Come funziona Linux

Seconda parte

Utenti e filesystem

Nella puntata precedente abbiamo visto come funziona il boot di Linux. Questa volta analizzeremo altri due aspetti importanti: gli utenti e la gestione dei dischi e del filesystem.

di Giuseppe Zanetti

Nella spiegazione cercherò di procedere velocemente con gli argomenti di cui un lettore medio di MC dovrebbe già avere padronanza, soffermandomi maggiormente sui concetti specifici di Linux. Negli esempi verrà presentato solamente l'utilizzo base dei comandi. Per ulteriori informazioni si lascia al lettore il compito di leggere i manuali in linea (es: man ls), eventualmente nella versione italiana, disponibile ormai sulla maggior parte delle distribuzioni di Linux oppure su http://www.pluto.linux.it/ildp/man.

Utenti e gruppi

Linux è un sistema multiutente. Più persone possono utilizzarlo contemporaneamente come se avessero una macchina dedicata e possono lanciare più programmi. Vedremo nel seguito che ad ogni risorsa del sistema (ad esempio un file) sono associati determinati permessi. Ogni programma in esecuzione sul sistema (processo) può accedere solamente alle risorse a cui può accedere l'utente che l'ha lanciato (proprietario, owner).

Alcuni degli esperimenti presentati presuppongono che vi colleghiate al sistema utilizzando l'utente di amministrazione root. Poiché root dispone pressoché di tutti i permessi, è buona norma evitarne l'utilizzo nella pratica comune, sia per ridurre il rischio di compiere errori che possano danneggiare il sistema, sia per evitare di eseguire senza protezioni eventuali programmi difettosi, cavalli di Troia o javascript maliziosi. La presenza dei permessi rende Linux un sistema operativo estremamente sicuro nei confronti di eventuali virus, tentiamo però di dargli anche noi una mano evitando comportamenti scorretti che possano aggirare tali protezioni. Collegatevi dunque come root solamente per compiere la manutenzione del sistema, mai per lavorare.

Generalmente per mandare in esecuzione dei processi ci si collega alla macchina con un determinato nome di utente (username, nome di login) utilizzando una shell (ad esempio aperta sulla console oppure da rete mediante telnet) e si esegue il programma desiderato.

Prima di poter accedere alla shell vengono richieste le credenziali dell'utente, ovvero username e password. Per terminare la sessione di lavoro si utilizza il comando exit.

```
Linux 2.2.12 (freddy.profuso.com)
(ttyp4)

freddy login: beppe
Password: ******
You have mail.
$ comandi dell'utente
...
$ exit
```

È però possibile lanciare processi con i permessi di un utente anche senza trovarsi fisicamente davanti ad una tastiera, ad esempio schedulandone l'avvio a determinati orari mediante il programma cron oppure richiedendo un servizio di rete (ad esempio una pagina Web che richiama uno script CGI). Anche il semplice invio di un messaggio di e-mail ad un server manda in esecuzione un processo che ne gestisce la consegna.

Per tutti questi motivi è necessario che ogni utente possa gestire solamente certi file. Nel caso della posta elettronica e di altri servizi, come il server Web o ftp, non si tratta di un utente "normale", ma bensì di un utente fittizio ("di sistema") dedicato allo scopo (mail, uucp, ftp, webmaster, ...). Per massimizzare la sicurezza, generalmente a questi utenti non è neppure associata una password.

La lista degli utenti definiti su una macchina è contenuta nel file /etc/passwd:

```
root:FqG58Gjj2F:0:0:Amministrator
  e di sistema:/:/bin/bash
mail:*:8:12:Gestore della po-
  sta:/var/spool/mail:
beppe:HJKhhst52:306:100:Giuseppe
  Zanetti:/home/beppe:/bin/bash
pippo:IGG8hg28R:307:100:Pippo de
  Pippis:/home/pippo:/usr/local/b
  in/bbs
```

Analizzandolo si scopre che in esso, oltre allo username dell'utente (da 2 a 8 caratteri), sono contenute altre informazioni importanti, separate dal simbolo di due punti: la password criptata, lo UID, il GID, nome e cognome (GECOS), il percorso della home directory (directory personale) dell'utente ed il programma

che deve essere mandato in esecuzione quando l'utente si collega (esegue il "login") alla macchina. Se non specificato viene usata la shell `/bin/sh`.

