20.1 Unità di riferimento

I programmi GNU che hanno a che fare con la misurazione di quantità di byte, possono essere configurati facilmente per ciò che concerne il sistema di misura da utilizzare. Esiste tradizionalmente un divario tra la misurazione dei dati e il SI (il *Sistema internazionale di unità*, *http://www.bipm.org/*). Per cercare di mettere un po' di ordine in questa confusione, è possibile intervenire su alcune variabili di ambiente.

Attraverso la variabile di ambiente ***BLOCK_SIZE***, è possibile stabilire la dimensione di un «blocco», cioè la quantità di byte usata come unità di riferimento. In generale, se la variabile di ambiente ***POSIXLY_CORRECT*** è stata definita (indipendentemente dal suo contenuto) e la variabile ***BLOCK_SIZE*** non c'è, i blocchi sono di 512 byte; diversamente, è la variabile ***BLOCK_SIZE*** a prevalere. Inoltre, se nessuna di queste variabili è presente, il blocco predefinito è di 1024 byte.

Nel gergo che si è sviluppato nei programmi GNU, dal momento che i blocchi di 512 byte sono difficili da valutare rapidamente, si fa riferimento al concetto di «*human readable*» per dire che i blocchi sono da 1024 byte. Tuttavia, la facilità di lettura che si sottintende in questo modo, è solo approssimativa, perché le convenzioni umane portano a pensare in base a una numerazione decimale. Pertanto, la variabile ***BLOCK_SIZE***, oltre a contenere un numero che rappresenta la dimensione del blocco in byte, può contenere due stringhe ben precise, a cui viene dato un significato determinato. Si veda la tabella 20.1.

Tabella 20.1. Valori assegnabili alla variabile di ambiente ***BLOCK_SIZE***.

Contenuto	Descrizione
n	Esprime la dimensione del blocco in byte.
`human-readable`	Indica un blocco di 1024 byte.
`si`	Indica un blocco di 1000 byte, secondo il SI.

In particolare, quando la variabile di ambiente ***BLOCK_SIZE*** contiene la stringa '**`si`**', il valore dei simboli usati come moltiplicatori, assume un significato diverso. Si osservi per questo la tabella 20.2.

Tabella 20.2. Moltiplicatori usati nei programmi GNU, a seconda dell'impostazione della variabile di ambiente ***BLOCK_SIZE***.

Simbolo	'**`human-readable`**'	'**`si`**'
k	2^{10}	10^{3}
M	2^{20}	10^{6}
G	2^{30}	10^{9}
T	2^{40}	10^{12}
P	2^{50}	10^{15}
E	2^{60}	10^{18}
Z	2^{70}	10^{21}
Y	2^{80}	10^{24}

A livello della riga di comando, si può intervenire attraverso opzioni comuni che, in generale, prevalgono sulle impostazioni delle variabili di ambiente. La tabella 20.3 riassume queste opzioni.

Tabella 20.3. Opzioni comuni relative alla definizione della dimensione dei blocchi.

Opzione	Descrizione
`--block-size=`***n***	Esprime la dimensione del blocco in byte.
`--block-size=human-readable`	Indica un blocco di 1024 byte.
`--block-size=si`	Indica un blocco di 1000 byte, secondo il SI.
`--kilobytes` `-k`	Indica un blocco di 1024 byte.
`--human-readable` `-h`	Indica un blocco di 1024 byte.
`--si` `-H`	Indica un blocco di 1000 byte, secondo il SI.

Sarebbero disponibili anche altre variabili di ambiente che permettono la configurazione specifica di ogni programma di servizio. Eventualmente si può consultare la documentazione originale.

20.2 Directory

La directory è un tipo speciale di file, il cui scopo è quello di contenere riferimenti ad altri file e ad altre directory. Pertanto, la directory è un indice di file ed eventualmente di altri sottoindici.

I permessi attribuiti a una directory vanno interpretati in maniera particolare:

- il permesso di lettura permette di conoscere il contenuto di una directory attraverso un programma come '**`ls`**' o simile, senza il quale, la directory può essere attraversata ugualmente;
- il permesso di scrittura permette di modificarne il contenuto, cioè di aggiungere o eliminare file e altre directory;
- il permesso di «esecuzione» permette il suo attraversamento, ovvero permette di raggiungere il suo contenuto o quello di altre directory discendenti.

20.2.1 Utilizzo di «mkdir»

Il programma di servizio '**`mkdir`**'[1] (*Make directory*) crea una o più directory:

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

In mancanza di indicazioni gli attributi della nuova directory sono 777_8 meno i bit della maschera dei permessi. Il valore tipico di questa maschera è 022_8 e di conseguenza gli attributi normali di una nuova

directory sono 755_8, cosa che in pratica permette a tutti di accedere e leggerne il contenuto, ma concede solo al proprietario di modificarle.

Tabella 20.4. Alcune opzioni.

Opzione	Descrizione
`-m` ***modalità_dei_permessi*** `--mode=`***modalità_dei_permessi***	Permette di definire esplicitamente la modalità dei permessi attribuiti alle directory che vengono create. Questa modalità può essere attribuita in forma numerica o in forma simbolica. La sintassi della forma simbolica è descritta in occasione della presentazione del programma '**chmod**' (20.7.3).
`-p` `--parents`	Fa in modo che vengano create anche le directory precedenti se queste non sono presenti. In tal caso la modalità utilizzata, per i permessi di queste directory precedenti, corrisponde a quanto stabilito per quella o quelle directory da creare con l'aggiunta (se necessario) dei permessi di scrittura e di 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**'.

20.2.2 Utilizzo di «rmdir»

Il programma di servizio '**rmdir**'[2] (*Remove directory*) elimina le directory indicate, se sono vuote:

`rmdir` [***opzioni***] ***directory***...

Tabella 20.5. Alcune opzioni.

Opzione	Descrizione
`-p` `--parents`	Elimina anche le directory precedenti se, dopo la cancellazione delle directory finali, queste restano vuote.

20.3 Percorsi

Il ***percorso*** o *path* è il modo con cui si identifica la posizione di un file o di una directory. File e directory vengono spesso indicati per nome facendo riferimento a una posizione sottintesa: la directory corrente (o attuale). File e directory possono essere indicati utilizzando un nome che comprende anche l'indicazione del percorso necessario a raggiungerli.

20.3.1 Utilizzo di «pwd»

Il programma di servizio '**pwd**'[3] (*Print working directory*) emette attraverso lo standard output il percorso assoluto della directory corrente. Viene mostrato il percorso reale, traducendo i collegamenti simbolici.

`pwd` [***opzioni***]

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.

20.3.2 Utilizzo di «basename»

Il programma di servizio '**basename**'[4] estrae il nome di un file o di una directory da un percorso:

`basename` ***percorso*** [***suffisso***]

In pratica: rimuove dal percorso la parte anteriore contenente l'informazione sulla directory; inoltre, eventualmente, rimuove anche il suffisso indicato dalla parte finale del nome rimanente. Il risultato viene emesso attraverso lo standard output.

Segue una sequenza di esempi senza spiegazione, da cui si vede comunque come si comporta il programma:

```
$ basename "/idrogeno/ossigeno" [Invio]

  ossigeno
$ basename "/idrogeno/eliografia.sh" ".sh" [Invio]

  eliografia
$ basename "/idrogeno/eliografia.sh" "grafia.sh" [Invio]

  elio
```

20.3.3 Utilizzo di «dirname»

Il programma di servizio '**dirname**' [5] (*Directory name*) estrae la directory da un percorso:

`dirname` ***percorso***

In pratica: rimuove dal percorso la parte finale a partire dall'ultima barra obliqua ('**/**') di divisione tra l'informazione della directory e il nome del file. Se il percorso contiene solo un nome di file, il risultato è un punto singolo ('**.**'), cioè la directory corrente. Il risultato viene emesso attraverso lo standard output.

Segue una sequenza di esempi senza spiegazione, da cui si vede comunque come si comporta il programma:

```
$ dirname "/idrogeno/ossigeno/elio" [Invio]

  /idrogeno/ossigeno
$ dirname "/idrogeno/ossigeno" [Invio]

  /idrogeno
$ dirname "/idrogeno" [Invio]

  /
$ dirname "idrogeno" [Invio]

  .
```

20.3.4 Utilizzo di «namei»

Il programma di servizio '**namei**' [6] scompone un percorso finché raggiunge un punto terminale:

`namei` [***opzioni***] ***percorso***...

In pratica, attraverso '**namei**' vengono analizzati i percorsi forniti, ne viene scomposto e descritto il contenuto nelle varie (eventuali) sottodirectory, quindi, se tra gli elementi contenuti nei percorsi richiesti esistono dei collegamenti simbolici, viene visualizzato anche l'elemento di destinazione. Questo programma è particolarmente utile per seguire i collegamenti simbolici, soprattutto quando questi hanno troppi livelli, cioè quando un collegamento punta a un altro collegamento ecc. I vari elementi visualizzati sono preceduti da una lettera che ne descrive le caratteristiche:

Sigla	Tipo corrispondente
`f:`	il percorso che si sta analizzando;
`d`	directory;
`l`	collegamento simbolico;
`s`	socket di dominio Unix;
`b`	file di dispositivo a blocchi;
`c`	file di dispositivo a caratteri;
`-`	file normale;
`?`	errore.

Segue la descrizione di alcuni esempi.

- `$ namei /usr/bin/X11` [*Invio*]

  Genera il risultato seguente:

```
f: /usr/bin/X11
 d /
 d usr
 d bin
 l X11 -> ../X11R6/bin
   d ..
   d X11R6
   d bin
```

  Da questo si intende che la directory '`/usr/bin/X11/`' in realtà non esiste, trattandosi di un collegamento simbolico alla vera directory '`/usr/X11R6/bin/`'.

- `$ namei /usr/bin/x-www-browser` [*Invio*]

```
f: x-www-browser
 l x-www-browser -> /etc/alternatives/x-www-browser
   d /
   d etc
   d alternatives
   l x-www-browser -> /usr/bin/mozilla
     d /
     d usr
     d bin
     l mozilla -> /etc/alternatives/mozilla
       d /
       d etc
       d alternatives
       l mozilla -> /usr/bin/mozilla-1.6
         d /
         d usr
         d bin
         - mozilla-1.6
```

  In questo caso si analizza il nome '`/usr/bin/x-www-browser`' che, nelle distribuzioni GNU/Linux Debian, rappresenta il «navigatore» grafico standard installato nel sistema. Come si vede, dopo una serie di passaggi, si arriva al programma '`/usr/bin/mozilla-1.6`'.

20.3.5 Utilizzo di «pathchk»

Il programma di servizio '**`pathchk`**'[7] (*Path check*) consente di controllare i percorsi:

```
pathchk [opzioni] [percorso...]
```

Per ogni percorso indicato come argomento viene eseguita una verifica e, se necessario, viene emesso attraverso lo standard output un messaggio per informare di uno dei problemi seguenti:

- una delle directory esistenti, indicate all'interno di uno dei percorsi, non ha il permesso di esecuzione necessario per essere attraversata;
- la lunghezza totale di un percorso è maggiore di quella gestibile con quel tipo di file system;

- la sola lunghezza di uno degli elementi di un percorso è maggiore di quella gestibile con quel tipo di file system.

Tabella 20.16. Alcune opzioni.

Opzione	Descrizione
`-p` `--portability`	Invece di eseguire un controllo in base alle possibilità del file system effettivamente in funzione, il programma si basa sulle specifiche minime stabilite dallo standard POSIX sulla portabilità; inoltre, viene controllato che non siano usati caratteri che potrebbero creare problemi di portabilità.

Il programma restituisce zero se tutti i percorsi hanno superato i controlli con successo; altrimenti restituisce il valore uno.

Seguono alcuni esempi autoesplicativi:

```
$ pathchk -p /home/perché [Invio]

  path '/home/perché' contains nonportable character 'é'
$ pathchk -p "/home/tizio/ciao bella" [Invio]

  path '/home/tizio/ciao bella' contains nonportable
  character ' '
```

Naturalmente, se il percorso è corretto, non si ottiene alcuna segnalazione.

20.4 Contenuti

Quando a un programma devono essere passati uno o più nomi di file tra gli argomenti, si possono rappresentare più nomi contemporaneamente attraverso un modello che fa uso di simboli adatti (a questo proposito si parla anche di *globbing*). La trasformazione del modello in elenchi di file (e directory) esistenti effettivamente, è compito della shell, cioè si tratta di qualcosa a cui gli altri programmi sono normalmente estranei. Nella sezione 17.2.2.9 viene trattato il modo con cui una shell POSIX si comporta al riguardo.

Il contenuto di una directory viene analizzato normalmente attraverso il programma '**`ls`**'. In particolare, la realizzazione GNU di '**`ls`**' può essere configurata per colorare i nomi dei file in modo diverso a seconda del tipo di questi.

20.4.1 Utilizzo di «ls», «dir» e «vdir»

Il programma di servizio '**`ls`**'[8] (*List*) visualizza i nomi di file o il contenuto delle directory indicate:

```
ls [opzioni] [nome...]
```

In mancanza dell'indicazione esplicita di file o di directory, 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.

> È importante ricordare che se vengono indicati dei nomi di file o directory nella riga di comando, è compito della shell espandere eventuali caratteri jolly (o metacaratteri). Di conseguenza, in questo caso, è la shell che non fornisce a '**`ls`**' i nomi che iniziano con un punto.

Il funzionamento predefinito di '**`ls`**' GNU dipende anche dalla configurazione fatta attraverso '**`dircolors`**'. In generale, se non viene indicato diversamente, '**`ls`**' genera un elenco ordinato per colonne se lo standard output è diretto allo schermo del terminale, oppure un elenco su un'unica colonna se viene diretto altrove. Questa particolarità è molto importante per poter gestire l'output di questo programma attraverso elaborazioni successive.

Tabella 20.19. Alcune opzioni.

Opzione	Descrizione
`-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 stessa ('`.`') e a quella genitrice ('`..`').

`-b`
`--escape`
`--quoting-style=escape`
	Se i nomi dei file contengono caratteri che non possono essere visualizzati, usa una notazione ottale al loro posto.

`-l`
`--format=long`
`--format=verbose`
	Oltre ai nomi, vengono visualizzati il tipo, i permessi, la quantità di collegamenti fisici, il nome dell'utente proprietario, il nome del gruppo, la dimensione in byte, la data di modifica.

`-q` `--hide-control-chars`	Utilizza il punto interrogativo per sostituire i caratteri «non stampabili» (quelli che non possono essere visualizzati) che dovessero essere contenuti eventualmente nei nomi.
`-R` `--recursive`	Vengono elencati i contenuti di tutte le directory in modo ricorsivo.
`-t` `--time=time`	Ordina il contenuto delle directory in funzione della data: dalla più recente alla più antica. Se non viene specificato diversamente, si fa riferimento alla data di modifica.

`-c`
`--time=ctime`
`--time=status`
	Utilizza la data di cambiamento dello stato dei file (ovvero la data di creazione, anche se questa definizione non è perfetta). Se viene usato il formato lungo di visualizzazione ('**`-l`**'), viene indicata questa data; se l'opzione '**`-c`**' viene usata insieme a '**`-t`**', l'elenco viene ordinato in base a questa data.

`-u`
`--time=atime`
`--time=access`
`--time=use`
	Utilizza la data di accesso ai file. Se viene usato il formato lungo di visualizzazione ('**`-l`**'), viene indicata questa data; se l'opzione '**`-u`**' viene usata insieme a '**`-t`**', l'elenco viene ordinato in base a questa data.

`-e` `--full-time`	Quando l'elenco comprende l'indicazione della data, questa viene espressa in modo dettagliato.
`-i` `--inode`	Emette, alla sinistra delle indicazioni inerenti i file, il numero di inode.
`-r` `--reverse`	Riordina in modo inverso rispetto al normale.
`-B` `--ignore-backups`	Esclude dall'elenco i file che terminano con il simbolo tilde ('`~`'). Infatti, questo simbolo viene utilizzato normalmente per distinguere le copie di sicurezza delle versioni precedenti di file che hanno la stessa radice.
`-C` `--format-vertical`	Emette un elenco organizzato in colonne, indipendentemente dalla destinazione dello standard output.

Opzione	Descrizione
`-F` `--classify`	Se non è già la modalità di funzionamento predefinita, aggiunge un carattere alla fine dei nomi dei file, in modo da riconoscerne il tipo: '`*`' eseguibile; '`/`' directory; '`@`' collegamento simbolico; '`|`' file FIFO (*pipe* con nome); '`=`' 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.
`-w` ***n_colonne*** `--width` ***n_colonne***	Definisce la larghezza a disposizione per l'elenco. L'argomento dell'opzione si riferisce al numero di caratteri utilizzabili. Di solito, la larghezza viene determinata in funzione del numero di colonne che ha a disposizione il terminale o la finestra del terminale.
`-I` ***modello*** `--ignore` ***modello***	Permette di escludere dall'elenco i file che sono rappresentati dal modello specificato, quando questi non sono indicati espressamente nella riga di comando. Bisogna tenere presente che il modello in questione deve essere interpretato da '**`ls`**' e non dalla shell. In pratica, è necessario delimitarlo o utilizzare dei caratteri di protezione per evitare l'intervento della shell.

Segue la descrizione di alcuni esempi.

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

  Visualizza un elenco lungo del contenuto della directory corrente.

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

  Visualizza un elenco lungo del contenuto della directory corrente, che include anche i nomi che iniziano con un punto.

- `$ ls -R /*/*/dir*` [*Invio*]

  Cerca, a partire dal secondo livello dopo la directory radice, gli elementi che iniziano per «dir».

- `$ ls -I \*.html` [*Invio*]

  Elenca il contenuto della directory corrente, escludendo i file corrispondenti al modello '`*.html`'. La barra obliqua inversa davanti all'asterisco serve per richiedere alla shell di non espanderlo e non viene passata a '**`ls`**'.

