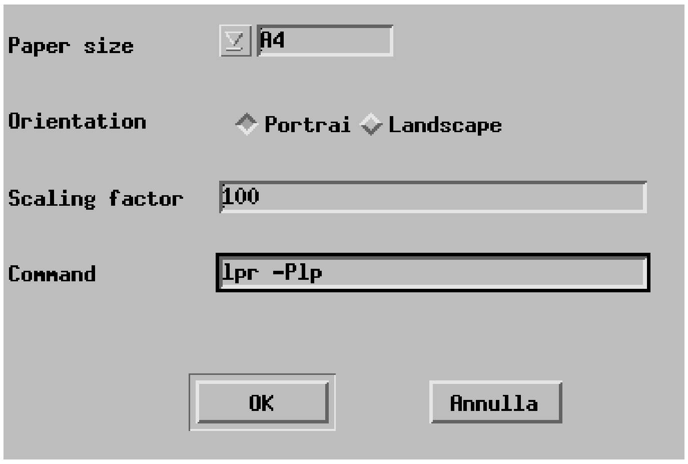

Figura 48.4: Questo è un esempio di un programma normale che permette l'indicazione di una riga di comando completa (o quasi). Non deve essere inserito il nome del file da stampare che di norma viene fornito attraverso lo *standard input*.

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.
PostScript is a trademark of Adobe Systems Incorporated.

49.1 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 i *font*, 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 b(Daniele)n(.G)o(iacomini@pluto)m(.linux.it)300
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
```

49.1.1 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.

49.2 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).

49.2.1 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 nella tabella 49.1.

`-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 49.2.

`-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 invece emesso attraverso lo *standard output*.

`-`

Se al posto del nome del file PostScript da convertire si indica un semplice trattino (**-**) isolato, i nomi dei file da convertire vengono letti dallo *standard input*.

Esempi

`$` ***`gs -dNOPAUSE -q -sDEVICE=cdjmono -sOutputFile=- esempio.ps < /dev/null | lpr`***

Invia al sistema di *spool* delle stampe (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.

49.3 Anteprima di stampa

Nello stesso modo in cui GhostScript viene utilizzato per convertire file PostScript in modo da poterli stampare anche attraverso stampanti normali, così è possibile ottenere una conversione in un formato grafico che possa essere mostrato attraverso lo schermo, solitamente all'interno del sistema grafico.

Alcuni strumenti grafici specifici, si occupano di guidare l'utente all'utilizzo di GhostScript in modo da ottenere una anteprima di stampa su schermo.

49.3.1 Ghostview

È un programma che facilita la visualizzazione di file PostScript all'interno dell'ambiente grafico X Window System attraverso una gestione automatizzata e semplificata di GhostScript.

appledmp	Apple Dot Matrix Printer (Imagewriter)
bj10e	Canon BubbleJet BJ10e
bj200	Canon BubbleJet BJ200
lbp8	Canon LBP-8II (laser)
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)
cdj500	HP DeskJet 500C (**`cdjcolor`**)
cdj550	HP DeskJet 550C
djet500c	HP DeskJet 500C
deskjet	HP DeskJet e HP DeskJet Plus
laserjet	HP LaserJet
ljet2p	HP LaserJet IId/IIp/III* (con compressione TIFF)
ljet3	HP LaserJet III* (con compressione Delta Row)
ljet4	HP LaserJet 4 (600 dpi)
ljet4l	HP LaserJet 4L (300 dpi)
ljetplus	HP LaserJet Plus
paintjet	HP PaintJet (colore)
pj	HP PaintJet XL
pjxl	HP PaintJet XL (colore)
pjxl300	HP PaintJet XL300 (colore)
declj250	DEC LJ250
la50	DEC LA50
la75	DEC LA75
ln03	DEC LN03
lj250	DEC LJ250 (colore)
dfaxhigh	DigiFAX (alta risoluzione)
dfaxlow	DigiFAX (risoluzione normale)
epson	Epson-compatibile (9 o 24 aghi)
eps9high	Epson-compatibile (9 aghi ad alta risoluzione)
epsonc	Epson LQ-2550 e Fujitsu 3400/2400/1200 a colori
escp2	Epson ESC/P2 (compresa Stylus 800)
ibmpro	IBM Proprinter (9 aghi)
jetp3852	IBM Jetprinter (a getto, colore, modello #3852)
m8510	C.Itoh M8510
necp6	NEC P6/P6+/P60 (360x360 dpi)
nwp533	Sony Microsystems NWP533 (laser)
oki182	Okidata MicroLine 182
r4081	Ricoh 4081 (laser)
sparc	SPARCprinter
t4693d2	Tektronix 4693d (colore, 2 bit)
t4693d4	Tektronix 4693d (colore, 4 bit)
t4693d8	Tektronix 4693d (colore, 8 bit)
tek4696	Tektronix 4695/4696 (plotter a getto d'inchiostro)

Tabella 49.1: Stampanti e unità utilizzabili con GhostScript.