La password in UNIX viene codificata mediante un meccanismo di tipo "one way", in modo che dalla password criptata non sia possibile, mediante un qualche procedimento inverso, risalire alla password in chiaro. Al momento del login la password inserita viene prima criptata e poi confrontata con quella contenuta in /etc/password. Solo se le due password sono uguali all'utente viene concesso il permesso di accedere al sistema. Nonostante non venga memorizzata la password in chiaro, il sistema non è completamente sicuro. Molti utenti infatti utilizzano password banali (ad esempio il nome della moglie, la targa della propria auto o una parola contenuta nel dizionario). In questo caso, dato che il file delle password è leggibile da tutti gli utenti, un qualche malintenzionato potrebbe prelevarlo e tentare una ricerca delle password per forza bruta a partire da un dizionario di parole banali (pippo, pluto, marco, anna, ...). Per questo motivo è necessario porre una certa attenzione nella scelta della password, tentando ad esempio di generarla a caso o di mischiare le lettere minuscole con maiuscole, numeri e segni di punteggiatura (es: PsG435J!a). Per evitare questo problema si possono utilizzare le shadow password, in cui le password criptate vengono tenute in un file `/etc/shadow` leggibile solo da root. I sistemi più recenti supportano metodi di autenticazione più raffinati e non necessariamente basati sul meccanismo delle password (es: NIS, PAM, ...).

Tornando alla descrizione di `/etc/passwd`, i campi UID (User ID) e GID (Group ID) meritano una spiegazione più accurata: il primo è un numero che rappresenta in modo univoco l'utente nel sistema (0 è lo UID standard che indica sempre l'utente root, mentre, in questo particolare esempio, 306 indica l'utente "beppe"). Il GID invece non è necessariamente univoco e indica il gruppo a cui appartiene di default l'utente. Ad un gruppo possono appartenere più utenti, ad esempio in questo caso al GID 100, corrispondente al gruppo "users", appartengono sia beppe che pippo. La lista dei gruppi è contenuta nel file `/etc/group`:

```
root::0:root
bin::1:root,bin,daemon
mail::12:mail
users::100:beppe,lorenz,edo
```

Il primo campo rappresenta il nome del gruppo, mentre il secondo è una eventuale password criptata. Seguono lo UID ed una lista di utenti che possono appartenere a questo gruppo oltre che al proprio gruppo di default. Infatti un determinato utente non appartiene necessariamente al solo gruppo indicato in `/etc/passwd` (gruppo di default) ma può essere assegnato a più gruppi, aggiungendo il suo username in `/etc/group`.

La seguente tabella riassume i comandi utili per la gestione delle password e dei gruppi. Ovviamente solo root può creare o cancellare utenti o gruppi. Per i dettagli si consultino i manuali in linea (es: `man useradd`).

`useradd -m -c"Giuseppe Zanetti" beppe`	aggiunge al sistema un utente
`userdel utente`	rimuove un utente senza rimuovere i file ad esso associati
`userdel -r utente`	rimuove un utente e i file ad esso associati
`chsh utente`	cambia la shell per un utente
`chfn utente`	modifica i dati personali di un utente
`passwd`	modifica la propria password
passwd utente	root può modificare la password di qualunque utente
`id`	restituisce lo uid dell'utente con cui siamo collegati
`su - root`	acquisisce i permessi di un altro utente (in questo esempio permette di diventare root)
`newgrp gruppo`	acquisisce i permessi di un gruppo diverso da quello di default (vedere seguito dell'articolo)
`groupadd -g 300 gruppo`	crea un nuovo gruppo (con GID 300)
`gpasswd -a utente gruppo`	aggiunge un utente ad un gruppo
`gpasswd -d utente gruppo`	cancella un utente da un gruppo
`gpasswd gruppo`	assegna/modifica la password ad un gruppo
`groupdel gruppo`	rimuove un gruppo

Il Filesystem

Le directory ed i file in Linux sono organizzati all'interno di un disco o di una partizione in una struttura ad albero, detta filesystem.

All'avvio del sistema viene montato il filesystem principale, detto root (la directory root è quella principale del sistema, accessibile come /), residente ad esempio nella prima partizione del primo disco (/dev/hda1). In una directory del filesystem principale possono essere poi montati altri filesystem, ad esempio altre partizioni di disco oppure un floppy disk. In Linux, contrariamente a DOS e Windows, non esiste il concetto di volume (A:, B:, C:, ...) e non si può perciò semplicemente inserire un dischetto, copiarci i dati che servono e rimuoverlo, ma è necessario inserirlo, montarlo in una directory, copiare i dati ed infine smontarlo. Vedremo che questo permette, a costo di un minimo di scomodità, di avere delle prestazioni migliori.