Il programma '**`ls`**' può essere avviato usando nomi differenti, ottenendo così un comportamento particolare:

```
dir [opzioni] [nome...]
```

```
vdir [opzioni] [nome...]
```

Usando il nome '**`dir`**' si ottiene l'equivalente di '**`ls -C -b`**', mentre con il nome '**`vdir`**' si ottiene l'equivalente di '**`ls -l -b`**'. In pratica, nel primo caso si vuole ricordare il funzionamento del comando '**`DIR`**' dei sistemi Dos, mentre nel secondo il prefisso «v» sta per *vertical*.

20.4.2 Utilizzo di «dircolors»

Il programma '**dircolors**' configura la colorazione e le modalità predefinite di funzionamento di '**ls**':

```
eval 'dircolors [opzioni] [file]'
```

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`', il quale si riferisce alla configurazione generale del sistema dei colori per '**ls**'.

Il programma '**dircolors**' è fatto per essere avviato immediatamente dopo l'esecuzione di una shell, in quanto la configurazione si traduce nella creazione della variabile di ambiente ***LS_COLORS***, con la quale si possono definire degli alias di shell per attuare in pratica questa configurazione. Per analizzarne il contenuto basta utilizzare il comando seguente:

```
$ echo "$LS_COLORS" [Invio]
```

Si ottiene un record molto lungo. Di seguito appare un esempio di questo, spezzato in più parti per poterlo consultare.

```
rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:↩
↪bd=40;33;01:cd=40;33;01:or=40;31;01:su=37;41:sg=30;43:↩
↪ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:↩
↪*.tgz=01;31:*.arj=01;31:*.taz=01;31:*.lzh=01;31:↩
↪*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.zip=01;31:↩
↪*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lz=01;31:↩
↪*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:↩
↪*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:↩
↪*.jar=01;31:*.rar=01;31:*.ace=01;31:*.zoo=01;31:↩
↪*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.jpg=01;35:↩
↪*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:↩
↪*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:↩
↪*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:↩
↪*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:↩
↪*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:↩
↪*.mkv=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:↩
↪*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:↩
↪*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:↩
↪*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:↩
↪*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:↩
↪*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.axv=01;35:↩
↪*.anx=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:↩
↪*.au=00;36:*.flac=00;36:*.mid=00;36:*.midi=00;36:↩
↪*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:↩
↪*.ra=00;36:*.wav=00;36:*.axa=00;36:*.oga=00;36:↩
↪*.spx=00;36:*.xspf=00;36:
```

Con questa variabile si può costruire un alias al programma '**ls**'.

```
$ alias ls='/bin/ls --color' [Invio]
```

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 nelle pagine di manuale *dircolors(1)* e *ls(1)*.

20.4.3 Utilizzo di «file»

Il programma di servizio '**file**'[9] consente di determinare il tipo di file:

```
file [opzioni] file…
```

Il programma analizza i file indicati come argomento e cerca di classificarli utilizzando l'ordine di analisi seguente: file system, 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.

20.4.4 Utilizzo di «du»[10]

Il programma di servizio '**du**'[11] (*Disk usage*) emette una sorta di statistica dell'utilizzo dello spazio da parte di un elenco di file o directory (in base al loro contenuto):

```
du [opzioni] file…
```


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 1 024 byte.

Tabella 20.21. Alcune opzioni.

Opzione	Descrizione
`-a` `--all`	Emette il conteggio riferito a tutti i file, non solo alle directory.
`-b` `--byte`	Emette le dimensioni in byte.
`-k` `--kilobytes`	Emette le dimensioni in kibibyte (simbolo: «Kibyte»). Questa opzione fa riferimento all'unità di misura predefinita, ma permette di fare ignorare a '**du**' la presenza eventuale della variabile ***POSIXLY_CORRECT***.
`-m` `--megabytes`	Emette le dimensioni in mebibyte (simbolo: «Mibyte»).
`-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 file system diverso da quello di partenza.

20.5 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***.

Per motivi di sicurezza, se si aggiunge il punto che simboleggia la directory corrente, nell'elenco dei percorsi della variabile ***PATH***, questo si colloca alla fine, come ultima possibilità.

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 file system. 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 [Invio]
```

20.5.1 Utilizzo di «which»

Il comando '**which**' simula la ricerca che farebbe la shell per avviare i programmi indicati negli argomenti e determina la posizione di quelli che verrebbero scelti. Ciò è utile per sapere: sia dove si trova un comando determinato, sia quale programma viene scelto effettivamente nel caso ne esistano diversi con lo stesso nome collocati in posizioni differenti nell'albero di directory.

```
which programma...
```

In pratica, invece di essere un programma vero e proprio, potrebbe trattarsi semplicemente di un alias a un comando di shell. In effetti, '**which**' compie lo stesso ruolo del comando '**type -path**' della shell Bash.

20.5.2 Utilizzo di «whereis»

Il programma '**whereis**' individua i file binari, i sorgenti e le pagine di manuale dei file specificati nell'argomento. [12]

```
whereis [opzioni] file...
```

Si veda eventualmente la pagina di manuale *whereis(1)*.

20.6 Proprietà

Ogni file e directory appartiene necessariamente a un utente e a un gruppo simultaneamente. L'appartenenza a un utente o a un gruppo particolare attribuisce significato ai permessi di accesso. Questi sono distinguibili in base al fatto che chi vuole accedere sia l'utente proprietario, o un utente del gruppo proprietario o un altro utente non appartenente a queste due categorie.

20.6.1 Utilizzo di «chown»

Il programma di servizio '**chown**'[13] (*Change owner*) cambia la proprietà dei file:

```
chown [opzioni] [utente][:[gruppo]] file...
```

Se viene fornito solo il nome dell'utente o il suo numero UID, questo diviene il nuovo proprietario dei file. Se il nome dell'utente, o il suo numero, è seguito da due punti verticali ('`:`') e dal nome o dal numero di un gruppo (GID), vengono cambiate la proprietà dell'utente e la proprietà del gruppo. Se dopo '`:`' non segue il nome del gruppo, viene attribuito il gruppo principale a cui appartiene l'utente. Se prima di '`:`' non viene indicato il nome dell'utente, viene cambiata solo la proprietà del gruppo.[14]

La versione GNU del programma '**chown**' consente di usare il punto ('`.`') al posto dei due punti ('`:`'). È bene evitare l'uso del punto per questo scopo, non essendo previsto dallo standard; tuttavia, la particolarità del programma '**chown**' GNU, suggerisce di non usare nominativi per gli utenti, o per i gruppi, contenenti dei punti.

Tabella 20.22. Alcune opzioni.

Opzione	Descrizione
`-R` `--recursive`	Esegue l'operazione anche nelle sottodirectory.
`--dereference`	Se si incontrano dei collegamenti simbolici, interviene nei file a cui questi puntano. Si tratta del comportamento predefinito.
`--from=`*utente*`:`*gruppo*	Modifica solo i file che inizialmente appartengono all'utente e al gruppo indicati.

Segue la descrizione di alcuni esempi.

- `# chown tizio mio_file` [*Invio*]

  L'utente '**root**' cambia l'utente proprietario del file '`mio_file`', facendo in modo che diventi '**tizio**'.

- `# chown tizio:users mio_file` [*Invio*]

  L'utente '**root**' cambia l'utente e il gruppo proprietario del file '`mio_file`', facendo in modo che diventino rispettivamente '**tizio**' e '**users**'.

- `$ chown :users mio_file` [*Invio*]

  L'utente proprietario del file '`mio_file`' cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente. Tuttavia, questa funzionalità non è standard.

- `# chown -R --from=tizio caio mia_dir` [*Invio*]

  L'utente '**root**' cambia la proprietà dei file contenuti a partire dalla directory '`mia_dir/`', in modo tale da trasferire a '**caio**' ciò che prima era proprietà di '**tizio**'. Anche in questo caso si tratta di una funzionalità particolare della versione GNU del programma.

20.6.2 Utilizzo di «chgrp»

il programma di servizio '**chgrp**'[15] (*Change group*) cambia il gruppo proprietario di file e directory:

```
chgrp [opzioni] gruppo file...
```

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.

Tabella 20.23. Alcune opzioni.

Opzione	Descrizione
`-R` `--recursive`	Esegue l'operazione anche nelle sottodirectory.
`--dereference`	Se si incontrano dei collegamenti simbolici, interviene nei file a cui questi puntano. Questa modalità di funzionamento è predefinita.

Segue la descrizione di alcuni esempi.

- `$ chgrp users mio_file` [*Invio*]

  L'utente proprietario del file '`mio_file`' cambia il gruppo. Il gruppo indicato fa parte di quelli a cui appartiene l'utente.

- `# chgrp -R users /home/tizio` [*Invio*]

  L'utente '**root**' cambia il gruppo alla directory '`/home/tizio/`' e a tutto il suo contenuto.

20.7 Modalità dei permessi

I permessi di accesso, attribuiti ai file o alle directory, definiscono le operazioni che con questi possono essere compiute a seconda dell'utente. La loro gestione è già introdotta nella sezione 3.21. Brevemente, si distinguono tre tipi di accesso: lettura, scrittura, esecuzione o attraversamento. Il significato del tipo di accesso dipende dal tipo di file a 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, ovvero di poter conoscere l'elenco dei 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 di accesso 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 (*read*), scrittura (*write*) ed esecuzione o attraversamento (*execution*, *cross*), mentre quando si utilizza la notazione ottale, il numero quattro rappresenta un permesso di lettura, il numero due rappresenta un permesso di scrittura e il numero uno rappresenta un permesso di esecuzione o attraversamento. 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.

Oltre ai permessi di accesso per un file o per una directory, si annotano altre informazioni, definibili nell'insieme come ***modalità dei permessi***. In tutto vengono usate quattro cifre ottali (12 bit), dove la prima riguarda alcune situazioni particolari:

1. Sticky (*Save text image*), se si tratta di un eseguibile, durante l'esecuzione salva l'immagine testo nella memoria virtuale (in sistemi Unix recenti questo tipo di richiesta è perfettamente inutile e non viene presa in considerazione), se invece si tratta di una directory, comporta che la cancellazione o la ridenominazione di un file (o di una sottodirectory) sia ammesso solo se eseguita dal proprietario di questo;
2. SGID, se si tratta di un eseguibile, attiva il numero del gruppo (GID) durante l'esecuzione, ovvero, attribuisce all'eseguibile in funzione i privilegi del gruppo a cui appartiene, se si tratta invece di una directory, comporta che i file creati al suo interno acquistino l'appartenenza al gruppo della stessa directory;
3. SUID, se si tratta di un eseguibile, attiva il numero dell'utente (UID) durante l'esecuzione, ovvero, attribuisce all'eseguibile in funzione i privilegi dell'utente a cui appartiene.

Le altre tre cifre, riguardano rispettivamente i permessi di accesso attribuiti all'utente proprietario, al gruppo e agli altri utenti. Per esempio, la modalità 755_8, pari a 0755_8, indica che l'utente proprietario può leggere, modificare ed eseguire il file, mentre, sia gli utenti del gruppo, sia gli altri, possono soltanto leggere ed eseguire il file.

Riquadro 20.24. Confusione sul significato di «modalità».

Ciò che si intende, nei sistemi Unix, con il termine di «modalità» di un file, è generalmente un valore a 16 bit, contenuto in un inode, usato per rappresentare il tipo di file, nei suoi quattro bit più significativi, e i permessi nei restanti dodici (a livello di codice sorgente, questa modalità è rappresentata dal tipo derivato '**mode_t**', previsto nel file di intestazione '`sys/types.h`'). Pertanto, a seconda del contesto, la modalità può riferirsi ai soli permessi (come nella funzione ***chmod()***), oppure anche al tipo di file (come nella funzione ***mknod()***).

20.7.1 SGID e SUID

Il modo migliore per comprendere il funzionamento delle modalità SUID e SGID è quello di fare qualche prova. Si inizia facendo una copia dell'eseguibile '**touch**' nella propria directory personale.

```
tizio$ cd [Invio]

tizio$ pwd [Invio]

  /home/tizio
tizio$ cp /bin/touch . [Invio]

tizio$ ls -l touch [Invio]

  -rwxr-xr-x   1 tizio    tizio       33156 Mar  2 08:46 touch
```


Si deve agire temporaneamente come utente '**root**' per cambiare la modalità dei 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 la modalità SUID.

```
root# chmod u+s touch [Invio]

root# ls -l touch [Invio]

  -rwsr-xr-x   1 root     root        33156 Mar  2 08:46 touch
```

Si può quindi ritornare allo stato precedente, lasciando i privilegi dell'utente '**root**' e riprendendo l'identità dell'utente '**tizio**'.

```
root# exit [Invio]
```

Si può provare a creare un file utilizzando l'eseguibile '**touch**' su cui è stato attivato il bit SUID.

```
tizio$ ./touch superfile [Invio]

tizio$ ls -l superfile [Invio]

  -rw-rw-r--   1 root     tizio       0 Mar  2 09:03 superfile
```

Si può osservare che il file creato appartiene all'utente '**root**', pur essendo stato creato da un utente comune. Si può comprendere quindi, quanto sia pericoloso utilizzare queste modalità speciali, SUID e SGID, senza oculatezza.

È importante chiarire che le modalità SUID e SGID per uno script non hanno senso, perché non si tratta di un programma autonomo, ma di qualcosa che viene eseguito da un interprete. Eventualmente, è l'interprete a dovere avere le modalità SUID o SGID attive, perché lo script possa agire con i privilegi di un altro utente.

È chiaro che si tratta di un'ipotesi astratta: l'idea di attribuire le modalità SUID e SGID a un interprete (o a una shell) è semplicemente terribile.

I tre bit iniziali della modalità dei permessi meritano un po' di attenzione anche quando si tratta di directory. 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 di scrittura sulla directory. Il bit Sticky viene attribuito generalmente alla directory '`/tmp/`' (oltre che a '`/var/tmp/`') quando questa risulta accessibile da ogni utente in tutti i modi: '**drwxrwxrwt**'. Ciò permette di evitare che i file possano essere cancellati o rinominati da utenti diversi dai proprietari.

La directory con il bit SGID attivo ('**d--x--s--x**') fa in modo che i file (e le directory) che vengono creati al suo interno appartengano al gruppo della directory stessa. Al contrario, il bit SUID attivo non dovrebbe avere alcun effetto.

20.7.2 Maschera dei permessi: umask

Quando viene creato un file, questo appartiene automaticamente all'utente che lo crea e al gruppo principale dell'utente stesso. I permessi gli vengono attribuiti in base alla maschera dei permessi (*umask*). Questa maschera rappresenta i permessi che non vengono attribuiti.

Di solito, il suo valore è 022_8 e con questo, non viene attribuito il permesso di scrittura (2_8) né al gruppo proprietario, né agli altri utenti. Il valore di questa maschera può essere modificato attraverso un comando interno di shell: '**umask**' (17.3.4).

20.7.3 Utilizzo di «chmod»

il programma di servizio '**chmod**'[16] (*Change mode*) cambia la modalità dei permessi sui file indicati come argomento:

```
chmod [opzioni] modalità_dei_permessi file...
```

Le modifiche della modalità dei permessi avvengono in base alle specifiche indicate nell'argomento precedente all'elenco dei file e si possono esprimere con la sintassi seguente:

```
[u|g|o|a]...[{+|-|=}{r|w|x|X|s|t|u|g|o}...][,...]
```

Una combinazione delle lettere '**u**', '**g**', '**o**', '**a**' controlla il tipo di utenti a cui si vuole riferire il cambiamento di permesso:

u	Utente proprietario del file.
g	Gruppo proprietario del file.
o	Utente diverso.
a	Tutti gli utenti indifferentemente.

Se l'indicazione degli utenti su cui intervenire non viene fornita, la variazione agisce in funzione della maschera dei permessi che può essere modificata attraverso il comando di shell '**umask**' (17.3.4). In pratica, la variazione riguarda tutti i tipi di utente, a esclusione dei bit attivati nella maschera dei permessi.

I segni '**+**', '**-**', '**=**' indicano il tipo di cambiamento sui permessi:

+	Le modalità dei permessi indicate vengono aggiunte.
-	Le modalità dei permessi indicate vengono tolte.
=	Le modalità dei permessi vengono modificate in modo da diventare esattamente come indicato.

Il gruppo finale di lettere '**r**', '**w**', '**x**', '**X**', '**s**', '**t**', '**u**', '**g**', '**o**' indica i permessi su cui agire:

r	Permesso di accesso in lettura.
w	Permesso di accesso in scrittura (modifica).
x	Permesso di esecuzione o di attraversamento se si tratta di directory.
X	La lettera '**X**' (maiuscola) si comporta come '**x**', ma interviene sulle directory e solo sui file che hanno già un permesso di esecuzione per un utente qualunque. In pratica, si cerca di intervenire solo sui file per i quali il permesso di esecuzione (o di attraversamento) può avere senso.
s	Riguarda solo i file eseguibili e le directory. Attiva il bit SUID, o il bit SGID a seconda che il cambiamento intervenga sull'utente, sul gruppo o su entrambi.
t	La lettera '**t**' riguarda solo i file eseguibili e le directory. Attiva il bit Sticky.
u	Attribuisce le stesse modalità dei permessi che ha già l'utente proprietario di quel file.
g	Attribuisce le stesse modalità dei permessi che ha già il gruppo proprietario di quel file.
o	Attribuisce le stesse modalità dei permessi che hanno già gli altri utenti per quel file.

Non è possibile cambiare i permessi dei collegamenti simbolici: se si interviene su un collegamento simbolico si agisce in realtà sul file di destinazione.

Tabella 20.33. Alcune opzioni.

Opzione	Descrizione
-R --recursive	Esegue l'operazione anche nelle sottodirectory.
-c --changes	Avvia dei cambiamenti apportati.

Segue la descrizione di alcuni esempi.

- `$ chmod -R go-rwx ~/*` [*Invio*]

  Toglie sia al gruppo, sia agli altri utenti, la possibilità di accedere in qualunque modo ai file della propria directory personale e anche nelle sottodirectory successive.

- `$ chmod -R a+rX ~/*` [*Invio*]

  Interviene ricorsivamente nel contenuto della propria directory personale, assegna a tutto ciò che incontra il permesso di lettura, inoltre assegna il permesso di attraversamento a tutte le directory, così come il permesso di esecuzione a tutti i file che già risultano eseguibili per alcuni utenti.

20.8 Attributi speciali

Le caratteristiche standard di un file in un sistema Unix sono la proprietà (utente e gruppo) e i permessi. In alcuni casi è possibile attribuire altri attributi come quando si utilizza un file system Second-extended (Ext2, Ext3 o Ext4). Naturalmente, è compito del kernel fare in modo che questi attributi siano gestiti in modo corretto.

20.8.1 Utilizzo di «chattr»

il programma di servizio '**chattr**'[17] (*Change second extended attributes*) cambia gli attributi su un file system di tipo Second extended:

```
chattr [opzioni] [modalità] file...
```

L'interpretazione corretta di questi attributi dipende dal kernel e non ci si deve aspettare che tutti siano funzionanti come progettato; pertanto, conviene sempre verificare nella documentazione originale.

Gli attributi vengono espressi attraverso una modalità simbolica secondo la sintassi seguente, che comunque ne rappresenta solo alcuni:

```
+|-|={A|S|a|c|d|i|s|u}...
```

La variazione degli attributi si esprime attraverso i simboli '**+**', '**-**' e '**=**', secondo lo schema seguente:

+	Gli attributi indicati vengono aggiunti.
-	Gli attributi indicati vengono tolti.
=	Gli attributi vengono modificati in modo da diventare esattamente come indicato.

Gli attributi da variare si individuano utilizzando delle lettere; lo schema seguente ne descrive alcune:

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*). Solo l'utente '**root**' può attribuire o togliere questo attributo.
c	Fa in modo che il kernel provveda a comprimere e decomprimere automaticamente i file in modo trasparente.
d	Serve al programma '**dump**' per sapere che il file in questione non è candidato per un recupero (*backup*).

`i`	Fa in modo che il file non sia modificabile, né cancellabile, né sia possibile cambiargli nome, né sia possibile creare un collegamento fisico verso di esso (i collegamenti simbolici restano ammissibili). Solo l'utente '**`root`**' può attribuire o togliere questo attributo.
`s`	Fa in modo che la cancellazione sovrascriva i dati cancellati con altri 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).

20.8.2 Utilizzo di «lsattr»

Il programma di servizio '**`lsattr`**'[18] (*List second extended attributes*) elenca gli attributi dei file su un file system di tipo Second-extended:

```
lsattr [opzioni] file...
```

Tabella 20.36. Alcune opzioni.

Opzione	Descrizione
`-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.

20.9 ACL POSIX con i sistemi GNU/Linux

La sigla «ACL» sta per *Access control list* e si riferisce qui a un'estensione della gestione dei permessi, rispetto alla tradizione dei sistemi Unix.[19] POSIX ha prodotto alcune bozze sulla possibilità di estendere la gestione dei permessi dei sistemi Unix (POSIX 1003.1e e POSIX 1003.2c), ma tali lavori sono rimasti incompiuti. Queste bozze sono pubbliche e diversi sistemi Unix mettono a disposizione alcune di queste estensioni. Le estensioni a cui si fa riferimento con la sigla ACL, o eventualmente con «ACL POSIX» (benché si tratti solo di bozze), sono solo una porzione dell'insieme complessivo e in questa sezione si vuole descrivere in particolare la realizzazione relativa ai sistemi GNU/Linux.

20.9.1 Kernel, file system e programmi di servizio

La gestione delle estensioni ACL POSIX richiede che il file system sia in grado di accumulare le informazioni necessarie e che il kernel sia in grado di gestirle. Nel caso di un kernel Linux occorre abilitare la funzione nell'ambito della voce che consente la gestione di un certo file system; per esempio, nel caso dei file system Second extended:

- *Filesystems*
  - *Second extended fs support*
    - *Ext2 extended attributes*
      - *Ext2 POSIX Access Control Lists*
  - *Ext3 journalling file system support*
    - *Ext3 extended attributes*
      - *Ext3 POSIX Access Control Lists*
  - *The Extended 4 (ext4) filesystem*
    - *Ext4 extended attributes*
      - *Ext4 POSIX Access Control Lists*

A ogni modo, in un sistema GNU/Linux, il fatto che il kernel sia predisposto correttamente e che il file system sia adatto, non basta ad attivare la gestione delle estensioni ACL POSIX, perché è necessario innestare il file system con l'opzione '**`acl`**':


```
/dev/sda2  /  auto  defaults,acl,errors=remount-ro  0  1
```

L'esempio mostra una riga del file '`/etc/fstab`', in cui si dichiara la partizione usata come file system principale. Come si vede, tra le opzioni appare anche la sigla '**`acl`**' per attivare la gestione che interessa.

Eventualmente, un file system può essere reinnestato aggiungendo tale opzione:

```
# mount -o remount,acl / [Invio]
```

Per disabilitare le estensioni ACL si può usare l'opzione '**`noacl`**':

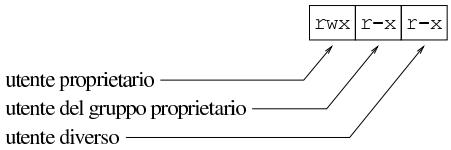
```
# mount -o remount,noacl / [Invio]
```

La gestione delle estensioni ACL POSIX richiede naturalmente la disponibilità di programmi di servizio appropriati. Nei sistemi GNU/Linux si usa il pacchetto ACL,[20] che si compone principalmente dei programmi '**`getfacl`**' e '**`setfacl`**'.

20.9.2 ACL POSIX

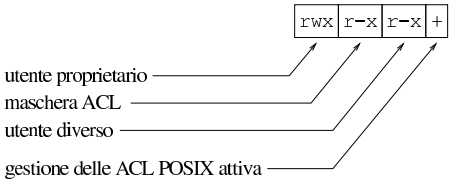
Nei confronti di un file (o di una directory), gli utenti di un sistema Unix sono classificati in tre classi: il proprietario del file; gli utenti che appartengono al gruppo a cui è associato il file; gli utenti che non rientrano nelle prime due classi. I permessi di un file (o di una directory) sono suddivisi in tre parti che riguardano rispettivamente le tre classi di utenti.

Figura 20.38. Associazione tra le classi di utenti e i permessi secondo lo schema tradizionale dei sistemi Unix.



Quando si inseriscono le ACL POSIX, i file e le directory possono contenere informazioni più articolate nei confronti dei permessi si accesso e degli utenti a cui questi permessi si rivolgono. Le informazioni riguardo ai permessi dell'utente proprietario e degli utenti che non sono proprietari e non appartengono nemmeno al gruppo associato al file, sono gestite come nel sistema tradizionale. Oltre a questo, è possibile dichiarare espressamente utenti, oppure gruppi, a cui si associano dei permessi specifici. La gestione di questi utenti e gruppi particolari, assieme alla gestione dei permessi del gruppo a cui appartiene il file, è sottoposto al filtro di una maschera, che però non è collegato alla maschera dei permessi tradizionale (*umask*).

Figura 20.39. Associazione tra le classi di utenti e i permessi mostrati dal comando '**`ls`**' quando è attiva la gestione delle ACL POSIX.



La maschera dei permessi ACL rappresenta precisamente il massimo delle possibilità che ha un utente appartenente a quell'insieme costituito da: utenti e gruppi particolari, assieme al gruppo associato al file stesso. In altri termini: se anche un utente particolare avesse i permessi di lettura e scrittura su quel file, ma la maschera concedesse, come nell'esempio, soltanto la lettura ed esecuzione, tale utente potrebbe accedere al file soltanto in lettura.

Questo tipo di maschera va inteso quindi come la rappresentazione di ciò che è concesso, mentre la maschera dei permessi tradizionale rappresenta ciò che viene sottratto nel momento in cui si crea un file o una directory.

Le informazioni sui permessi relative alle estensioni ACL POSIX sono elencate nella tabella successiva, dove si vede in particolare il modo con cui questi vengono rappresentati.

Tabella 20.40. I tipi di voci che possono comporre i permessi secondo le estensioni ACL POSIX.

Voce	Descrizione
`user::`***permessi***	Dichiarazione dei permessi associati all'utente proprietario (questi permessi non sono filtrati dalla maschera ACL).
`group::`***permessi***	Dichiarazione dei permessi associati agli utenti appartenenti al gruppo proprietario.
`user:`***utente***`:`***permessi***	Dichiarazione dei permessi associati a un utente particolare.
`user:`***gruppo***`:`***permessi***	Dichiarazione dei permessi associati agli utenti di un gruppo particolare.
`mask::`***permessi***	Dichiarazione della maschera dei permessi concessi a tutte le classi di utenti, escluso il proprietario e gli utenti che non ricadono in alcuna categoria specificata.
`other::`***permessi***	Dichiarazione dei permessi associati agli utenti che non vengono individuati in alcuna categoria particolare (questi permessi non sono filtrati dalla maschera ACL).

20.9.3 ACL in pratica

Se esistono i presupposti per usare le estensioni ACL POSIX (un kernel adatto e un file system compatibile, innestato con l'opzione '**`acl`**'), si possono fare delle prove per verificare la logica di questo sistema. Si crea inizialmente la directory '`/tmp/ACL/`' e ci si sposta al suo interno per comodità:

```
$ mkdir /tmp/ACL [Invio]

$ ls -l /tmp [Invio]

  …
  drwxr-xr-x    2 tizio    tizio     4096 2012-03-18 15:30 ACL
  …

$ cd /tmp/ACL [Invio]
```

Da quanto mostrato, si comprende che si sta operando utilizzando l'utenza '**`tizio`**', appartenente al gruppo '**`tizio`**' (evidentemente si utilizza la gestione dei gruppi privati); inoltre, si comprende che la maschera dei permessi tradizionale ha il valore 0022_8, ma tanto vale controllare:

```
$ umask [Invio]

  0022
```

Si crea un file vuoto e poi si verificano i permessi:

```
$ touch primo [Invio]

$ ls -l primo [Invio]

  totale 0
  -rw-r--r--    1 tizio    tizio      0 2012-03-18 15:36 primo
```

Con l'ausilio di '**`getfacl`**' si può vedere la descrizione dei permessi tradizionali in forma di ACL POSIX:

```
$ getfacl primo [Invio]

  # file: primo
  # owner: tizio
  # group: tizio
  user::rw-
  group::r--
  other::r--
```

Come si può osservare, le informazioni che si ottengono sono equivalenti alle solite che mostra il comando '**`ls`**', tradotte però secondo la forma che appare nella tabella 20.40.

Con l'ausilio di '**`setfacl`**' si aggiungono dei permessi particolari di accesso all'utente '**`caio`**':

```
$ setfacl -m user:caio:rw- primo [Invio]
```


Il comando, oltre che attribuire i permessi all'utente '**`caio`**', predispone in modo predefinito una maschera appropriata:

```
$ getfacl primo [Invio]

  # file: primo
  # owner: tizio
  # group: tizio
  user::rw-
  user:caio:rw-
  group::r--
  mask::rw-
  other::r--
```

Conviene osservare anche cosa mostra il comando '**`ls`**':

```
$ ls -l primo [Invio]

  -rw-rw-r--+   1 tizio    tizio      0 2012-03-18 15:36 primo
```

Come si vede, alla fine della stringa che esprime i permessi appare il segno '**`+`**', a indicare che sono presenti delle estensioni ACL e che il gruppo centrale di permessi è riferito alla maschera di queste estensioni e non più al gruppo proprietario.

A completamento dell'esempio si può aggiungere al file una voce riferita al gruppo '**`sempronio`**', a cui si vuole concedere di scrivere e di eseguire il file:

```
$ setfacl -m group:sempronio:-wx primo [Invio]
```

Si può osservare che in condizioni normali, se non si specificano altre opzioni, la voce che rappresenta la maschera viene adattata in modo automatico:

```
$ getfacl primo [Invio]

  # file: primo
  # owner: tizio
  # group: tizio
  user::rw-
  user:caio:rw-
  group::r--
  group:sempronio:-wx
  mask::rwx
  other::r--

$ ls -l primo [Invio]

  -rw-rwxr--+   1 tizio    tizio      0 2012-03-18 15:36 primo
```

A questo punto si decide di intervenire direttamente sulla maschera, ma per farlo si può agire in due modi equivalenti:

```
$ setfacl -m mask::-w- primo [Invio]
```

Oppure:

```
$ chmod g=w primo [Invio]
```

Naturalmente, si può usare '**`chmod`**' anche per modificare i permessi associati all'utente proprietario o agli utenti che non sono individuabili in altri modi, così come si può usare '**`setfacl`**' con le voci '**`user::`***permessi*** ' e '**`other::`***permessi*** '.

```
$ getfacl primo [Invio]

  # file: primo
  # owner: tizio
  # group: tizio
  user::rw-
  user:caio:rw-           #effective:-w-
  group::r--              #effective:---
  group:sempronio:-wx     #effective:-w-
  mask::-w-
  other::r--
```

A questo punto si vede che la maschera, ridotta al solo permesso di scrittura, interviene sui permessi «efficaci» associati all'utente '**`caio`**', al gruppo '**`sempronio`**' e agli utenti del gruppo proprietario. In base all'esempio, gli utenti del gruppo proprietario, avendo originariamente soltanto il permesso di lettura, dopo l'applicazione della maschera non hanno più alcun permesso di accesso.

Si osservi che l'adattamento automatico della maschera continua a persistere, nonostante la modifica manuale. Pertanto, se si aggiungono altri utenti e gruppi particolari, o se si interviene su quelli esistenti, la maschera viene aggiornata di conseguenza. Per evitarlo, quando si usa '**`setfacl`**' si aggiunge l'opzione '**`-n`**'.

Per cambiare i permessi di un utente o di un gruppo particolare basta impartire nuovamente il comando relativo, mentre per eliminare le voci di utenti e di gruppi particolari si usa l'opzione '**`-x`**' e non si specificano i permessi. Ecco come si eliminano le voci dell'utente '**`caio`**' e del gruppo '**`sempronio`**':

```
$ setfacl -x user:caio: primo [Invio]

$ setfacl -x group:sempronio: primo [Invio]

$ getfacl primo [Invio]

# file: primo
# owner: tizio
# group: tizio
user::rw-
group::r--
mask::r--
other::r--
```

Se si elimina anche la maschera, il comando '**`ls`**' non mostra più la presenza di estensioni ACL POSIX:

```
$ setfacl -x mask:: primo [Invio]

$ ls -l primo [Invio]

-rw-r--r--    1 tizio    tizio      0 2012-03-18 15:36 primo
```

20.9.4 Ereditarietà

I permessi con le estensioni ACL POSIX per le directory funzionano come per i file normali, con la differenza che è possibile stabilire delle voci predefinite.

Si inseriscono delle voci predefinite con l'opzione '**`-d`**' di '**`setfacl`**' e la prima volta vengono utilizzati anche i dati delle voci generali che appartengono già alla directory (utenti e gruppi particolari vengono ignorati). Per comprendere il meccanismo è necessario procedere con degli esempi, i quali continuano da quanto lasciato nella sezione precedente.

```
$ mkdir seconda [Invio]
```

Inizialmente, nulla di strano:

```
$ getfacl seconda [Invio]

# file: seconda
# owner: tizio
# group: tizio
user::rwx
group::r-x
other::r-x
```

Si aggiunge una voce relativa all'utente '**`caio`**', cui si concede anche di aggiungere e cancellare dei file:

```
$ setfacl -m user:caio:rwx seconda [Invio]

$ getfacl seconda [Invio]

# file: seconda
# owner: tizio
# group: tizio
user::rwx
user:caio:rwx
group::r-x
mask::rwx
other::r-x
```

Come già visto nella sezione precedente, la gestione delle estensioni ACL POSIX diventa attiva e si ottiene anche una maschera predefinita.

A questo punto si stabiliscono i permessi predefiniti, aggiungendo un altro utente particolare, che però può solo modificare e attraversare le directory:

```
$ setfacl -d -m user:mevio:-wx seconda [Invio]

$ getfacl seconda [Invio]

# file: seconda
# owner: tizio
# group: tizio
user::rwx
user:caio:rwx
group::r-x
mask::rwx
other::r-x
default:user::rwx
default:user:mevio:-wx
default:group::r-x
default:mask::rwx
default:other::r-x
```

Come si vede, appaiono alla fine delle voci precedute dalla sigla '**`default`**'.

Per dare un senso alle voci predefinite, occorre creare qualcosa all'interno della directory in questione:

```
$ cd seconda [Invio]

$ touch terzo [Invio]

$ getfacl terzo [Invio]

# file: terzo
# owner: tizio
# group: tizio
user::rw-
user:mevio:-wx  #effective:-w-
group::r-x      #effective:r--
mask::rw-
other::r--
```

I permessi (estesi) che ha ottenuto il file '**`terzo`**' dipendono però da quelli che il file otterrebbe in mancanza delle estensioni ACL POSIX. In questo caso, in presenza di una maschera dei permessi 0022_8, trattandosi di un file, si otterrebbero permessi del tipo 0644_8, ovvero '**`rw-r--r--`**'. Da quanto ottenuto si intende che la maschera ACL non ne è influenzata.

Se si crea una directory, questa eredita anche le voci predefinite:

```
$ mkdir quarta [Invio]

$ getfacl quarta [Invio]

# file: quarta
# owner: tizio
# group: tizio
user::rwx
user:mevio:-wx
group::r-x
mask::rwx
other::r-x
default:user::rwx
default:user:mevio:-wx
default:group::r-x
default:mask::rwx
default:other::r-x
```

Per eliminare le voci predefinite, si usa l'opzione '**`-k`**':

```
$ setfacl -k quarta [Invio]

$ getfacl quarta [Invio]

# file: quarta
# owner: tizio
# group: tizio
user::rwx
user:mevio:-wx
group::r-x
mask::rwx
other::r-x
```

20.10 Data

Tutti i file riportano tre indicazioni data-orario:

`ctime`	la data e l'ora di «creazione» (riferita all'inode): questa viene modificata in particolare quando si cambia lo stato del file (permessi e proprietà);
`mtime`	la data e l'ora di modifica: questa viene cambiata 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.

20.10.1 Utilizzo di «touch»

Il programma di servizio '**touch**'[21] cambia la data (si intende sia la data che l'ora) di accesso e di aggiornamento dei file:

```
touch [opzioni] file...
```

Se non si specifica una data, viene utilizzata la data e l'ora ottenuta dall'orologio del sistema nel momento in cui viene eseguito il comando. Se si specificano file che non esistono, questi vengono creati vuoti.

Tabella 20.59. Alcune opzioni.

Opzione	Descrizione

`-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` [[*SS*]*AA*]*MMGGhhmm*[*.ss*]	Usa l'argomento (secolo, anno, mese, giorno, ore, minuti, secondi) invece di utilizzare la data corrente.

20.11 Copia e collegamento

La copia genera un altro file o un'altra directory, il collegamento genera un riferimento aggiuntivo agli stessi dati di origine: assomiglia alla copia, ma rappresenta solo un modo per fare apparire la stessa cosa in più punti differenti.

Nei sistemi Unix i collegamenti sono molto importanti e vengono usati di frequente. Si distinguono due tipi di questi: collegamenti simbolici (*symbolic link*) e collegamenti fisici (*hard link*). Attraverso il collegamento fisico si creano dei riferimenti a dati esistenti in modo non distinguibile da quelli originali; i collegamenti simbolici sono dei file speciali e per questo distinguibili dai file originali.

A fianco del problema della copia di file (o di directory), cioè di entità virtuali per il contenimento dei dati, ci può essere il problema elementare (anche se complicato per l'utente) di trasferire dati attraverso i dispositivi in modo diretto (copia a basso livello).

20.11.1 Collegamenti simbolici

Si è accennato al fatto che i collegamenti simbolici sono dei file speciali, distinguibili dai file originali. Si creano normalmente utilizzando il programma '**ln**', con l'opzione '**-s**', come nell'esempio seguente:

```
$ ln -s /bin/sh ./sh [Invio]
```

Seguendo l'esempio, se si leggono le caratteristiche del file '`./sh`' attraverso '**ls**', si può notare l'indicazione esplicita del fatto che si tratta di un riferimento al file '`/bin/sh`' (il quale potrebbe essere un altro collegamento, ma questo adesso non è importante).