formato	larghezza 1/72"	altezza 1/72"	larghezza pollici	altezza pollici
letter	612	792	8.50	11.00
note	540	720	7.50	10.00
legal	612	1008	8.50	14.00
a0	2380	3368	33.06	46.78
a1	1684	2380	23.39	33.06
a2	1190	1684	16.53	23.39
a3	842	1190	11.69	16.53
a4	595	842	8.26	11.69
a5	421	595	5.85	8.26
a6	297	421	4.12	5.85
a7	210	297	2.92	4.12
a8	148	210	2.06	2.92
a9	105	148	1.46	2.06
a10	74	105	1.03	1.46
b0	2836	4008	39.39	55.67
b1	2004	2836	27.83	39.39
b2	1418	2004	19.69	27.83
b3	1002	1418	13.92	19.69
b4	709	1002	9.85	13.92
b5	501	709	6.96	9.85
archE	2592	3456	36.00	48.00
archD	1728	2592	24.00	36.00
archC	1296	1728	18.00	24.00
archB	864	1296	12.00	18.00
archA	648	864	9.00	12.00
flsa	612	936	8.50	13.00
flse	612	936	8.50	13.00
halfletter	396	612	5.50	8.50
11x17	792	1224	11.00	17.00
ledger	1224	792	17.00	11.00

Tabella 49.2: Formati di stampa di ghostscript.

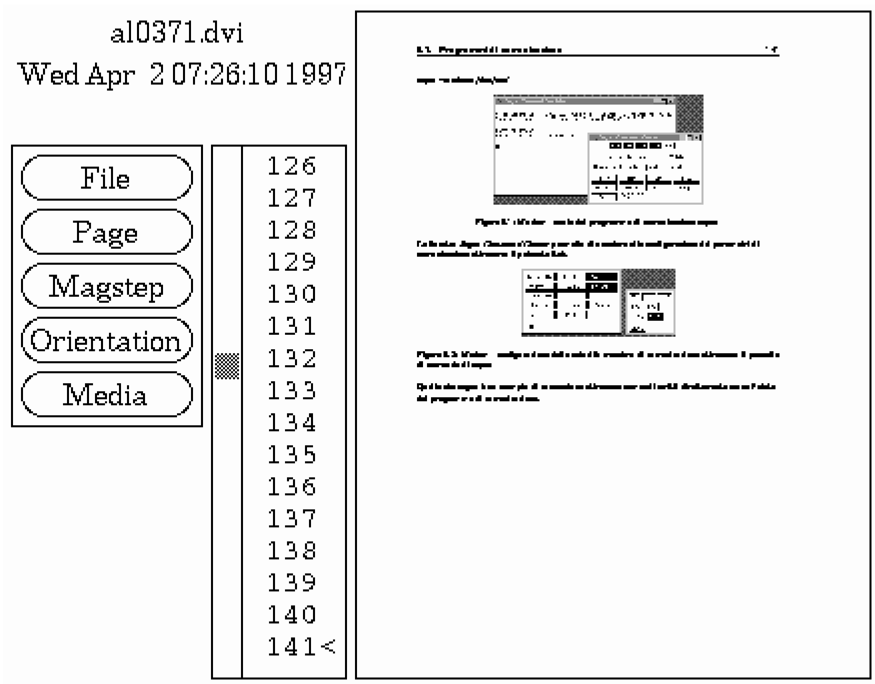

Figura 49.1: Il programma **ghostview**.

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*⟩`]`

`ghostview` viene utilizzato generalmente senza alcun argomento, eventualmente può essere fornito il nome del file PostScript che si vuole visualizzare.

49.3.1.1 Uso della tastiera

La libreria grafica utilizzata per questo programma, non è molto comoda da utilizzare con il solo mouse. Per questo, è conveniente conoscere alcuni comandi che si possono dare utilizzando 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).