I comandi base

I comandi per navigare nel filesystem e per lavorare con i file sono simili a quelli degli altri sistemi operativi. Quando si indica un percorso, il simbolo da utilizzare per separare le diverse directory che lo compongono è / (e non \).

Inizialmente la directory corrente è la home directory dell'utente (es: /home/beppe) ma è possibile spostarsi fra le directory mediante il comando `cd`. Le directory . e .. sono particolari, in quanto rappresentano rispettivamente la directory corrente e la directory precedente nell'albero.

La tabella seguente illustra l'utilizzo dei principali comandi per spostarsi fra le directory. Per ulteriori dettagli si rimanda ai manuali in linea (es: `man ls`).

`cd directory`	cambia la directory corrente di lavoro. Inizialmente l'utente ha come directory di lavoro la propria home directory
`cd`	torna alla propria home directory
`cd ..`	si sposta alla directory precedente
`cd /`	si sposta alla directory root
`cd /usr/local/bin`	si sposta alla directory /usr/local/bin
`cd doc`	si sposta in una sotto directory
`cd ../doc`	torna indietro di una directory e poi ridiscende ad una directory allo stesso livello di quella iniziale
`pwd`	stampa il percorso della directory corrente

Comandi per gestire i file

L'albero delle directory in Linux è simile per tutte le distribuzioni ed i file sono collocati nel sistema secondo una logica ben precisa, definita dallo standard FSSTND (si veda il riquadro).

Anche i comandi per listare il contenuto delle directory sono molto semplici:

`ls`	stampa la lista dei file nella directory corrente
`ls /home/beppe/doc`	stampa la lista dei file nella directory indicata
`ls -l`	stampa la lista in formato esteso
`ls -a`	visualizza tutti i file, compresi quelli nascosti (in UNIX vengono considerati "nascosti" i file inizianti con un punto, ad esempio .profile o la directory ..)
`ls -la`	utilizza assieme i due switch di opzione visti sopra, ovvero stampa la lista di tutti i file e in formato esteso
`ls file1 file2 ...`	lista più file o directory

Altri comandi utili sono quelli che permettono di conoscere l'occupazione di disco, sia di una directory (con tutte le sottodirectory ed i file in essi contenuti), sia di un intero filesystem:

I comandi che gestiscono file generalmente accettano nella linea di comando un numero indefinito di parametri.

Ciò consente di specificare più file, ma permette anche l'espansione delle espressioni regolari, che in Linux vengono espanse prima di essere passate al comando.

Ad esempio se nella directory corrente sono presenti i file pippo, pluto e paperino, la linea di comando

```
ls p*
```

viene prima espansa come

```
ls pippo pluto paperino
```

e solo in seguito eseguita. Il risultato è che l'utilizzo di una espressione regolare (o wildcard, o caratteri jolly) causa il passaggio al comando di un numero a priori indefinito di parametri.

Per questo motivo la maggior parte dei comandi che agiscono sui file accetta un numero indefinito di parametri. Nei manuali in linea tale situazione viene indicata col simbolo ...

I file (e le directory) possono essere

du directory	permette di conoscere l'occupazione di disco, di una directory, con tutte le sottodirectory ed i file in esse contenuti. du senza parametri calcola l'occupazione della directory corrente
`df directory`	calcola lo spazio utilizzato e quello ancora libero nel filesystem dove risie de la directory specificata. Se non si specifica una directory vengono elencati tutti i filesystem montati

Linux Filesystem Standard (FSSTND)

È un documento che definisce e standardizza i percorsi che devono esistere in un sistema Linux, oltre a definire delle regole su come posizionare i file all'interno delle directory. Attualmente la maggior parte delle distribuzioni di Linux si adegua alle indicazioni contenute in questo documento, in modo da massimizzare la compatibilità e da facilitare la portabilità dei pacchetti software da una distribuzione all'altra.

L'idea principale in FSSTND è di tenere separati i diversi file a seconda che:

- siano locali ad una sola macchina o condivisibili fra più macchine in rete
- si tratti di dati a sola lettura (`/usr`) o scrivibili (`/var`)

Queste distinzioni tendono a rendere più sicuro il sistema (in quanto meno file si possono scrivere, meno possibilità c'è di causare danni) e a facilitare la creazione di cluster di macchine. Inoltre facilitano la creazione di distribuzioni funzionanti quasi interamente su CD-ROM oppure di sistemi embedded. In entrambi i casi la maggior parte del software risiede su un supporto a sola lettura (CD-ROM o E2PROM) e le poche directory che devono poter essere scrivibili sono invece create su una ramdisk.