```
$ ls -l sh [Invio]

lrwxrwxrwx  1 tizio   tizio     7 Mar  2 10:16 sh -> /bin/sh
```

La lettera che appare all'inizio dei permessi, «l», indica esplicitamente che si tratta di un collegamento simbolico. Alla fine, viene indicato anche a chi punta il collegamento: '**-> /bin/sh**'.

Si può osservare inoltre che i permessi di un collegamento simbolico non esistono. Formalmente vengono mostrati come attivi tutti i permessi degli ultimi 9 bit (lettura, scrittura ed esecuzione per tutti gli utenti), perché quelli che contano sono in realtà i permessi del file (o della directory) cui effettivamente punta il collegamento simbolico.

L'esistenza dei collegamenti simbolici altera la logica normale della copia: ha senso copiare i file a cui i collegamenti puntano, o ha senso copiare i collegamenti? Solitamente si considera che la gestione dei collegamenti simbolici debba essere trasparente, come se questi non esistessero e si trattasse effettivamente dei file a cui loro puntano. Ma ciò vale fino a quando non si fa esplicitamente riferimento ai collegamenti in quanto tali.

20.11.2 Collegamenti fisici

La gestione dei collegamenti fisici è più seria, nel senso che deve essere riservata a situazioni di particolare necessità. Attraverso il collegamento fisico si creano dei riferimenti a dati esistenti in modo non distinguibile da quelli originali; in pratica, due o più voci nella stessa directory, o in directory differenti, possono puntare allo stesso file.

Quando si cancella un file, si elimina il riferimento al suo inode dalla directory che lo contiene formalmente. Quando un inode non ha più riferimenti, viene considerato libero e può essere riutilizzato per un altro file. In altre parole, se si utilizzano i collegamenti fisici, un file viene cancellato effettivamente quando sono stati eliminati tutti i riferimenti a questo.

Per comprendere in pratica cosa accade, si può provare con gli esempi seguenti.

```
$ touch mio_file [Invio]

$ ls -l mio_file [Invio]

  -rw-rw-r--   1 tizio    tizio       0 Mar  2 10:48 mio_file
$ ln mio_file tuo_file [Invio]

$ ls -l mio_file tuo_file [Invio]

  -rw-rw-r--   2 tizio    tizio       0 Mar  2 10:48 mio_file
  -rw-rw-r--   2 tizio    tizio       0 Mar  2 10:48 tuo_file
```

Come si vede, con questa serie di operazioni si è giunti ad avere due file, apparentemente indipendenti, ma se viene modificato il contenuto di uno si vedono le modifiche anche nell'altro. Dal momento che i permessi e la proprietà dei file (UID e GID) sono informazioni contenute nell'inode, la modifica di questi si ripercuote su tutti i collegamenti.

Si può osservare il numero che appare dopo i permessi, due, che indica quanti riferimenti ha l'inode corrispondente. In pratica, quel numero indica quante voci puntano a quello stesso file. Non si può sa-

pere facilmente quali siano gli altri riferimenti. Si può solo conoscere il numero dell'inode.

```
$ ls -l -i mio_file tuo_file [Invio]

  270385 -rw-rw-r--  2 tizio  tizio    0 Mar  2 10:48 mio_file
  270385 -rw-rw-r--  2 tizio  tizio    0 Mar  2 10:48 tuo_file
```

Come si vede, i due file hanno lo stesso inode (il numero che appare prima dei permessi), quindi **sono lo stesso file**.

20.11.3 Directory e collegamenti fisici

Ogni directory contiene due riferimenti convenzionali: uno a se stessa e uno alla directory genitrice ('`.`' e '`..`'). Si tratta di nomi di file a tutti gli effetti, i quali 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 genitrice, e quello rappresentato dal punto singolo (la stessa directory).

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 genitrice.

```
$ mkdir miadir [Invio]

$ ls -l -d -i miadir [Invio]

  157715 drwxrwxr-x  2 tizio  tizio  1024 Mar  2 11:22 miadir
```

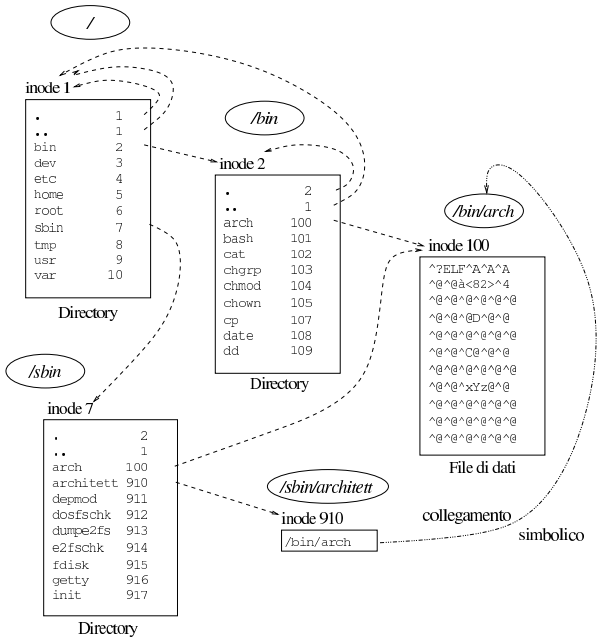
L'esempio mostra semplicemente il riferimento alla directory '`miadir/`' contenuto nella sua directory genitrice. Si può provare a leggere il contenuto della directory appena creata.

```
$ 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 punto singolo ('`.`') ha lo stesso numero di inode della directory '`miadir/`', cosa che spiega il motivo per cui una directory ha almeno due riferimenti (collegamenti fisici).

La directory genitrice, rappresentata dai due punti in sequenza ('`..`'), ha tre riferimenti totali per il solo fatto che esiste questa directory (in pratica: i due riferimenti naturali, più questo, perché esiste questa directory).

Figura 20.66. Modello di esempio del comportamento dei collegamenti.



20.11.4 Errori comuni con i collegamenti simbolici

Quando si creano collegamenti fisici, gli argomenti del comando '**`ln`**' che rappresentano i file di origine, servono a individuare gli inode, a cui si vuole fare riferimento nella destinazione. Al contrario, quando si creano collegamenti simbolici, gli argomenti che rappresentano l'origine vengono trattati semplicemente come stringhe da riprodurre nei riferimenti della destinazione. Ignorando questo particolare, è facile fare degli errori senza comprenderne la ragione. Si osservino i comandi seguenti:

```
$ cd [Invio]

$ pwd [Invio]

  /home/tizio
$ mkdir prova [Invio]

$ cd prova [Invio]

$ pwd [Invio]

  /home/tizio/prova
$ ln -s /bin/* . [Invio]
```

Così facendo si creano dei collegamenti simbolici ai file e alle directory contenuti all'interno della directory '`/bin/`', con lo stesso nome che hanno nell'origine. Per esempio, il collegamento simbolico '`/home/tizio/prova/sh`' punta precisamente a '`/bin/sh`'. Si osservi invece la sequenza di comandi seguente:

```
$ cd /bin [Invio]

$ pwd [Invio]

  /bin
$ mkdir ~/prova [Invio]

$ ln -s * ~/prova [Invio]
```

Se si va nella directory '`~/prova/`', si può osservare che i collegamenti puntano invece a loro stessi; per esempio, '`~/prova/sh`' punta a '`sh`', ovvero a nulla che possa servire. Infatti, quando si dà il comando '**`ln -s * ~/prova`**', si ottiene l'espansione dell'asterisco nell'elenco dei nomi (file e directory) contenuti nella directory

corrente, i quali vengono usati tali e quali per la preparazione dei collegamenti. Pertanto, nella destinazione i collegamenti puntano poi ai loro stessi nomi.

20.11.5 Utilizzo di «cp»

Il programma di servizio '**cp**'[22] (*copy*) copia i file:

```
cp [opzioni] origine destinazione
```

```
cp [opzioni] origine... directory
```

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 vengono generate le copie di tutti i file nella directory di destinazione. In mancanza di opzioni particolari, le directory non vengono copiate.

Nell'uso di '**cp**' occorre fare attenzione, perché si possono sovrascrivere altri file senza ottenere alcun 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 utilizzare l'opzione '**-f**'.

Tabella 20.70. Alcune opzioni di '**cp**' nella versione GNU. Le opzioni evidenziate sono conformi allo standard POSIX.

Opzione	Descrizione
`-a` `--archive`	Equivalente a '**-dpR**', utile per l'archiviazione o comunque per la copia di collegamenti simbolici così come sono.
`-b` `--backup`	Mantiene delle copie di sicurezza dei file che vengono sovrascritti con la copia.
`-d` `--no-dereference`	Copia i collegamenti simbolici mantenendoli come tali, invece di copiare i file a cui i collegamenti si riferiscono.
`-f` `--force`	Sovrascrittura forzata dei file di destinazione.
`-i` `--interactive`	Richiede una conferma per la sovrascrittura nel caso in cui esistano già dei file con i nomi uguali a quelli di destinazione della copia.
`-l` `--link`	Crea un collegamento fisico invece di copiare i file (non vale per le directory).
`-s` `--symbolic-link`	Crea un collegamento simbolico invece di copiare i file (non vale per le directory).

`-P`
`-d`
`--no-dereference`
	Copia i collegamenti simbolici come tali, invece di copiare i file a cui puntano.

`--parents`	Copia anche il percorso indicato nel file di origine.
`-p` `--preserve`	Mantiene le proprietà, le modalità dei permessi originali e le date originali.

Opzione	Descrizione
`-r`	Copia file e directory in modo ricorsivo (includendo le sottodirectory), considerando tutto ciò che non è una directory come un file normale. L'opzione '**-r**' è prevista dallo standard POSIX, ma viene considerata superata e candidata per l'eliminazione futura.
`-R` `--recursive`	Copia file e directory in modo ricorsivo (includendo le sottodirectory).
`-S` ***suffisso_di_backup*** `--suffix=`***suffisso_di_backup***	Permette di definire il suffisso da utilizzare per le eventuali copie di sicurezza delle versioni precedenti. Se non viene specificato con questa opzione, si utilizza il simbolo contenuto nella variabile di ambiente ***SIMPLE_BACKUP_SUFFIX***. Se anche questa variabile non è stata predisposta, si utilizza il simbolo tilde ('**~**').
`-V` ***tipo_di_backup*** `--version-control=`***tipo_di_backup***	Permette di definire esplicitamente il modo con cui gestire le copie di sicurezza delle versioni precedenti, quando si usa anche l'opzione '**-b**'. Per la precisione cambia il tipo di estensione che viene aggiunto ai file: con '**t**' o '**numbered**' le copie di sicurezza hanno un'estensione numerata; con '**nil**' o '**existing**' si mantengono le copie di sicurezza solo per i file che hanno già una o più copie di sicurezza numerate; con '**never**' o '**simple**' si ottiene una copia di sicurezza semplice, costituita da una sola copia. Se questa opzione non viene indicata, si prende in considerazione il valore della variabile di ambiente ***VERSION_CONTROL***.

Tabella 20.71. Variabili di ambiente

Variabile	Descrizione
`VERSION_CONTROL`	Permette di definire la modalità di gestione delle copie di sicurezza delle versioni precedenti in modo predefinito. I valori attribuibili a questa variabile sono gli stessi utilizzati come argomento dell'opzione '**-V**'.
`SIMPLE_BACKUP_SUFFIX`	Definisce il simbolo da utilizzare come suffisso per i nomi dei file che rappresentano le copie di sicurezza.

Segue la descrizione di alcuni esempi.

- `$ cp -R /test/* ~/prova` [*Invio*]

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

- `$ cp -R /test ~/prova` [*Invio*]

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

- `$ cp --parents aa/bb/cc miadir` [*Invio*]

  Copia il file '`aa/bb/cc`' in modo da ottenere '`miadir/aa/bb/cc`'. Le directory intermedie, eventualmente mancanti, vengono create.

- `$ cp --no-dereference /test/* ~/prova` [*Invio*]

  Copia il contenuto della directory '`/test/`' in '`~/prova/`', riproducendo i collegamenti simbolici così come sono nell'origine.

È molto importante comprendere la differenza tra le opzioni '**`-r`**' e '**`-R`**'. La seconda è quella che rappresenta meglio la ricorsività della copia, perché la prima tratta i file FIFO e alcuni file speciali come file normali. Per comprendere la cosa, si può procedere con l'esempio seguente, in cui si crea prima una directory, all'interno della quale si inserisce un file FIFO:

```
$ mkdir prova_1 [Invio]

$ mknod coda p [Invio]
```

In questo modo, si ottiene il file FIFO '`prova_1/coda`'. Volendo copiare la directory '`prova_1`' in '`prova_2`', includendo il file FIFO '`coda`', si deve procedere nel modo seguente:

```
$ cp -R prova_1 prova_2 [Invio]
```

Al contrario, utilizzando l'opzione '**`-r`**', la copia non si concluderebbe, perché questa resterebbe in attesa di dati dal file '`prova_1/coda`', come se fosse un file normale, per generare un file '`prova_2/coda`' (di tipo normale), con tale contenuto.

20.11.6 Utilizzo di «ln»

Il programma di servizio '**`ln`**'[23] (*link*) crea un collegamento tra file o tra directory:

```
ln [opzioni] origine destinazione
```

```
ln [opzioni] origine... directory
```

Se viene specificata un'origine e una destinazione, questa ultima diventa il nuovo collegamento che punta al nome indicato come origine (e può trattarsi anche di una directory). Se vengono specificati più nomi nell'origine, l'ultimo argomento deve essere una directory e si intende che al suo interno devono essere creati tanti collegamenti quanti sono i nomi indicati come origine. Se non viene specificato diversamente attraverso le opzioni, vengono creati dei collegamenti fisici e non dei collegamenti simbolici.

Il programma '**`ln`**' utilizza le variabili di ambiente ***VERSION_CONTROL*** e ***SIMPLE_BACKUP_SUFFIX*** nello stesso modo di '**`cp`**'.

Tabella 20.72. Alcune opzioni della versione GNU di '**`ln`**'. Le opzioni conformi allo standard POSIX sono evidenziate.

Opzione	Descrizione

`-b`
`--backup`
`-f`
`--force`
`-i`
`--interactive`
`-S`
`--suffix`
`-V`
`--version-control`
	Queste opzioni funzionano nello stesso modo di '**`cp`**'.

`-s` `--symbolic-link`	Crea un collegamento simbolico invece di creare un collegamento fisico.

Opzione	Descrizione

`-d`
`-F`
`--directory`
	Permette all'utente '**`root`**' di creare un collegamento fisico per una directory, ma questa operazione potrebbe essere impedita poi dal kernel.

`-n` `--no-dereference`	Quando la destinazione corrisponde a un collegamento simbolico preesistente che punta verso una directory, il funzionamento normale prevederebbe la creazione del collegamento in quella directory. Usando questa opzione si intende evitare ciò, rimpiazzando quel collegamento simbolico. Per poter attuare in pratica la cosa, occorre anche utilizzare l'opzione '**`-f`**'.

Segue la descrizione di alcuni esempi.

- `$ ln -s /bin/ls ~/elenco` [*Invio*]

  Crea il collegamento simbolico '`elenco`', all'interno della directory personale, il quale punta a '`/bin/ls`'. Eseguendo il file '`~/elenco`' si ottiene in pratica di eseguire il comando '**`ls`**'.

- `$ ln /bin/ls ~/elenco` [*Invio*]

  Crea il collegamento fisico '`elenco`', all'interno della directory personale, il quale punta a '`/bin/ls`'. Eseguendo il file '**`~/elenco`**' si ottiene in pratica di eseguire il comando '**`ls`**'.

- `$ ln -s /bin/* ~/` [*Invio*]

  Crea dei collegamenti simbolici all'interno della directory personale per tutti i file contenuti in '`/bin`'. Per ogni collegamento simbolico che viene creato, il percorso di questo risulta assoluto e inizia con '`/bin/`'.

- `$ cd /bin ; ln -s * ~/` [*Invio*]

  In questo esempio, rispetto a quanto mostrato in quello precedente, il comando di creazione dei collegamenti simbolici viene dato nel momento in cui ci si trova nella directory '`/bin/`', in riferimento a tutti i file della stessa. Quello che si ottiene nella directory personale dell'utente è la creazione di collegamenti simbolici diretti a loro stessi e **perfettamente inutili**.