49.3.1.2 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.

49.3.1.3 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 la combinazione di tasti equivalente.

- **`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 evidenziate.

- **`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.

49.3.2 gv

`gv` è un programma derivato da Ghostview con lo stesso scopo, ma con una interfaccia grafica più comoda e intuitiva.

`gv [`⟨*file*⟩`] [`⟨*opzioni*⟩`]`

`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)`**.

49.4 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 una 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: **`a3`**, **`a4`**, **`a5`**, **`b5`**, **`letter`**, **`legal`**, **`tabloid`**, **`statement`**, **`executive`**, **`folio`**, **`quarto`** e **`10x14`**.

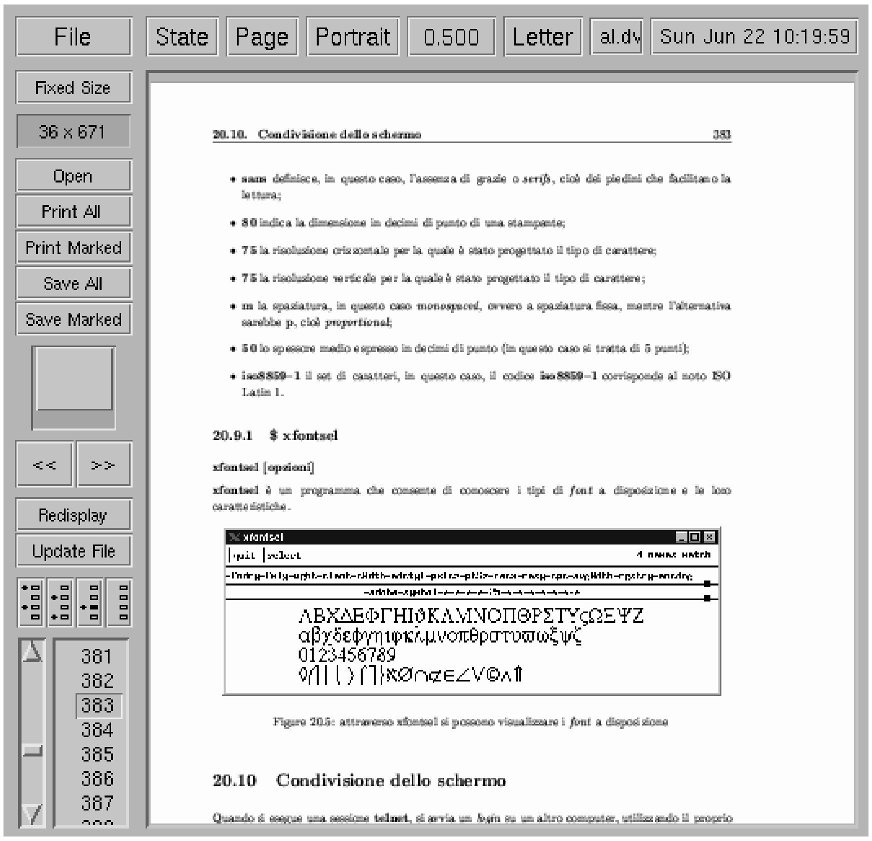

Figura 49.2: Il programma **gv**.

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.

49.4.1 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 stampa, occorre stabilire come deve essere girata la carta, e se deve essere invertita la sequenza dei fogli.

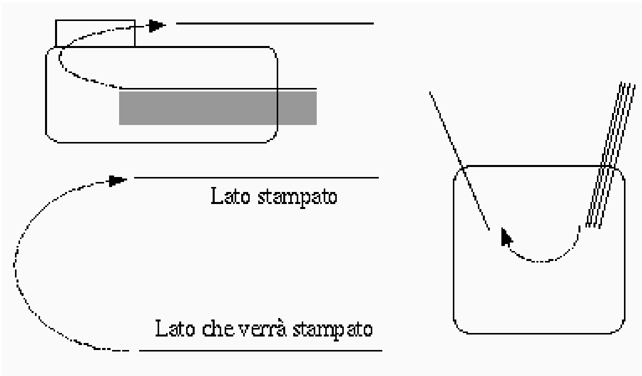

Figura 49.3: Movimento della carta in una tipica stampante laser.