Analizziamo velocemente le principali directory che devono essere presenti in un sistema Linux, rimandando al documento FSSTND (prelevabile da http://www.pathname.com/fhs/) per ulteriori informazioni.

`/dev`

Contiene i file speciali che permettono di accedere alle periferiche hardware della macchina.

```
/etc
  |- X11
  +- skel
```

Contiene i file di configurazioni locali al proprio sistema. Rispetto ai vecchi sistemi ora in /etc non possono essere contenuti file binari, che devono essere spostati in /bin o /sbin.

La sottodirectory `X11` contiene i file di configurazione di X Window, mentre skel contiene i file di default che vengono copiati nella home directory di un utente quando questi viene creato mediante il comando adduser. Possono esistere altre sottodirectory per i file di configurazione di particolari programmi (es: `/etc/ppp` oppure `/etc/httpd`).

```
/lib
  +- modules
```

Contiene le librerie condivise (shared libraries) necessarie ad eseguire i programmi in /bin e /usr/bin. La sottodirectory modules contiene i moduli del kernel.

`/proc`

È il mount point per il filesystem virtuale proc, che permette di estrarre o fornire informazioni al kernel (informazioni sui processi, statistiche, tuning, ...). Ad esempio `cat /proc/interrupts` permette di conoscere l'utilizzo degli interrupt hardware nel proprio sistema.

`/sbin`

Contiene l'insieme minimo di programmi eseguibili (linkati staticamente, ovvero che per funzionare non si appoggiano a librerie dinamiche) strettamente necessari per il boot del sistema: `clock, getty, init, update, mkswap, swapon, swapoff, halt, reboot, shutdown, fdisk, fsck.*, mkfs.*, lilo, arp, ifconfig, route`.

```
/usr
  |- X11R6
  |- bin
```

indicati utilizzando un indirizzamento relativo alla directory corrente (es: `doc/lettera.txt`) oppure assoluto, ovvero scrivendo il percorso completo espresso a partire dalla root (es: `/home/beppe/doc/lettera.txt`).

Un trucco da vero Amministratore di sistema (con la A maiuscola) per eseguire un programma contenuto nella directory corrente anche se questa non è contenuta nella variabile PATH (che indica i percorsi dove la shell deve cercare i comandi) consiste nello scrivere il comando indicandone il percorso in modo relativo alla directory corrente, ovvero facendone precedere il nome da ./

```
./comando
```

La tabella che segue contiene alcuni esempi di comandi per copiare, spostare e visualizzare file.

`cp file1 file2`	copia un file con un nuovo nome
`cp file directory`	copia il file in una directory
`cp file directory/file`	copia un file in una directory cambiandogli nome (es: `cp pippo.txt /home/beppe/pluto.txt`)
`cp file1 ... fileN directory`	copia più file in una directory (es: cp `/usr/doc/*` . copia più file provenienti dall'espansione di una espressione regolare nella directory corrente, indicata da un punto)
`mv file1 file2`	sposta (rinomina) un file
`mv file1 ... fileN directory`	sposta più file
`rm file1 ... fileN`	cancella uno o più file
`mkdir directory`	crea una directory (es: `mkdir /home/beppe/doc`)
`rmdir directory`	cancella una directory (es: `rmdir doc`)
`more file`	visualizza un file (`q` per uscire, `spazio, b, enter` per muoversi all'interno del file, */stringa* per cercare una stringa, `:n` per saltare al prossimo file)
`vi file`	editor di file; altri editor sono joe, pico, emacs, ...
`cat file`	stampa su video uno o più file (vedremo che usando la ridirezione dell'output è possibile creare un file dall'unione di più file: `cat file1 file2 >file3`)
`grep stringa file`	cerca una stringa all'interno di un file (per ottenere l'output paginato usare: `grep beppe *.txt|more`)
`lpr file`	stampa il file sulla stampante

```
    |- dict
    |- doc
    |- etc
    |- games
    |- include
    |- info
    |- lib
    |- local
    |- man
    |- sbin
    |- share
    +- src
```

Contiene file condivisibili fra più macchine. La directory `/usr` generalmente ha una propria partizione e dovrebbe essere possibile montarla a sola lettura.