- `$ ln -s /bin ~/binari` [*Invio*]

  Crea il collegamento simbolico '`~/binari`' alla directory '`/bin/`'. In questo modo, eseguendo '**`cd ~/binari`**' ci si ritrova in pratica in '`/bin/`'.

20.11.7 Utilizzo di «install»

Il programma di servizio '**`install`**'[24] copia i file attribuendo i permessi e le proprietà stabilite:

```
install [opzioni] origine... destinazione
```

```
install [opzioni] -d directory...
```

In pratica, si comporta in modo simile a '**`cp`**' con in più la possibilità di definire gli attributi dopo la copia e di creare tutte le directory necessarie. È usato tipicamente per l'installazione di programmi.

Tabella 20.73. Alcune opzioni di '**`install`**' nella versione GNU. Benché si tratti di un programma di uso comune, non fa parte dello standard POSIX.

Opzione	Descrizione

`-b`
`--backup`
`-S`
`--suffix`
`-V`
`--version-control`
	Queste opzioni funzionano nello stesso modo di '**cp**'.

`-d` ***directory***… `--directory=`***directory***…	Crea le directory indicate, definisce l'utente proprietario, il gruppo proprietario e i permessi in base alle altre opzioni.
`-g` ***gruppo*** `--group=`***gruppo***	Definisce il gruppo proprietario dei file installati o delle directory.
`-m` ***modalità*** `--mode=`***modalità***	Definisce i permessi in modo analogo alla sintassi di '**chmod**' (20.7.3).
`-o` ***proprietario*** `--owner=`***proprietario***	Definisce l'utente proprietario dei file installati o delle directory.

20.11.8 Utilizzo di «dd»

Il programma di servizio '**dd**'[25] (*data duplicator* o *data dump*) è un programma di copia a basso livello:

`dd` [***opzioni***]

Le opzioni sono definite in modo strano rispetto ai normali programmi di servizio Unix, in quanto non sono prefissate dal solito trattino ('**-**'). Se tra le opzioni non vengono definiti i file di input o di output, si usano rispettivamente lo standard input e lo standard output.

Molte delle opzioni utilizzano un argomento numerico. Questi argomenti numerici possono essere indicati anche con l'ausilio di moltiplicatori posti subito dopo il numero stesso. La tabellina seguente riepiloga quelli di uso più comune nella versione GNU di '**dd**'; tenendo conto che sono disponibili altri moltiplicatori molto più grandi. I casi previsti anche dallo standard POSIX sono evidenziati:

Modello	Valore corrispondente in byte	Modello	Valore corrispondente in byte
n`c`	***n***	***n***`w`	***n*** * 2
n`b`	***n*** * 512	***n***`x`***m***	***n*** * ***m***
n`kB`	***n*** * 1000	***n***`k` ***n***`K`	***n*** * 1024
n`MB`	***n*** * 1000^2	***n***`M`	***n*** * 1024^2
n`GB`	***n*** * 1000^3	***n***`G`	***n*** * 1024^3
n`TB`	***n*** * 1000^4	***n***`T`	***n*** * 1024^4

Tabella 20.75. Alcune opzioni, secondo lo standard POSIX.

Opzione	Descrizione
`if=`***file***	*input file* Legge i dati dal file indicato invece che dallo standard input.

Opzione	Descrizione
`of=`***file***	*output 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. Ma la regola non vale più se si utilizza un tipo di conversione '**notrunc**' (viene descritto in un'altra tabella).
`ibs=`***numero_di_byte***	*input block size* Legge a blocchi di byte della quantità indicata dall'argomento.
`obs=`***numero_di_byte***	*output block size* Scrive a blocchi di byte della quantità indicata dall'argomento.
`bs=`***numero_di_byte***	*block size* 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***	*conversion block size* Definisce la dimensione della memoria di conversione (*buffer*). In pratica determina la dimensione del blocco da utilizzare quando si devono effettuare delle conversioni nella codifica. Più avanti viene mostrato 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*>, pari a 00_{16}).
`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 si saltino delle porzioni con l'opzione '**skip**').

Opzione	Descrizione
`conv=`*conversione* [*, conversione*]…	Permette di definire il tipo di conversione, anche attraverso passaggi successivi. Il tipo di conversione viene specificato con il nome che lo identifica. Se si intendono applicare passaggi successivi, i tipi di conversione si separano con una virgola senza spazi prima o dopo la stessa.

Tabella 20.76. Tipi di conversione conformi allo standard POSIX.

Tipo	Descrizione
`ascii`	Converte dalla codifica EBCDIC a ASCII.
`ebcdic`	Converte dalla codifica ASCII a EBCDIC.
`ibm`	Converte dalla codifica ASCII-IBM a EBCDIC.
`block`	Tratta le righe di ingresso come record terminati dal codice di interruzione di riga. Questi record vengono troncati o allungati in modo da corrispondere alla dimensione indicata attraverso l'opzione '**cbs**'. Alla fine, i codici di interruzione di riga risultano trasformati in spazi normali (<*SP*>), a meno che i record non siano stati troncati prima; inoltre, se si è reso necessario un allungamento dei record, è sempre il carattere spazio a essere aggiunto. In pratica, il risultato finale è quello di un file con i record di dimensione uguale e per questo senza più alcuna terminazione attraverso codici di interruzione di riga.
`unblock`	Esegue l'operazione opposta di '**block**': il file in ingresso viene letto a blocchi di dimensione stabilita attraverso l'opzione '**cbs**' e gli spazi finali di ogni blocco vengono sostituiti con il codice di interruzione di riga.
`lcase`	*lower case* Trasforma le lettere maiuscole in minuscole.
`ucase`	*upper case* Trasforma le lettere minuscole in maiuscole.
`swab`	*swap byte* Scambia le coppie di byte: ciò può essere utile quando i dati in questione sono interi a 16 bit da trasformare in, o da, una piattaforma Intel. (Nelle piattaforme Intel, gli interi a 16 bit sono scritti in modo da invertire la sequenza normale dei due byte che si utilizzano).
`noerror`	Nel caso si verifichi un errore di lettura, continua ugualmente l'operazione.
`notrunc`	Il file in uscita non viene troncato. Questo argomento è utile nel caso si scriva su file già esistenti: se dopo la trasformazione che si fa, la dimensione dei dati in uscita è inferiore a quella che ha già il file su cui si scrive, i dati rimanenti si lasciano come sono senza ridurre la dimensione di questo file.
`sync`	Aggiusta la lunghezza di ogni blocco in ingresso, aggiungendo eventualmente il carattere <*NUL*> (00_{16}), in modo che la sua dimensione sia uguale a quanto stabilito attraverso l'opzione '**ibs**'.

Il programma '**dd**' viene usato normalmente per riprodurre le immagini di unità di memorizzazione di massa, anche se nella maggior parte dei casi, con un sistema GNU/Linux è sufficiente usare '**cp**'. A questo proposito, vengono mostrati alcuni esempi riferiti ai vecchi

dischetti da 1440 Kibyte.

- `# dd if=disk.img of=/dev/fd0` [*Invio*]

  In questo caso si trasferisce semplicemente il file '`disk.img`' nel dischetto (inizializzato precedentemente a bassi livello). 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` [*Invio*]

  Rispetto all'esempio precedente, si immagina di avere a disposizione un dischetto da 1440 Kibyte (e naturalmente che il file-immagine sia adatto a questo tipo di dischetto). Un dischetto da 90 mm (3,5 pollici) con questo formato era composto da cilindri contenenti 18 + 18 settori di 512 Kibyte: 2 * 18 * 512 = 18 Kibyte. Specificando l'opzione '**obs=18k**' si intende fare in modo che '**dd**' fornisca al dispositivo '`/dev/fd0`' blocchi di quella dimensione per facilitare l'operazione di scrittura.

- `# dd if=disk.img of=/dev/fd0 obs=18k count=80` [*Invio*]

  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.

20.11.9 Utilizzo di «ddrescue»

Il programma di servizio '**ddrescue**'[26] è analogo a '**dd**', con la differenza fondamentale che la copia non si arresta in presenza di errori di lettura, pertanto viene usato principalmente per il recupero dei dati da unità di memorizzazione difettose:

```
ddrescue [opzioni] file_in_ingresso file_in_uscita [registro]
```

La sintassi per l'uso di questo programma è diversa rispetto al '**dd**' tradizionale, in quanto utilizza opzioni precedute da trattini, come avviene di solito, inoltre prevede la possibilità di associare un file per annotare l'esito del processo di recupero dati (l'ultimo nome opzionale che appare nella riga di comando).

Tabella 20.77. Alcune opzioni per l'uso di '**ddrescue**'.

Opzione	Descrizione
`-b` ***n_byte*** `--block-size=`***n_byte***	Dichiara la dimensione dei blocchi da usare in fase di lettura. Il valore predefinito è di 512 byte per le unità di memorizzazione normali e 2048 byte per i dischi ottici.
`-c` ***n*** `--cluster-size=`***n***	Richiede la lettura simultanea di ***n*** blocchi (***n*** volte la dimensione del blocco stabilita con l'opzione '**-b**'). Per esempio, per la lettura di dischetti convengono blocchi da 18 unità, in quanto un dischetto comune è composto da tracce da 18 settori (i settori, in questo caso, sono da 512 byte). Il valore predefinito di questi blocchi di lettura è complessivamente di 64 Kibyte, ovvero 64 Kibyte diviso il valore specificato con l'opzione '**-b**'.
`-i` ***n*** `--input-position=`***n***	Richiede di iniziare la lettura a partire dal byte ***n***; se non viene specificato, si intende che l'inizio debba avvenire dalla posizione zero.
`-o` ***n*** `--output-position=`***n***	Richiede di iniziare la scrittura a partire dal byte ***n*** nel file di destinazione; se non viene specificato, si intende lo stesso valore dell'opzione '**-i**'.
`-r` ***n*** `--max-retries=`***n***	Richiede di ritentare ***n*** volte (oltre la prima lettura), se si presenta un errore. Il valore predefinito per questa opzione è zero, nel senso che si tenta una sola lettura; se si usa il valore `-1`, si intende richiedere una quantità infinita di tentativi.

Gli argomenti delle opzioni che rappresentano una quantità di byte,

come nel caso di '**-c**', '**-i**' e '**-o**', consentono l'uso di suffissi che moltiplicano il valore numerico rappresentato per un valore costante dato, in un modo simile a quello che avviene per il programma '**dd**'; tuttavia, le sigle usate non sono sempre compatibili.

Tabella 20.78. Rappresentazione di valori numerici che esprimono una quantità di byte, con l'ausilio di moltiplicatori, nelle opzioni di '**ddrescue**' (nella versione GNU). Sono disponibili anche altri moltiplicatori, per valori molto più grandi.

Modello	Valore corrispondente in byte	Modello	Valore corrispondente in byte
		n`b`	***n*** blocchi fisici, in base alle caratteristiche dell'unità di memorizzazione.
n`k`	***n*** * 1000	***n***`Ki`	***n*** * 1024
n`M`	***n*** * 1000^2	***n***`Mi`	***n*** * 1024^2
n`G`	***n*** * 1000^3	***n***`Gi`	***n*** * 1024^3
n`T`	***n*** * 1000^4	***n***`Ti`	***n*** * 1024^4

Segue la descrizione di alcuni esempi.

- `# ddrescue /dev/sdc immagine registro.log` [*Invio*]

  Legge il contenuto di '`/dev/sdc`', per produrre il file '`immagine`', memorizzando le annotazioni sul procedimento nel file '`registro.log`'.

- `# ddrescue -r 3 /dev/sdc immagine registro.log` [*Invio*]

  Come nell'esempio precedente, legge il contenuto di '`/dev/sdc`', per produrre il file '`immagine`', memorizzando le annotazioni sul procedimento nel file '`registro.log`'. Ma in più si richiede che i tentativi di lettura in caso di errori vengano ritentati per tre volte.

Quando si dispone di più copie di unità di memorizzazione identiche (per esempio dei dischi ottici) che risultano danneggiate in modo differente, è possibile riavviare il procedimento di recupero facendo riferimento allo stesso file di destinazione e lo stesso file per le annotazioni. Il programma '**ddrescue**', nei tentativi successivi, si occupa di leggere solo le porzioni di dati che in quelli precedenti non hanno prodotto un risultato soddisfacente.

Quando si vuole recuperare il contenuto di una partizione di un'unità di memorizzazione comune, occorre tenere presente che questa non deve essere innestata, o almeno occorre avere l'accortezza di utilizzarla in sola lettura. Dopo il recupero in un file-immagine, si procede normalmente con programmi come '**fsck**' per cercare di sistemare in qualche modo il file-immagine stesso, prima di riutilizzarlo.

Altri programmi affini.

sdd(1)[27]	Si tratta di un programma simile, dal punto di vista funzionale, a '**dd**', adatto in particolare al trasferimento integrale di dischi interi.
cstream(1)[28]	Si tratta di un programma simile, dal punto di vista funzionale, a '**dd**', ma con una sintassi completamente diversa, che ha in più la capacità di limitare il flusso di dati a un valore stabilito (per flusso di dati si intende la quantità di dati trasmessa per unità di tempo).

20.12 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; inoltre, occorre tenere in considerazione il problema quando si valuta l'eventuale pericolosità di questo tipo di operazione: cambiare

nome a un file in modo errato può provocare la sovrascrittura di un altro.

La cancellazione è sempre l'operazione più pericolosa. Nei file system Second-extended non è facile recuperare i dati cancellati. Piuttosto di cancellare, sarebbe meno pericoloso spostare temporaneamente i file in una directory che funge da cestino. Nella sezione 20.12.3 viene mostrato uno script in grado di gestire agevolmente una sorta di cestino del genere.

La cancellazione si ottiene normalmente eliminando la voce relativa all'interno di una directory e liberando l'inode corrispondente, mentre l'informazione rimane presente fino a quando lo stesso spazio viene riutilizzato nel disco. Quando si devono cancellare anche le tracce residue di un'informazione, occorre usare strumenti appositi che, prima di cancellare un file, lo sovrascrivono varie volte con dati più o meno casuali. Tuttavia, questa tecnica non funziona se la memoria di massa è organizzata in modo diverso dal solito per qualche fine, come può essere un sistema ridondante, un sistema che mantiene lo storico di tutti i cambiamenti apportati al file system o un sistema che si avvale della rete per la condivisione degli stessi dati.

20.12.1 Utilizzo di «mv»

Il programma di servizio '**mv**'[29] (*move*) sposta i file o le directory:

`mv` [***opzioni***] ***origine*** … ***destinazione***

Se vengono specificati solo i nomi di due elementi (file o directory), il primo viene spostato o rinominato in modo da ottenere quanto indicato come destinazione. Se vengono indicati più elementi (file o directory), l'ultimo argomento **deve** essere una directory, all'interno della quale vengono spostati tutti gli elementi elencati. Nel caso di spostamenti attraverso diversi file system, vengono spostati solo i file normali, quindi: né collegamenti, né directory.

Il programma '**mv**' può essere pericoloso perché può sovrascrivere altri file senza preavviso. Per ridurre le possibilità di errori, conviene creare un alias in modo che '**mv**' funzioni sempre con l'opzione '**-i**'. Se poi si ha la necessità di sovrascrivere i file di destinazione, si può sempre utilizzare l'opzione '**-f**'.

Tabella 20.80. Alcune opzioni di '**mv**', nella versione GNU. Le opzioni conformi allo standard POSIX sono evidenziate.

Opzione	Descrizione

`-b`
`--backup`
`-f`
`--force`
`-i`
`--interactive`
`-S`
`--suffix`
`-v`
`--version-control`
	Queste opzioni funzionano nello stesso modo di '**cp**'.

20.12.2 Utilizzo di «rm»

Il programma di servizio '**rm**'[30] (*remove*) rimuove i file indicati come argomento:

`rm` [***opzioni***] ***nome*** …

In mancanza dell'indicazione delle opzioni necessarie, non vengono rimosse le directory.

Tabella 20.81. Alcune opzioni di '**rm**', nella versione GNU. Le opzioni conformi allo standard POSIX sono evidenziate.

Opzione	Descrizione

`-r`
`-R`
`--recursive`
	Rimuove il contenuto delle directory in modo ricorsivo.

`-i` `--interactive`	Chiede una conferma esplicita per la cancellazione di ogni file.
`-d` `--directory`	Elimina le directory trattandole come se fossero dei file normali. In pratica, i file e le altre directory che dovessero eventualmente essere contenuti, non vengono rimossi prima: viene semplicemente interrotto il loro collegamento. L'operazione può essere pericolosa perché ci potrebbero essere dei file aperti al di sotto di queste directory che si rimuovono, senza che tale possibilità venga verificata. Inoltre, dopo un'azione di questo tipo, il file system 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.

Segue la descrizione di alcuni esempi.

- `$ rm prova` [*Invio*]

  Elimina il file '`prova`'.

- `$ rm ./-r` [*Invio*]

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

- `$ rm -r ~/varie` [*Invio*]

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

Il programma '**`rm`**' è pericolosissimo perché è potente e irreversibile. Gli errori più frequenti e disastrosi, sono causati da sbagli nella digitazione dei comandi o da cattiva valutazione dell'effetto di uno di questi. Ci sono tre cose da fare per ridurre i rischi di disastri:

- evitare il più possibile di accedere come utente '**`root`**';
- controllare il comando che si vuole eseguire;
- creare un alias in modo che '**`rm`**' funzioni sempre con l'opzione '**`-i`**'.

Gli errori più frequenti **da evitare** sono descritti negli esempi seguenti.

- `$ rm prova *` [*Invio*]

  Probabilmente, l'intenzione è quella di eliminare solo i file che iniziano con la parola '**`prova`**', mentre in realtà viene inserito uno spazio involontario tra '**`prova`**' e l'asterisco. In tal modo, prima viene cancellato il file '`prova`' e subito dopo tutto quello che si trova nella directory corrente.