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 49.4; rappresenta la sequenza necessaria per la stampa corretta di 4 facciate su un unico foglio, tenendo conto che questo deve essere reimmesso nella stampante.

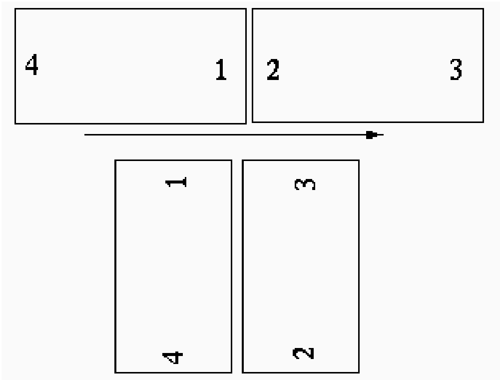

Figura 49.4: Sequenza di stampa per la stampa di 4 facciate su un foglio normale.

Se la stampante funziona come mostrato nella figura 49.3, si può comprendere il meccanismo osservando la sequenza di operazioni mostrata dalla figura 49.5. In pratica, dopo la stampa della prima facciata, occorre prendere il foglio senza ruotarlo e reimmetterlo in ingresso per la stampa.

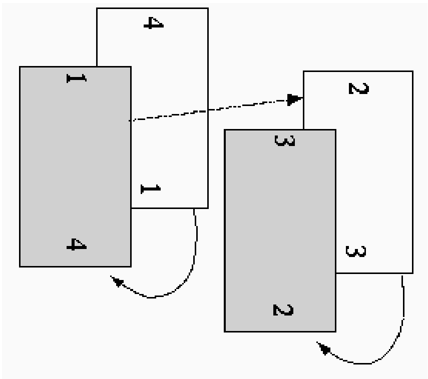

Figura 49.5: Sequenza pratica per la stampa di 4 facciate su un foglio normale.

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 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, 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 figli 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'.

49.4.2 $ 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*.

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 finale, 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, si sopprime questa segnalazione.

49.4.3 $ 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 da 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, si sopprime questa segnalazione.

Esempi

`$` ***`psselect -p1,3 documento.ps miofile.ps`***

Estrae dal file `documento.ps` la prima e la terza pagina, generando il file `miofile.ps`.

`$` ***`psselect -p1,_3 documento.ps miofile.ps`***

Estrae dal file `documento.ps` la prima e la terza pagina dalla fine, generando il file `miofile.ps`.

$ ***psselect -p1-3,10-15 documento.ps miofile.ps***

Estrae dal file `documento.ps` le prime tre pagine e le pagine dalla 10 alla 15, generando il file `miofile.ps`.

$ ***psselect -p-3,10-15 documento.ps miofile.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 miofile.ps***

Estrae dal file `documento.ps` la terza pagina e tutte le pagine a partire dalla 150, generando il file `miofile.ps`.

$ ***psselect -e -p150- documento.ps miofile.ps***

Estrae dal file `documento.ps` tutte le pagine pari (*even*) a partire dalla 150, generando il file `miofile.ps`.

$ ***psselect -o -r -p150- documento.ps miofile.ps***

Estrae dal file `documento.ps` tutte le pagine dispari (*odd*) a partire dalla 150, in ordine inverso, generando il file `miofile.ps`.

49.4.4 $ psnup

`psnup [`⟨*opzioni*⟩`] [`⟨*file-originale*⟩ `[`⟨*file-elaborato*⟩`]]`

`psnup` elabora un file PostScript generando un file in cui diverse pagine di origine sono assemblate in una unica finale. In pratica permette di ottenere due o più pagine in una 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*.

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 finale, 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, si sopprime questa segnalazione.

`-`*n*

Un trattino seguito da un numero indica la quantità di pagine che si vogliono unire in una unica 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 miofile.ps***

Elabora il file `documento.ps` (A4) generando il file `miofile.ps` (A4) che, per ogni pagina, conterrà due pagine del documento originale.

$ ***psnup -4 documento.ps miofile.ps***

Come nell'esempio precedente, con la differenza che vengono riunite 4 pagine in una sola facciata.

$ ***psnup -Pletter -4 documento.ps miofile.ps***

Come nell'esempio precedente, con la differenza che il file originale conteneva pagine in formato **letter**.

Annotazioni

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**.

49.4.5 $ 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.