La directory `X11R6` contiene l'X Window System; `bin` e `sbin` contengono i programmi eseguibili (rispettivamente per l'utilizzo da parte di tutti gli utenti e quelli riservati all'amministratore); `doc` e `info` contengono la documentazione; `include` contiene gli header per il linguaggio C; `lib` contiene librerie; `man` le pagine dei manuali in linea. Infine `src` contiene i sorgenti dei programmi (i sorgenti di Linux sono `in /usr/src/linux`).

```
/usr/local
        |- bin
        |- doc
        |- etc
        |- games
        |- include
        |- info
        |- lib
        |- man
        |- sbin
        +- src
```

La directory `/usr/local` è simile nella struttura a `/usr`, ma è pensata per installare programmi o documentazioni locali alla macchina (aggiunti in seguito dall'amministratore di sistema).

```
/var
   |- log
   |- catman
   |- lib
   |- local
   |- named
   |- nis
   |- preserve
   |- run
   |- lock
   |- tmp
   +- spool
         |- at
         |- cron
         |- lpd
         |- mail
         |- mqueue
         |- rwho
         |- smail
         |- uucp
         +- news
```

FSSTND richiede che `/usr` sia montabile a sola lettura. Per questo motivo tutti i file scrivibili o variabili (directory di spool per stampanti e posta, file di log, file temporanei, ...) dovrebbero trovare posto in `/var`.

I log di sistema si trovano in `/var/log`, mentre `/var/spool/mail` contiene le mailbox degli utenti.

`/tmp`

In questa directory, scrivibile da tutti gli utenti, trovano posto i file temporanei utilizzati dai programmi, ad esempio contenenti i risultati intermedi di elaborazioni. In teoria il programma che crea un file temporaneo dovrebbe poi cancellarlo, ma in alcuni casi ciò non avviene (ad esempio se il programma muore prematuramente). Per questo motivo è buona norma tenere pulita tale directory cancellando a mano eventuali file inutili. In alcune distribuzioni di Linux tale operazione viene compiuta automaticamente ad ogni reboot.

Montaggio e smontaggio di filesystem

Esistono diversi tipi di filesystem, che corrispondono a diversi modi di organizzare logicamente i dati nel supporto magnetico, ognuno dei quali ha le sue proprie caratteristiche e limitazioni. Ad esempio il filesystem utilizzato da msdos (msdos o FAT) può essere montato nell'albero dei filesystem di Linux, di solito in /dos, ma comporta delle limitazioni sulle lunghezze dei nomi (8+3 caratteri), non distingue fra caratteri minuscoli e maiuscoli e non gestisce i permessi sui file. Perciò copiando un file da un'altra directory a /dos potrebbe succedere che il nome venga troncato, reso tutto minuscolo ed i permessi persi.

Il tipo di filesystem standard di Linux è ext2, ma sono supportati e possono essere letti e scritti anche filesystem particolari (romfs, iso9660) oppure propri di altri sistemi operativi (fat, vfat, minix, ufs, ...). È possibile inoltre condividere filesystem in rete (NFS, NetBIOS/SMB, Novell, coda, ...) ed utilizzare dischi compressi o criptati.

La tabella che segue indica i tipi di filesystem maggiormente utilizzati:

ext2	il filesystem standard di Linux
minix	i nomi sono limitati a 14 caratteri, il primo filesystem di Linux
msdos	permette di accedere a dischi MSDOS, gestisce solo 8+3 caratteri per i nomi, non gestisce i permessi e la differenza fra maiuscole e minuscole
umsdos	si tratta di un vero filesystem UNIX costruito "sopra" in filesystem DOS. Non vi sono limitazioni sui nomi e gestisce permessi, link simbolici e quant'altro serve. È piuttosto lento, se confrontato con ext2, ma permette di provare un vero sistema Linux senza dover ripartizionare il disco
vfat	permette di accedere al file system di Windows, non gestisce i permessi di UNIX
iso9660	è il filesystem utilizato per i CD-ROM. È a sola lettura e presenta delle limitazioni sulla lunghezza dei nomi e sulla profondità delle directory. Non gestisce i permessi. Diventa un vero filesystem UNIX se si usano le Rock Ridge extensions. Per crearlo occorre un software apposito
smbfs	è un filesystem compatibile NetBIOS che utilizza TCP/IP per il trasporto dei dati. Permette l'integrazione con le reti Windows. È leggermente diverso da un filesystem UNIX
proc	è un filesystem virtuale che permette di accedere/modificare le informazioni generate dal kernel (informazioni sui processi, statistiche, ...). Trattandosi di un filesystem generato al volo dal kernel, non occupa spazio su disco
nfs	È il filesystem di rete proprio del mondo UNIX