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

  L'intenzione sembra essere quella di eliminare tutti i file e le directory nascoste (tutto ciò che inizia con un punto) contenute nella directory corrente. In realtà si individuano sì i file e le directory nascoste, ma con esse anche la directory corrente ('`.`') e quella precedente, ovvero la genitrice ('`..`')! A meno che il programma '**`rm`**' preveda espressamente questi casi in modo da evitare danni, si può ottenere la rimozione della directory corrente e anche della genitrice stessa.

  Si osservi che spesso la directory personale dell'utente '**`root`**' è '`/root/`'; se il comando appena mostrato viene dato da tale posizione, il cancellare la directory precedente (con tutto il suo contenuto), significa fare riferimento alla directory radice e quindi a **tutto il file system**.

20.12.3 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 una sorta di cestino:

```
#!/bin/sh
# ricicla FILE...
# nome_completo NAME

function nome_completo () {
    local primo_carattere
    local comando_sed
    comando_sed="sed -n s/^\\(.\\).*/\\1/p"
    primo_carattere=‘echo $1 | $comando_sed‘
    if [ $primo_carattere == "/" ]
    then
        # Si tratta di un percorso assoluto.
        echo $1
    else
        # È necessario aggiungere il percorso precedente.
        echo ‘pwd‘/$1
    fi
}
# Crea una directory temporanea da usare come cestino.
CESTINO="/var/tmp/cestino"
DATA=$(date +%Y%m%d%H%M%S)
NOME_COMPLETO=""
mkdir $CESTINO 2> /dev/null
# Verifica.
if ! touch $CESTINO/$DATA
then
    # Non è stato possibile creare il cestino: forse ci sono
    # problemi di permessi.
    echo "Non è possibile accedere alla directory"
    echo "$CESTINO"
    # Lo script termina restituendo un valore falso.
    exit 1
else
    # Elimina il file di prova all’interno del cestino.
    rm -f $CESTINO/$DATA
fi
# Copia all’interno del cestino.
for filename in $@
do
    # Trova il percorso assoluto.
    NOME_COMPLETO=‘nome_completo $filename‘
    # Sposta il file o la directory.
    if cp -dpRfv --parents $NOME_COMPLETO $CESTINO
    then
        rm -rf $NOME_COMPLETO
    fi
done
```

L'esempio mostrato riproduce il percorso in cui si trovano i file a partire dalla directory che svolge il ruolo di cestino. In questo modo si possono creare delle collisioni, per esempio quando si tenta di sovrascrivere una directory con un file. Se si preferisce si può usare la versione alternativa che viene mostrata nel seguito, in cui l'inizio del percorso del cestino è ottenuto da una sottodirectory composta dalla data e dall'ora in cui è stata svolta l'operazione:

```
#!/bin/sh
# ricicla FILE...
# nome_completo NAME

function nome_completo () {
```

```
    local primo_carattere
    local comando_sed
    comando_sed="sed -n s/^\\(.\\).*/\\1/p"
    primo_carattere=‘echo $1 | $comando_sed‘
    if [ $primo_carattere == "/" ]
    then
        # Si tratta di un percorso assoluto.
        echo $1
    else
        # È necessario aggiungere il percorso precedente.
        echo ‘pwd‘/$1
    fi
}
# Crea una directory temporanea da usare come cestino.
CESTINO="/var/tmp/cestino"
DATA=$(date +%Y%m%d%H%M%S)
NOME_COMPLETO=""
mkdir $CESTINO 2> /dev/null
# Verifica.
if ! touch $CESTINO/$DATA
then
    # Non è stato possibile creare il cestino: forse ci sono
    # problemi di permessi.
    echo "Non è possibile accedere alla directory"
    echo "$CESTINO"
    # Lo script termina restituendo un valore falso.
    exit 1
else
    # Elimina il file di prova all’interno del cestino.
    rm -f $CESTINO/$DATA
fi
# Fa in modo che il cestino abbia una sottodirectory
# diversa ogni volta.
CESTINO="$CESTINO/$DATA"
mkdir $CESTINO 2> /dev/null
# Copia all’interno del cestino.
for filename in $@
do
    # Trova il percorso assoluto.
    NOME_COMPLETO=‘nome_completo $filename‘
    # Sposta il file o la directory.
    if cp -dpRfv --parents $NOME_COMPLETO $CESTINO
    then
        rm -rf $NOME_COMPLETO
    fi
done
```

20.12.4 Utilizzo di «unlink»

Il programma di servizio ‘**unlink**’[31] è probabilmente qualcosa di superfluo, ma esiste nei sistemi Unix, pertanto è incluso anche nei sistemi GNU. Il suo scopo è quello di cancellare il riferimento a un file, cosa che equivale alla cancellazione; tuttavia, in alcuni sistemi, questo programma può anche cancellare il riferimento a una directory, cosa che però non corrisponde alla cancellazione del suo contenuto, il quale continua così a esistere come inode, anche se può essere irraggiungibile attraverso un percorso. Per motivi di sicurezza, nei sistemi GNU, ‘**unlink**’ non può essere usato con le directory.

```
unlink file…
```

20.12.5 Utilizzo di «shred»

Il programma di servizio ‘**shred**’[32] serve a cancellare il contenuto di un file, in modo da renderne impossibile il recupero. Il programma ‘**shred**’ esegue questo compito sovrascrivendo più volte i dati del file, con dati più o meno casuali:

```
shred [file] file…
```

In condizioni normali, quando non si specifica qualcosa di diverso attraverso le opzioni, il file viene sovrascritto fino alla fine dell’ultimo blocco che lo contiene. Per blocco si intende l’unità minima di dati in base al contesto, la quale può essere il blocco di dati del file system o il blocco dell’unità a blocchi, se si fa riferimento a file di dispositivo il cui accesso è regolato in questo modo.

Dal momento che si può indicare un file su disco o un file di dispositivo, il programma non cancella il file alla fine della sovrascrittura, a meno che sia richiesto espressamente con le opzioni.

Per approfondire l’uso del programma, si può consultare la documentazione originale: *info shred* oppure la pagina di manuale *shred(1)*. Si ricordi anche di verificare se, in base al proprio contesto operativo, l’uso del programma può produrre effettivamente il risultato atteso.

20.13 Archiviazione

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 file system. Per lo stesso motivo, l’archiviazione è il modo con cui si possono trasferire agevolmente i dati attraverso piattaforme differenti. L’archiviazione pura e semplice non ottiene alcun risparmio nello spazio utilizzato dai dati; per poter ottenere invece un compattamento, si deve utilizzare la compressione.

L’archiviazione pura e semplice è ottenuta normalmente attraverso il programma ‘**tar**’ o il programma ‘**cpio**’. Questi due sono equivalenti, almeno a livello teorico. In pratica, è l’utilizzatore che sceglie quello che per qualche motivo gli è più simpatico, specializzandosi nell’uso delle sue opzioni particolari. Va però osservato che lo standard POSIX prevede solo il programma ‘**pax**’, anche se questo dovrebbe essere in grado di elaborare i vecchi formati di Cpio e Tar.

20.13.1 Utilizzo di Cpio

Il programma di servizio ‘**cpio**’[33] copia file da e verso archivi ‘**cpio**’ o ‘**tar**’:

```
cpio -o [opzioni] [< elenco_nomi] [> archivio]
```

```
cpio -i [opzioni] [modello] [< archivio]
```

```
cpio -p [opzioni] directory_di_destinazione [< elenco_nomi]
```

L’archivio può essere un file su disco, un nastro magnetico o il flusso di un condotto. Le tre sintassi indicate rappresentano le tre modalità operative del comando.

- *copy-out* (archiviazione)
  Dallo standard input viene letto un elenco di nomi di file (uno per riga) e l’archivio di questi file viene generato ed emesso attraverso lo standard output.
- *copy-in* (lettura di un archivio)
  Dallo standard input viene letto il contenuto di un archivio dal quale si possono estrarre i file in esso contenuti.
- *copy-pass* (copia)
  Dallo standard input viene letto un elenco di nomi di file (uno per riga) e questi file (con il loro contenuto) vengono copiati nella directory di destinazione.

Si può consultare il documento *info cpio*, oppure la pagina di manuale *cpio(1)* per maggiori dettagli sull’uso di questo programma.

Tabella 20.84. Alcune opzioni per il «copy-out».

Opzione	Descrizione
`-o` `--create`	Funziona in modalità *copy-out*.

Opzione	Descrizione
`-A` `--append`	Aggiunge dati a un archivio esistente che deve essere specificato con l'opzione '**`-O`**'.
`-L` `--dereference`	Quando incontra dei collegamenti simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i collegamenti.
`-O` ***nome_archivio***	Specifica il nome dell'archivio da creare o incrementare, invece di utilizzare lo standard output.

Tabella 20.85. Alcune opzioni per il «copy-in».

Opzione	Descrizione
`-i` `--extract`	Funziona in modalità *copy-in*.
`-d` `--make-directories`	Crea le directory necessarie.
`-E` ***file*** `--pattern-file=`***file***	Legge il modello che esprime i nomi dei file da estrarre, o l'elenco dei nomi stessi, dal file indicato come argomento dell'opzione.
`-f` `--nomatching`	Copia soltanto i file che non corrispondono al modello indicato.
`-I` ***archivio***	Permette di specificare il nome dell'archivio da usare, invece di riceverlo dallo standard input.
`-t` `--list`	Elenca il contenuto dell'archivio.

Tabella 20.86. Alcune opzioni per il «copy-pass».

Opzione	Descrizione
`-p` `--pass-through`	Funziona in modalità *copy-pass*.
`-d` `--make-directories`	Crea le directory necessarie.
`-l` `--link`	Se possibile, crea dei collegamenti invece di copiare i file.
`-L` `--dereference`	Quando incontra dei collegamenti simbolici, copia i file a cui questi puntano, invece di copiare semplicemente i collegamenti.

Segue la descrizione di alcuni esempi:

- `$` **`cpio -o < elenco > /tmp/archivio.cpio`** [*Invio*]

  Archivia i file e le directory elencati nel file '`elenco`' generando il file '`/tmp/archivio.cpio`'.

- `$` **`ls *.sgml | cpio -o > /tmp/archivio.cpio`** [*Invio*]

  Archivia i file e le directory corrispondenti al modello '`*.sgml`' generando il file '`/tmp/archivio.cpio`'.

- `$` **`find lavoro -print | cpio -o > /tmp/archivio.cpio`** [*Invio*]

  Archivia la directory '`lavoro/`' e tutto il suo contenuto, generando il file '`/tmp/archivio.cpio`'.

- `$` **`cpio -i -t < /tmp/archivio.cpio`** [*Invio*]

  Elenca il contenuto dell'archivio '`/tmp/archivio.cpio`'.

- `$` **`cpio -i < /tmp/archivio.cpio`** [*Invio*]

  estrae l'archivio '`/tmp/archivio.cpio`' a partire dalla directory corrente.

- `$` **`cat elenco | cpio -pd /tmp/prova`** [*Invio*]

  Copia i file e le directory elencati nel file '`elenco`', a partire dalla directory '`/tmp/prova/`'.

20.13.2 Utilizzo di Tar

Il programma di servizio '**`tar`**' [34] (*Tape archive*) è un programma di archiviazione nato originariamente per essere usato con i nastri:

```
tar opzione_di_funzionamento [opzioni] file...
```

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 con le opzioni, l'archivio viene emesso attraverso lo standard output.

Il '**`tar`**' tradizionale ammette l'uso di opzioni senza il trattino anteriore ('**`-`**') consueto. Questa tradizione è stata mantenuta anche nel '**`tar`**' GNU a cui si vuole fare riferimento qui, ma tale forma deve essere usata consapevolmente e con prudenza. Negli esempi viene mostrato in che modo potrebbero essere usate tali opzioni senza trattino.

Per la descrizione completa di questo programma, conviene consultare il documento *info tar*, oppure la pagina di manuale *tar(1)*.

Tabella 20.87. Opzioni di funzionamento per rappresentare l'operazione da compiere. Di queste, può e deve esserne utilizzata una sola. Di solito, data la loro importanza, queste opzioni appaiono all'inizio degli argomenti di '**`tar`**'.

Opzione	Descrizione

`A`
`-A`
`--catenate`
`--concatenate`
	Aggiunge dei file '**`tar`**' a un archivio già esistente.

`c`
`-c`
`--create`
	Crea un nuovo archivio.

`d`
`-d`
`--diff`
`--compare`
	Trova le differenze tra l'archivio e i file esistenti effettivamente.

`--delete`	Cancella dall'archivio i file indicati. Non può essere usato per un archivio su nastro.

`r`
`-r`
`--append`
	Aggiunge dati a un archivio già esistente.

`t`
`-t`
`--list`
	Elenca il contenuto di un archivio.

`u`
`-u`
`--update`
	Aggiunge solo i file più recenti rispetto a quanto già contenuto nell'archivio.

Opzione	Descrizione

`x`
`-x`
`--extract`
`--get`
	Estrae i file da un archivio.

Tabella 20.88. Altre opzioni.

Opzione	Descrizione
`--atime-preserve`	Fa in modo che la data di accesso dei file che vengono archiviati non venga modificata.
`-f` ***file*** `--file=`***file***	Emette l'archivio nel file o nel dispositivo. Se si tratta di un file normale, questo viene creato.
`-h` `--dereference`	Non copia i collegamenti simbolici, ma i file a cui questi fanno riferimento.
`-k` `--keep-old-files`	In fase di estrazione da un archivio, non sovrascrive i file eventualmente già esistenti.
`-l` `--one-file-system`	Quando viene creato un archivio, resta in un solo file system: quello di partenza.
`-L` ***Kibyte*** `--tape-length=`***Kibyte***	Definisce la dimensione massima dei vari segmenti di copia multivolume.
`-m` `--modification-time`	In fase di estrazione da un archivio, non viene ripristinata la data di modifica dei file.
`-M` `--multi-volume`	Permette di creare, elencare o estrarre, un archivio multivolume.
`-N` ***data*** `--after-date=`***data*** `--newer` ***data***	Archivia solo i file la cui data è più recente di quella indicata come argomento.
`-O` `--to-stdout`	Estrae i file emettendoli attraverso lo standard output.

`-p`
`--same-permissions`
`--preserve-permissions`
	Estrae tutti i permessi associati ai file. Se non viene usata questa opzione, i file ottengono i permessi predefiniti, anche in funzione della maschera dei permessi dell'utente che esegue l'operazione.

`-P` `--absolute-path`	Estrae i file utilizzando i percorsi assoluti, cioè senza eliminare la prima barra ('`/`') che appare nei nomi di percorso (*pathname*).
`--remove-files`	In fase di creazione di un nuovo archivio, elimina i file archiviati.
`--same-owner`	Durante l'estrazione da un archivio, assegna ai file estratti gli utenti e i gruppi proprietari originali.
`-v` `--verbose`	Elenca i file che vengono elaborati.
`-W` `--verify`	Cerca di verificare la validità dell'archivio dopo averlo creato.

Opzione	Descrizione

`-Z`
`--compress`
`--uncompress`
	Filtra l'archivio attraverso il programma di compressione '**`compress`**'.

`-z`
`--gzip`
`--ungzip`
	Filtra l'archivio attraverso il programma di compressione '**`gzip`**'.

`-j` `--bzip2`	Filtra l'archivio attraverso il programma di compressione '**`bzip2`**' (sia per la compressione che per l'estrazione).
`--use-compress-program` ***prog***	Filtra l'archivio attraverso il programma di compressione indicato nell'argomento. Questo programma di compressione deve riconoscere l'opzione '**`-d`**', come fa '**`gzip`**', allo scopo di decomprimere i dati.

Segue la descrizione di alcuni esempi.

- `# tar -c -f /dev/fd0 -L 1440 -M -v /usr` [*Invio*]

  Archivia la directory '`/usr/`' con tutto il suo contenuto, comprese le sottodirectory, utilizzando i dischetti (da 1440 Kibyte).

  Con la copia multivolume, come in questo caso, non è possibile utilizzare la compressione automatica attraverso l'opzione '**`-z`**' o '**`-Z`**'.

- `# tar cf /dev/fd0 -L 1440 -M -v /usr` [*Invio*]

  Esattamente come nell'esempio precedente, con la differenza che le opzioni '**`-c`**' e '**`-f`**' sono indicate senza il trattino iniziale.

- `# tar cvf /dev/fd0 -L 1440 -M /usr` [*Invio*]

  Esattamente come nell'esempio precedente.

- `# tar -cfv /dev/fd0 -L 1440 -M /usr` [*Invio*]

  `# tar cfv /dev/fd0 -L 1440 -M /usr` [*Invio*]

  Questi due esempi sono identici ed **errati** ugualmente. Non è possibile accodare lettere di altre opzioni dopo la «f», dal momento che questa richiede un argomento.

  In molti documenti su '**`tar`**' si vedono esempi errati di questo tipo. Possono anche funzionare, ma sono errati concettualmente ed è molto probabile incontrare un programma '**`tar`**' che in tali situazioni faccia qualcosa di diverso da quello che ci si aspetterebbe.

- `$ tar -t -f /dev/fd0 -L 1440 -M -v` [*Invio*]

  Visualizza l'elenco del contenuto dell'archivio fatto su dischetti.

- `$ tar -x -f /dev/fd0 -L 1440 -M -v -p --same-owner` [*Invio*]

  Estrae il contenuto dell'archivio su dischetti a partire dalla posizione corrente.

  È probabile che l'opzione '**`--same-owner`**' sia già predefinita all'interno di '**`tar`**', ma in generale vale la pena di ricordarsene. Tuttavia, in questi esempi, dal momento che si tratta di un utente comune (lo si vede dal dollaro che viene indicato come invito), non ha significato, dal momento che l'utente comune non ha la possibilità di assegnare a un altro la proprietà dei file che crea.

- `$ tar xpvf /dev/fd0 -L 1440 -M --same-owner` [*Invio*]

  Come nell'esempio precedente, aggregando alcune opzioni e togliendo il trattino iniziale di queste.

- `$ tar -c -f /tmp/archivio.tgz -z -v /usr` [*Invio*]

  Archivia il contenuto della directory '`/usr/`' nel file '`/tmp/archivio.tgz`' dopo averlo compresso con '**`gzip`**'.