I gruppi di pagine possono essere di dimensioni maggiori, 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, si sopprime questa 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 miofile.ps***

Elabora il file documento.ps generando il file miofile.ps con una sequenza del tipo 4-1+2-3.

$ ***psbook -s8 documento.ps miofile.ps***

Elabora il file documento.ps generando il file miofile.ps con una sequenza del tipo 8-1+2-7+6-3+4-5.

49.4.6 Esempi composti

I programmi del pacchetto PSUtils sono potentissimi, ma anche complicati da usare. Alcuni esempi per comprendere come combinarli assieme dovrebbe essere di aiuto.

$ ***psbook -s4 originale.ps | psnup -2 > miofile.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.

$ ***psbook -s4 originale.ps | psnup -2 | psselect -e > miofile.ps***

Come nell'esempio precedente, selezionando solo le pagine pari del risultato finale.

$ ***psbook -s4 originale.ps | psnup -2 | psselect -o -r > miofile.ps***

Come nell'esempio precedente, selezionando solo le pagine dispari del risultato finale, e invertendone l'ordine.

$ ***psbook originale.ps | psnup -2 > miofile.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.

49.5 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/>

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 generalmente distribuiti anche gli strumenti adatti a gestire tale formato.

50.1 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 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.

50.1.1 $ 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 miofile.ps miofile.dvi`***

Elabora il file `miofile.dvi` generando il file `miofile.ps`, in formato A4.

`$` ***`dvips -t a4 miofile.dvi`***

Elabora il file `miofile.dvi`, trasformandolo in PostScript, inviando il risultato alla coda di stampa predefinita.

50.2 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.

50.2.1 xdvi

È un programma che permette la visualizzazione di file DVI all'interno dell'ambiente grafico X Window System. In presenza di immagini incorporate di tipo PostScript, richiede la presenza di GhostScript.

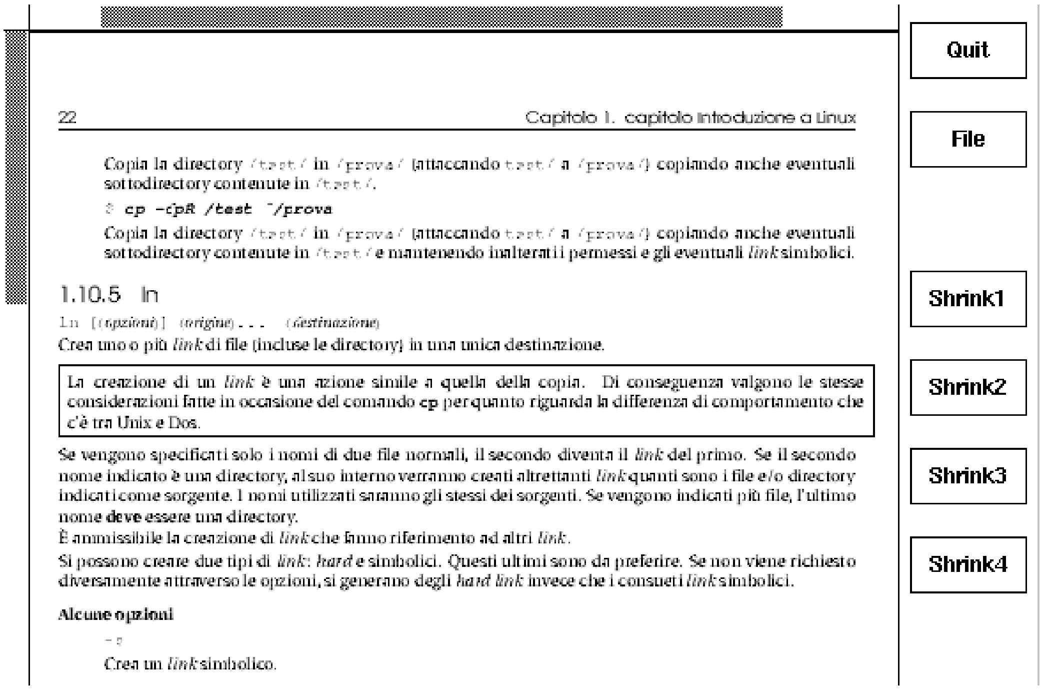

Figura 50.1: Il programma **`xdvi`**.

`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*⟩]

`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.