- `$ tar czvf /tmp/archivio.tgz /usr` [*Invio*]

  Come nell'esempio precedente.

- `$ tar -c -f /tmp/archivio.tar.bz2 -j -v /usr` [*Invio*]

  Archivia il contenuto della directory '`/usr/`' nel file '`/tmp/archivio.tar.bz2`' dopo averlo compresso con '**`bzip2`**'.

- `$ tar cjvf /tmp/archivio.tar.bz2 /usr` [*Invio*]

  Come nell'esempio precedente.

20.14 Compressione

La compressione dei dati è una tecnica che consente di risparmiare senza perdere informazioni. L'operazione avviene di norma in modo sequenziale, per cui può essere gestita attraverso dei programmi filtro, che alle volte permettono di rendere trasparente l'operazione. Data la facilità con cui nei sistemi Unix si possono combinare assieme delle tecniche di questo genere, in tali ambienti si tende a preferire l'archiviazione seguita da una compressione complessiva.

20.14.1 Utilizzo di «gzip», «gunzip» e «zcat»

Il programma di servizio '**`gzip`**' [35] viene usato per comprimere o decomprimere ogni file indicato negli argomenti:

```
gzip [opzioni] [file...]
```

```
gunzip [opzioni] [file...]
```

```
zcat [opzioni] [file...]
```

Il programma '**`gzip`**' è in grado di comprimere solo file normali, in modo singolo: 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 trattino isolato ('`-`'), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.

Il nome '**`gunzip`**' è un collegamento a '**`gzip`**'. Se '**`gzip`**' viene avviato con il nome '**`gunzip`**' si comporta come se fosse stata utilizzata l'opzione '**`-d`**'.

Il nome '**`zcat`**' è un collegamento a '**`gzip`**'. Se '**`gzip`**' viene avviato con il nome '**`zcat`**' si comporta come se fossero state utilizzate simultaneamente le opzioni '**`-d`**' e '**`-c`**'. In alcuni sistemi, invece di '**`zcat`**' potrebbe essere presente il collegamento '**`gzcat`**'.

Si può consultare il documento *info gzip*, oppure la pagina di manuale *gzip(1)* per maggiori dettagli sull'uso di questo programma.

Tabella 20.89. Alcune opzioni.

Opzione	Descrizione

`-c`
`--stdout`
`--to-stdout`
	Emette il risultato attraverso lo standard output. Il programma utilizza automaticamente questa opzione quando viene eseguito con il nome '**`zcat`**'.

`-d`
`--decompress`
`--uncompress`
	Decomprime un file compresso. Il programma utilizza automaticamente questa opzione quando viene eseguito con il nome '**`gunzip`**'.

Opzione	Descrizione
`-r` `--recursive`	Se tra i nomi indicati nella riga di comando appaiono delle directory, vengono compressi o decompressi tutti i file in esse contenuti.
`-t` `--test`	Controlla l'integrità dei file compressi.

`-1`
`-2`
`-3`
`-4`
`-5`
`-6`
`-7`
`-8`
`-9`
	Permette di definire il livello di compressione: '**`-1`**' rappresenta la compressione minima, che in compenso richiede meno elaborazione; '**`-9`**' rappresenta la compressione massima, a scapito del tempo di elaborazione. Se non viene specificata questa opzione, si utilizza un livello intermedio, corrispondente a '**`-6`**'.

Segue la descrizione di alcuni esempi:

- `$ gzip *.sgml` [*Invio*]

  Comprime i file '`*.sgml`', utilizzando un livello intermedio di compressione, sostituendo i file originali con quelli compressi: '`*.sgml.gz`'.

- `$ gzip -d *.sgml.gz` [*Invio*]

  Espande i file corrispondenti al modello '`*.sgml.gz`', togliendo loro l'estensione '`.gz`'.

- `$ cat pippo | gzip -9 > pippo.gz` [*Invio*]

  Genera il file '`pippo.gz`' come risultato della compressione di '`pippo`'. In particolare, viene utilizzato il livello di compressione massima e il file originale non viene cancellato.

- `$ cat pippo.gz | gzip -d > pippo` [*Invio*]

  Fa l'opposto dell'esempio precedente: espande il file '`pippo.gz`' generando '`pippo`', senza cancellare il file originale.

20.14.2 Utilizzo di «bzip2» e «bunzip2»

Il programma di servizio '**`bzip2`**' [36] è un programma di compressione funzionalmente analogo a '**`gzip`**', nel senso che viene creato un file compresso per ogni file indicato negli argomenti:

```
bzip2 [opzioni] [file...]
```

```
bunzip2 [opzioni] [file...]
```

Il programma '**`bzip2`**', come '**`gzip`**', è in grado di comprimere solo file normali, in modo singolo, dove per ogni file ne viene generato un altro con l'estensione '`.bz2`'. Se non viene indicato alcun file o se si utilizza espressamente un solo trattino isolato ('`-`'), lo standard input viene compresso e il risultato viene emesso attraverso lo standard output.

Il programma '**`bzip2`**' utilizza un algoritmo di compressione differente, rispetto a '**`gzip`**', con un carico di elaborazione maggiore, che diventa efficace solo in presenza di file di grandi dimensioni. In generale, per garantire la massima portabilità di un archivio compresso, conviene utilizzare '**`gzip`**', salvo quando le dimensioni dell'archivio sono tali da rendere realmente conveniente l'utilizzo di '**`bzip2`**'.

La sintassi di '**`bzip2`**' è molto simile a quella di '**`gzip`**', anche se non è del tutto identica. Prima di decidere di utilizzare '**`bzip2`**' per

archiviare i propri dati, conviene leggere la documentazione originale: il documento *info bzip2*, oppure la pagina di manuale *bzip2(1)*, in modo da poter valutare correttamente.

Il nome '**bunzip2**' è un collegamento a '**bzip2**', il quale, se avviato con questo nome, utilizza implicitamente l'opzione '**-d**' per decomprimere i file indicati alla fine della riga di comando.

Tabella 20.90. Alcune opzioni.

Opzione	Descrizione
`-c` `--stdout`	Comprime o decomprime emettendo il risultato attraverso lo standard output. La decompressione ammette l'emissione di più file, mentre in caso di compressione, se ne può emettere solo uno.
`-d` `--decompress`	Forza la modalità di decompressione dei dati. Questo è il comportamento predefinito, quando il programma viene eseguito con il nome '**bunzip2**'.
`-f` `--compress`	Forza la modalità di compressione dei dati. Serve a imporre la compressione, indipendentemente dal nome utilizzato per avviare '**bzip2**'.
`-t` `--test`	Controlla l'integrità dei file compressi.

`-1`
`-2`
`-3`
`-4`
`-5`
`-6`
`-7`
`-8`
`-9`
	Permette di definire il livello di compressione: '**-1**' rappresenta la compressione minima, che in compenso richiede blocchi più piccoli (100 Kibyte) e meno elaborazione; '**-9**' rappresenta la compressione massima, a scapito della dimensione dei blocchi che aumenta in modo considerevole (900 Kibyte) e del tempo di elaborazione.

20.14.3 UPX

UPX, ovvero il programma di servizio '**upx**',[37] consente di comprimere dei programmi eseguibili, in modo da poter poi essere avviati senza bisogno di procedere alla loro espansione. In pratica, si prende un programma, lo si comprime e, all'apparenza, questo continua a funzionare come prima.

Il programma UPX è realizzato per vari tipi eseguibili e il suo comportamento si adatta alle circostanze. In particolare, per quanto riguarda i sistemi GNU/Linux, l'avvio del programma compresso implica una fase di estrazione nella directory temporanea '`/tmp/`' e l'avvio successivo di quanto estratto. Come conseguenza più importante si ha che il processo che viene messo in funzione ha un'apparenza diversa rispetto al solito. A titolo di esempio, si suppone di avere compresso il programma '**yes**' e di averlo avviato in questo modo:

```
$ yes ciao > /dev/null &[Invio]
```

Se si osserva con il programma '**ps**', in certi casi potrebbe sembrare ancora tutto normale:

```
$ ps x[Invio]

  PID TTY      STAT   TIME COMMAND
…
 7513 tty5     R      0:41 yes ciao
 7529 tty5     R+     0:00 ps x
```

Ma in altri casi, il processo si mostra attraverso un numero:

```
$ ps[Invio]
```



```
  PID TTY          TIME CMD
 6173 tty5     00:00:00 sh
 7513 tty5     00:01:36 3
 7545 tty5     00:00:00 ps
```

```
$ pstree[Invio]

init-+-ahc_dv_0
     …
      |-sh-+-3
      |    `-pstree
     …
```

Segue la descrizione del modello sintattico per l'uso di '**upx**':

```
upx [comando] [altre_opzioni] [file_eseguibile…]
```

Il modello sintattico indica la presenza di un'opzione iniziale, con lo scopo di dichiarare il tipo di azione da compiere (se viene omessa, si intende la volontà di comprimere i file), seguita eventualmente da altre opzioni, quindi dai file da comprimere. La tabella successiva riepiloga i comandi disponibili; si rammenti che si può usare una sola opzione di questo tipo, ovvero si può indicare un solo comando.

Tabella 20.94. Comandi.

Opzione	Descrizione
	L'assenza dell'opzione di comando implica l'intenzione di comprimere i file secondo la modalità predefinita.
`-1|2|3|4|5|6|7|8|9` `--best`	Specifica il livello di compressione: con '**-1**' si ottiene un livello minimo, mentre con '**-9**' richiede una compressione molto migliore. Utilizzando l'opzione '**--best**' si intende richiedere la compressione migliore che sia disponibile.
`-d`	Decomprime i file compressi in precedenza, riportandoli così al loro stato originale.
`-t`	Verifica l'integrità dei file che si presume siano stati compressi in precedenza.
`-l`	Mostra alcune informazioni sullo stato di file che si presume siano stati compressi in precedenza.

Segue la descrizione di alcuni esempi.

- `$ upx mio_programma[Invio]`

  Comprime il file '`./mio_programma`':

```
                       Ultimate Packer for eXecutables
          Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
UPX 1.25        Markus F.X.J. Oberhumer & Laszlo Molnar        Jun 29th 2004

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     12536 ->      7822   62.40%    linux/386    mio_programma

Packed 1 file.
```

- `$ upx -d mio_programma[Invio]`

  Estrae il file compresso in precedenza:

```
                       Ultimate Packer for eXecutables
          Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
UPX 1.25        Markus F.X.J. Oberhumer & Laszlo Molnar        Jun 29th 2004

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     12536 <-      7822   62.40%    linux/386    mio_programma

Unpacked 1 file.
```

- `$ upx --best mio_programma[Invio]`

  Comprime il file '`./mio_programma`' utilizzando il grado di compressione maggiore, anche se in tal caso non si ottengono miglioramenti:

```
                    Ultimate Packer for eXecutables
       Copyright (C) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004
UPX 1.25         Markus F.X.J. Oberhumer & Laszlo Molnar        Jun 29th 2004

        File size         Ratio      Format      Name
   --------------------   ------   -----------   -----------
     12536 ->      7822   62.40%    linux/386    mio_programma

Packed 1 file.
```

20.14.4 Utilizzo di «unp» e di «ucat»

Unp[38] è un pacchetto composto di piccoli programmi frontali per facilitare l'estrazione di file compressi, senza dover ricordare comandi e sintassi differenti; comandi che invece vengono avviati in modo trasparente e appropriato.

```
unp file... [-- opzioni_specifiche]
```

```
ucat file...
```

Il primo dei due modelli sintattici mostrati fa riferimento al programma '**unp**', con il quale si estraggono i file indicati come argomento; eventualmente, preceduti da due trattini '**--**', si possono aggiungere delle opzioni specifiche per il programma che si ritiene venga usato da '**unp**' per l'estrazione, ma ovviamente in questo modo perde di significato l'uso del programma frontale. Il secondo dei due modelli fa riferimento al programma '**ucat**', da intendere come un programma che legge i file indicati, li estrae e li emette attraverso lo standard output. Segue la descrizione di alcuni esempi.

- `$ unp mio.tar.gz` [*Invio*]

  In questo caso, si presume che il file '**mio.tar.gz**' sia stato generato con la coppia «tar+gzip». Il programma lo fa estrarre, come se venisse usato il comando seguente:

  `$ tar xzvf mio.tar.gz` [*Invio*]

- `$ unp mio.gz` [*Invio*]

  In questo caso, si presume che il file '**mio.gz**' sia stato generato da Gzip. Il programma lo fa estrarre, come se venisse usato il comando seguente:

  `$ gunzip < mio.gz > mio` [*Invio*]

  In pratica, rispetto al comportamento comune di Gzip, il file originale rimane intatto e l'estrazione avviene nella directory corrente.

- `$ ucat mio.bz2` [*Invio*]

  Mostra il contenuto del file '`mio.bz2`'.

Altri programmi affini.

star(1)[39]	Si tratta di un programma Tar alternativo e compatibile con i vecchi standard.
arc(1)[40]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, originario per sistemi CP/M e Dos.
arj(1)[41]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, originario per sistemi Dos. Si osservi che nelle sue versioni originali, questo programma non era software libero.
zoo(1)[42]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, sviluppato originariamente per sistemi Dos.
zip(1), *unzip(1)*[43]	Si tratta di un programma di archiviazione e compressione, per gruppi di file, sviluppato originariamente per i sistemi Dos.
lzop(1)[44]	Si tratta di un programma di compressione, per file singoli, che funziona sostanzialmente come Gzip, ma con l'obiettivo di essere più veloce.
PPMd(1)[45]	Si tratta di un programma di compressione, per file singoli, che funziona sostanzialmente come Gzip.

dact(1)[46]	Si tratta di un programma di compressione, per file singoli, che funziona sostanzialmente come Gzip, utilizzando algoritmi di compressione differenti e selezionando di volta in volta quello che si dimostra più adatto.
makeself(1)[47]	Si tratta di uno script, in grado di creare un altro script che incorpora un archivio compresso di dati. Tale script ottenuto, se avviato, provvede da solo a usare gli strumenti necessari a estrarre il contenuto incorporato. Si osservi che l'archivio ottenuto è sì autoestraente, ma per questo si avvale di programmi esterni, anche se comuni, pertanto non è autonomo.

20.15 File FIFO

Ciò che è noto comunemente come «*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», indicato a volte 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.

20.15.1 Utilizzo di «mkfifo»

Il programma di servizio '**mkfifo**'[48] crea uno o più file FIFO (*pipe* con nome):

```
mkfifo [opzioni] file...
```

Opzione	Descrizione
`-m` ***modalità_dei_permessi*** `--mode=`***modalità_dei_permessi***	Questa opzione permette di specificare esplicitamente i permessi del file che viene creato. La modalità può essere espressa sia in forma numerica, sia simbolica, come è possibile fare con il programma '**chmod**' (20.7.3). Il valore predefinito di questi permessi è 0666_8, meno il valore della maschera dei permessi.

Nell'esempio seguente vengono mostrati una sequenza di comandi con i quali, creando due file FIFO, si ottiene lo stesso risultato di un condotto come '**cat mio_file | sort | lpr**'.

`$ mkfifo fifo1 fifo2` [*Invio*]

Crea due file FIFO: '`fifo1`' e '`fifo2`'.

`$ cat mio_file >> fifo1 &` [*Invio*]

Invia '`mio_file`' a '`fifo1`' senza attendere ('**&**').

`$ sort < fifo1 >> fifo2 &` [*Invio*]

Esegue il riordino di quanto ottenuto da '`fifo1`' e invia il risultato a '`fifo2`' senza attendere ('**&**').

`$ lpr < fifo2` [*Invio*]

Accoda la stampa di quanto ottenuto da '**fifo2**'.

20.16 File di dispositivo

I file di dispositivo sono riferimenti a funzionalità contenute nel kernel. Nei sistemi Unix, questi file di dispositivo devono indicare due numeri, detti ***primario*** e ***secondario*** (oppure *major* e *minor*, secondo la terminologia originale), dove il primo rappresenta il tipo di dispositivo e il secondo serve a identificare esattamente un dispositivo particolare. 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 file speciali, contenuti tradizionalmente nella directory '`/dev/`'. Anche i nomi che si danno a questi file possono variare da un sistema Unix all'al-

tro; in certi casi ci sono piccole differenze tra le stesse distribuzioni GNU/Linux.[49]

Dal momento che questi file servono solo in quanto contengono i numeri primario e secondario di un certo dispositivo, potrebbero funzionare anche collocati al di fuori della loro directory tradizionale, utilizzando eventualmente nomi differenti. Questa possibilità viene sfruttata da alcune distribuzioni GNU/Linux, nella fase di installazione, quando nei dischetti di avvio vengono creati al volo i file di dispositivo necessari a completare l'operazione, utilizzando eventualmente la stessa 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 caratteri, presi ognuno singolarmente, oppure richieda che i dati siano raggruppati in blocchi di una dimensione determinata. Nel primo caso si parla di dispositivo a caratteri, mentre nel secondo di dispositivo a blocchi.

Dato che i dispositivi fisici sono gestiti attraverso file di dispositivo, l'accesso all'hardware viene controllato con i permessi che vengono dati a questi file. La gestione dei permessi è molto importante nell'impostazione che viene data al sistema ed è uno dei punti su cui si trovano le differenze significative tra le varie distribuzioni GNU/Linux. Inoltre, l'esistenza di utenti e gruppi fittizi, con nomi come '**floppy**', '**sys**', '**daemon**' e altri, dipende spesso da questa esigenza di controllo dell'accesso ai dispositivi.

20.16.1 Utilizzo di «mknod»

Il programma di servizio '**mknod**' [50] permette di creare un file FIFO oppure un file di dispositivo:

```
mknod [opzioni] file tipo [numero_primario numero_secondario]
```

Il tipo di file viene indicato attraverso una lettera, mentre i numeri primario e secondario sono richiesti solo quando non si tratta della creazione di un file FIFO. La creazione di file di dispositivo è riservata all'utente '**root**'.

Tipo	Descrizione
`p`	La lettera '**p**' indica un file FIFO.
`b`	La lettera '**b**' indica un dispositivo a blocchi con memoria tampone (*buffer*).
`c`	La lettera '**c**' indica un dispositivo a caratteri con memoria tampone.
`u`	La lettera '**u**' indica un dispositivo a caratteri senza memoria tampone.

Opzione	Descrizione
`-m` ***modalità_dei_permessi*** `--mode=`***modalità_dei_permessi***	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**' (20.7.3). Il valore predefinito di questi permessi è 0666_8, meno il valore della maschera dei permessi.

Segue la descrizione di alcuni esempi.

- `$` **`mknod fifo1 p`** [*Invio*]

  Crea il file FIFO '`fifo1`' esattamente come si potrebbe fare utilizzando il programma '**mkfifo**'.

- `#` **`mknod -m 0600 tty9 c 4 9`** [*Invio*]

  Crea il file di dispositivo a caratteri '`tty9`', nella directory corrente, utilizzando dei permessi opportuni.

- `#` **`mknod -m 0660 hda1 b 3 1`** [*Invio*]

  Crea il file di dispositivo a blocchi '`hda1`', nella directory corrente, utilizzando dei permessi opportuni.

20.16.2 File «/dev/MAKEDEV»

Il file '`/dev/MAKEDEV`' è uno script molto importante che si occupa di ricreare i file di dispositivo, rispettando le convenzioni del proprio sistema particolare:

```
/dev/MAKEDEV dispositivo...
```

Infatti, non c'è solo il problema di definire il nome e i numeri primario e secondario: occorre anche stabilire i permessi corretti, l'utente e il gruppo proprietari. Trascurando questi particolari, si rischierebbe di aprire dei buchi, anche gravi, nella sicurezza del sistema.

In tal senso, questo script è diverso da un sistema operativo all'altro. Solo il nome e la collocazione sono definiti dallo standard generale dei sistemi Unix.

Generalmente si possono indicare come argomento uno o più nomi di file di dispositivo, senza il percorso. Questi dovrebbero essere creati nella directory corrente. Si vedano gli esempi successivi.

- `#` **`/dev/MAKEDEV tty1`** [*Invio*]

  Crea il file di dispositivo corrispondente alla prima console virtuale, assegnandogli tutti gli altri attributi corretti.

- `#` **`/dev/MAKEDEV sda`** [*Invio*]

  Crea il file di dispositivo corrispondente alla prima unità di memorizzazione di massa, assegnandogli tutti gli altri attributi corretti.

Normalmente, lo script riconosce al posto del nome di un file di dispositivo, il nome di un gruppo di questi. La tabella successiva riguarda alcuni nomi di gruppi di file che si utilizzano nella distribuzione GNU/Linux Debian.

Tabella 20.102. Alcuni nomi di gruppi di file di dispositivo nella distribuzione GNU/Linux Debian.

Nome	Descrizione
`generic`	Crea i file di dispositivo più comuni.
`usb`	Crea i file di dispositivo utili per i componenti collegati attraverso un bus USB.
`md`	Crea i file di dispositivo utili per la gestione di dischi RAID gestiti via software.

Tabella 20.103. La lettera che appare all'inizio dei permessi dei file, quando si usa '**ls**'.

lettera	Significato
`-`	Un file di dati puro e semplice.
`d`	Directory.
`l`	Collegamento simbolico.
`p`	File FIFO (*pipe* con nome).
`c`	Dispositivo a caratteri.
`b`	Dispositivo a blocchi.
`s`	Socket di dominio Unix.

20.17 Tabelle riepilogative

Tabella 20.104. Directory, percorsi e contenuti

Comando	Descrizione
`cd [~]`	Cambia la directory corrente, raggiungendo la directory personale dell'utente.
`cd` ***percorso***	Cambia la directory corrente per raggiungere la directory indicata; se il percorso è relativo, questo si aggiunge alla directory corrente di partenza.

Comando	Descrizione
`cd ~`***utente***	Cambia la directory corrente per raggiungere la directory personale dell'utente indicato (sempre che i permessi di accesso alla directory lo consentano).
`pwd`	Mostra la directory corrente.
`ls [.]`	Elenca i file e il contenuto della directory corrente.
`ls` ***nome***…	Elenca i file e il contenuto delle directory indicate; oppure, in mancanza di argomenti, elenca il contenuto della directory corrente.
`ls -l` ***nome***…	Elenca quanto richiesto, dando maggiori informazioni.
`ls -i` ***nome***…	Elenca quanto richiesto, aggiungendo il numero di inode.
`ls -a` ***nome***…	Elenca quanto richiesto, senza escludere i nomi che iniziano con un punto.
`ls -R` ***nome***…	Elenca quanto richiesto, continuando ricorsivamente nelle sottodirectory.
`ls -F` ***nome***…	Elenca i file, aggiungendovi un simbolo alla fine, per evidenziarne il tipo.
`mkdir -p` ***directory***…	Crea le directory indicate e anche le eventuali directory precedenti, se necessario.
`mkdir -m` ***permessi directory***…	Crea le directory indicate specificando la modalità di accesso (i permessi).
`rmdir -p` ***directory***…	Cancella le directory indicate e anche le eventuali directory precedenti, se queste risultano vuote.
`basename` ***percorso*** `[`***estensione***`]`	Estrae il nome finale di un percorso, togliendo anche il suffisso, se indicato.
`dirname` ***percorso***	Estrae la directory da un percorso. In pratica, elimina il nome finale dal percorso.
`file` ***nome***…	Elenca i file classificandone il tipo, in base al magic number.
`du -s` ***nome***…	Calcola lo spazio totale utilizzato, espresso in kibibyte (simbolo: «Kibyte»), per ogni directory indicata.
`which` ***nome***…	Restituisce i percorsi dei file binari che verrebbero utilizzati per i comandi indicati come argomento.
`whereis` ***nome***…	Localizza i file binari e le pagine di manuale dei comandi indicati come argomento.

Tabella 20.105. Proprietà, permessi e attributi

Comando	Descrizione
`chown` ***utente file***…	Cambia la proprietà dei file indicati, in modo che appartengano all'utente specificato.
`chown -R` ***utente***`:` ***directory***	Cambia la proprietà della directory indicata e dei suoi file, comprese le sottodirectory, in modo che appartengano all'utente indicato e al gruppo dello stesso utente.
`chgrp -R` ***gruppo directory***	Cambia la proprietà della directory indicata e dei suoi file, comprese le sottodirectory, in modo che appartengano al gruppo indicato, senza cambiare l'utente.

Comando	Descrizione
`chmod -R a+X` ***directory***	Attribuisce i permessi di «esecuzione» a tutte le directory e ai file che ne hanno già almeno uno (per il proprietario, o il gruppo o gli altri), per tutti i tipi di utenti, a partire dalla directory indicata.
`find` ***directory*** `-type d ↩` `↪-exec chmod g+s \{\} \;`	Attiva il bit SGID per le sole directory a partire da quella di partenza indicata (si suppone di utilizzare la shell Bash).
`find` ***directory*** `-type f ↩` `↪-exec chmod 0444 \{\} \;`	Definisce la modalità, in sola lettura, per tutti i file normali, a partire dalla directory indicata (si suppone di utilizzare la shell Bash).

Tabella 20.106. Copia, collegamento, spostamento, cancellazione e archiviazione

Comando	Descrizione
`cp` ***file_origine file_destinazione***	Copia un file normale generandone un altro con il nome che appare alla fine della riga di comando.
`cp` ***file_origine… directory_di_destinazione***	Copia i file di origine nella directory di destinazione indicata, mantenendo così gli stessi nomi di partenza.
`cp -dp` ***file_origine… directory_di_destinazione***	Copia i file di origine, riproducendo il più possibile le caratteristiche originali. In particolare, i collegamenti simbolici vengono mantenuti come tali.
`cp -dpR` ***origine… directory_di_destinazione***	Copia i file o le directory di origine, in modo ricorsivo, riproducendo il più possibile le caratteristiche originali. In particolare, i collegamenti simbolici vengono mantenuti come tali.
`cp -dpRl` ***origine… directory_di_destinazione***	Copia le directory di origine ed eventuali discendenti, mentre per i file vengono generati solo collegamenti fisici.
`ln` ***file_origine file_destinazione***	Crea un collegamento fisico con il nome che appare alla fine della riga di comando sullo stesso inode del primo.
`ln -s` ***file_origine file_destinazione***	Crea un collegamento simbolico con il nome che appare alla fine della riga di comando, che punta al file di origine.
`ln [-s]` ***origine… directory_di_destinazione***	Crea dei collegamenti nella directory di destinazione, con lo stesso nome dei file di origine.
`mv` ***nome_origine nome_destinazione***	Cambia il nome di un file o di una directory.
`mv` ***nome_origine… directory_destinazione***	Sposta i file o le directory di origine nella directory di destinazione indicata.

`for a in *.`***ext_1***`; ↩`
`↪do ↩`
`↪mv $a 'basename $a .`***ext_1***`'.`***ext_2***`; ↩`
`↪done`
	Sostituisce l'estensione '***.ext_1***' a tutti i file contenuti nella directory, mettendo al suo posto l'estensione '***.ext_2***'.

Comando	Descrizione
`rm` ***nome*…**	Cancella i file indicati.
`rm -r` ***nome*…**	Cancella file e directory in modo ricorsivo.
`find / -type f -name core` ↩ ↪`-exec rm -i \{\} \;`	Elimina tutti i file '`core`' (solo i file normali).
`tar cf` ***archivio_di_destinazione*** ↩ ↪***directory_di_origine***	Archivia una directory in un file, utilizzando '**`tar`**' senza compressione.
`tar czf` ***archivio_di_destinazione*** ↩ ↪***directory_di_origine***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo con '**`gzip`**'.
`tar cjf` ***archivio_di_destinazione*** ↩ ↪***directory_di_origine***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo con '**`bzip2`**'.
`tar cf -` ***directory_di_origine*** ↩ ↪`| gzip -9 >` ***archivio_di_destinazione***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo al massimo con '**`gzip`**'.
`tar cf -` ***directory_di_origine*** ↩ ↪`| bzip2 -9 >` ***archivio_di_destinazione***	Archivia una directory in un file, utilizzando '**`tar`**' e comprimendolo al massimo con '**`bzip2`**'.
`tar tf` ***archivio_di_origine*** ↩ ↪[***'modello_da_estrarre'***]…	Elenca il contenuto di un archivio '**`tar`**'.
`tar tzf` ***archivio_di_origine*** ↩ ↪[***'modello_da_estrarre'***]…	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**'.
`tar tjf` ***archivio_di_origine*** ↩ ↪[***'modello_da_estrarre'***]…	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**'.
`gunzip <` ***archivio_di_origine*** ↩ ↪`| tar tf -` [***'modello_da_estrarre'***]…	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**', utilizzando un condotto.
`bunzip2 <` ***archivio_di_origine*** ↩ ↪`| tar tf -` [***'modello_da_estrarre'***]…	Elenca il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**'.
`tar xpf` ***archivio_di_origine*** ↩ ↪[***'modello_da_estrarre'***]…	Estrae il contenuto di un archivio '**`tar`**' a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`tar xpzf` ***archivio_di_origine*** ↩ ↪[***'modello_da_estrarre'***]…	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**', a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`tar xpjf` ***archivio_di_origine*** ↩ ↪[***'modello_da_estrarre'***]…	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**', a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`gunzip <` ***archivio_di_origine*** ↩ ↪`| tar xpf -` [***'modello_da_estrarre'***]…	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`gzip`**' (utilizzando un condotto), a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.

Comando	Descrizione
`bunzip2 <` ***archivio_di_origine*** ↩ ↪`| tar xpf -` [***'modello_da_estrarre'***]…	Estrae il contenuto di un archivio '**`tar`**', compresso con '**`bzip2`**', a partire dalla directory corrente, mantenendo il più possibile inalterati gli attributi originali dei file.
`tar cf /dev/fd0 -L 1440` ↩ ↪`-M` ***origine_da_archiviare***	Archivia una directory, o un file, su una serie di dischetti inizializzati a 1440 Kibyte, ma senza file system.
`tar tf /dev/fd0 -L 1440` ↩ ↪`-M` [***'modello_da_estrarre'***]…	Elenca il contenuto di un archivio contenuto in una serie di dischetti da 1440 Kibyte senza file system.
`tar xpf /dev/fd0 -L 1440` ↩ ↪`-M` [***'modello_da_estrarre'***]…	Estrae, nella directory corrente, il contenuto di un archivio contenuto in una serie di dischetti da 1440 Kibyte senza file system.
`dd if=`***file_da_suddividere*** ↩ ↪`of=`***file_suddiviso_1*** ↩ ↪`bs=100k count=1 skip=0` `dd if=`***file_da_suddividere*** ↩ ↪`of=`***file_suddiviso_2*** ↩ ↪`bs=100k count=1 skip=1` … `dd if=`***file_da_suddividere*** ↩ ↪`of=`***file_suddiviso_n*** ↩ ↪`bs=100k count=1 skip=(`***n***`-1)`	Suddivide un file in pezzetti da 100 Kibyte l'uno.
`cat` ***file_suddiviso*…** `>` ***file_riaggregato***	Ricomposizione di un file suddiviso in pezzetti.

Tabella 20.107. Contenuto dei file e ricerche

Comando	Descrizione
`cat` ***file*…**	Emette i file indicati, in sequenza, attraverso lo standard output.
`cat` ***file*…** `>` ***file_unito***	Concatena i file assieme in un file unico, indicato alla fine.
`zcat` ***file*…**	Emette il contenuto dei file compressi indicati, in sequenza, attraverso lo standard output.
`more` ***file*** `less` ***file***	Scorre il file indicato sullo schermo.
`cat` ***file*…** `| more` ***file*** `cat` ***file*…** `| less` ***file***	Scorre il file indicato sullo schermo.
`grep` ***'espressione_regolare' file*…**	Cerca nei file le righe corrispondenti al modello indicato dall'espressione regolare.
`zgrep` ***'espressione_regolare' file_compresso*…**	Cerca all'interno di file compressi le righe corrispondenti al modello indicato dall'espressione regolare.
`find` ***directory_di_partenza*** ↩ ↪`-name` ***'modello'*** `-print`	Cerca i file che corrispondono al modello, a partire da una directory particolare.
`find / -type f -name core -print`	Cerca tutti i file '`core`' (solo i file normali).

Comando	Descrizione
`find` ***directory_di_partenza*** ↩ ↪`-name` ***'modello'*** ↩ ↪`-exec grep` ***'espressione_regolare'*** ↩ ↪`\{\} \; -print`	Cerca tra i file che corrispondono al modello, le righe che corrispondono all'espressione regolare di Grep. Il nome del file che ha trovato una corrispondenza con Grep viene visualizzato subito dopo la riga relativa.

20.18 Riferimenti

«

- Andreas Grünbacher, *Extended Attributes*, *http://acl.bestbits.at/about.html*
- *SuSE Linux administration guide: Access control lists in Linux*, *http://www.suse.de/~agruen/acl/chapter/fs_acl-en.pdf*
- Andreas Grünbacher, *POSIX access control lists on Linux*, *http://www.suse.de/~agruen/acl/linux-acls/*

[1] **GNU core utilities** GNU GPL

[2] **GNU core utilities** GNU GPL

[3] **GNU core utilities** GNU GPL

[4] **GNU core utilities** GNU GPL

[5] **GNU core utilities** GNU GPL

[6] **util-linux: namei** GNU GPL

[7] **GNU core utilities** GNU GPL

[8] **GNU core utilities** GNU GPL

[9] **File** software libero con licenza speciale

[10] Questa sezione su '**du**' è identica alla 19.4.1.10.

[11] **GNU core utilities** GNU GPL

[12] **util-linux: whereis** UCB BSD

[13] **GNU core utilities** GNU GPL

[14] La facoltà di indicare i due punti, senza mettere esplicitamente il nome del gruppo, è un'estensione non prevista dallo standard; lo stesso vale per la possibilità di specificare il gruppo, ma non l'utente.

[15] **GNU core utilities** GNU GPL

[16] **GNU core utilities** GNU GPL

[17] **Second extended file system management programs** GNU GPL

[18] **Second extended file system management programs** GNU GPL

[19] Come succede spesso, i nomi che rappresentano acronimi possono indicare cose differenti in contesti diversi. Nel caso particolare della sigla ACL, questa si usa anche in altre situazioni, specie nella gestione dell'accesso a servizi HTTP (serventi o proxy), dove si vuole regolare l'accesso al servizio o a delle risorse particolari. Ciò che si deve intendere è che la sigla ACL, anche se, come acronimo, fa riferimento alle stesse parole, rappresenta situazioni differenti in base al contesto.

[20] **ACL** GNU LGPL e GNU GPL

[21] **GNU core utilities** GNU GPL

[22] **GNU core utilities** GNU GPL

[23] **GNU core utilities** GNU GPL

[24] **GNU core utilities** GNU GPL

[25] **GNU core utilities** GNU GPL

[26] **GNU ddrescue** GNU GPL

[27] **sdd** GNU GPL

[28] **cstream** software libero con licenza speciale

[29] **GNU core utilities** GNU GPL

[30] **GNU core utilities** GNU GPL

[31] **GNU core utilities** GNU GPL

[32] **GNU core utilities** GNU GPL

[33] **GNU cpio** GNU GPL

[34] **GNU tar** GNU GPL

[35] **Gzip** GNU GPL

[36] **Bzip2** software libero con licenza speciale

[37] **UPX** GNU GPL con eccezioni e precisazioni

[38] **Unp** GNU GPL

[39] **Star** GNU GPL

[40] **Arc** GNU GPL

[41] **Arj** GNU GPL

[42] **Zoo** dominio pubblico

[43] **InfoZip** software libero con licenza speciale

[44] **Lzop** GNU GPL

[45] **PPMd** software libero con licenza speciale

[46] **Dact** GNU GPL

[47] **Makeself** GNU GPL

[48] **GNU core utilities** GNU GPL

[49] Nei sistemi GNU/Linux, i nomi di riferimento dei file di dispositivo dovrebbero essere quelli indicati nella documentazione interna ai sorgenti del kernel, precisamente il file '***sorgenti_linux***`/Documentation/devices.txt`'.

[50] **GNU core utilities** GNU GPL