Abbiamo già avuto modo di vedere nella prima puntata di questo corso che il metodo utilizzato dai sistemi UNIX per accedere ad una periferica hardware è quello di passare attraverso un file speciale in /dev. In questo modo per l'accesso ad un disco o ad una partizione si possono utilizzare i comuni comandi e chiamate di sistema che si utilizzano per accedere ad un file.

Il filesystem principale / viene montato automaticamente dal sistema operativo, mentre per montare filesystem aggiuntivi è necessario utilizzare un comando del tipo:

```
mount -t tipo -oopzioni /dev/partizione /directory
```

Ad esempio

```
mount -t ext2 /dev/hda2 /extra
```

permette di montare il filesystem di tipo ext2 presente nella seconda partizione del primo hard disk (`/dev/hda2`) nella directory `/extra`. Non essendo specificate delle opzioni, il filesystem viene montato con le opzioni predefinite (ovvero permettendo sia la lettura che la scrittura).

Il seguente comando:

```
mount -t iso9660 -oro /dev/cdrom /mnt/cdrom
```

monta invece il CD-ROM (filesystem in formato ISO9660) nella directory `/mnt/cdrom`. Da notare che esso viene montato in sola lettura (ro=read only).

Infine un esempio valido per un floppy disk (`/dev/fd0` corrisponde ad A: in DOS, mentre `/dev/fd1` corrisponde a B:):

```
mount -t msdos /dev/fd0 /mnt/floppy
```

Per evitare di dover ogni volta specificare tutte le opzioni necessarie al mounting dei dischi nel filesystem è possibile scrivere tali informazioni nel file /etc/fstab. I programmi che sovrintendono all'avvio del sistema (che abbiamo visto la volta scorsa) potranno così montare tutti i filesystem che servono con un semplice `mount -a`:

```
/dev/hda1     /            ext2     defaults
/dev/hda2     /extra       ext2     defaults
/dev/cdrom    /mnt/cdrom   iso9660  user,ro,defaults
/dev/fd0      /mnt/floppy  msdos    user,defaults,noauto
```

Inoltre l'utente potrà montare un filesystem semplicemente scrivendo:

```
mount /mnt/floppy
```

L'opzione user consente il mounting del floppy e del CD-ROM anche ad un utente comune (di solito bisogna essere collegati al sistema come root), mentre noauto fa in modo che il floppy non venga montato automaticamente al boot.

Nelle versioni recenti di Linux possono essere specificati in `/etc/fstab` anche ulteriori parametri che indicano l'ordine con cui eseguire il montaggio o il check del filesystem al boot.

Per smontare un disco montato si utilizza il comando umount (nelle prime versioni di UNIX i comandi potevano avere fino a 7 caratteri e perciò la "n" è stata sacrificata "per ragioni storiche"). È necessario ricordarsi di smontare un filesystem montato prima di rimuovere l'hardware (ad esempio un dischetto), altrimenti potrebbero rimanere parti non scritte sul supporto.

```
umount /mnt/floppy
```

L'operazione di smontaggio funziona solamente se nessun processo sta utilizzando file contenuti nella directory montata.

Spesso i colpevoli siamo noi stessi, che abbiamo fatto un cd in una delle directory contenute nel filesystem montato. In questo caso è sufficiente riprovare a smontare il filesystem dopo essere ritornati alla root:

```
cd /
umount /mnt/floppy
```

Essendo Linux un sistema multitasking (più processi possono essere eseguiti contemporaneamente) e multiutente (più utenti possono accedere contemporaneamente al sistema), per non bloccare il lavoro degli utenti, è necessario rendere le operazioni d'accesso al filesystem le più veloci possibile. Viene perciò utilizzato un meccanismo di caching in memoria dei dati da scrivere su disco. I dati non vengono scritti immediatamente sul supporto, bensì vengono parcheggiati in apposite aree di memoria.

In questo modo l'utente riacquista immediatamente il controllo del sistema, senza dover aspettare il completamento delle operazioni del disco.

Ovviamente vi sono dei vantaggi anche in lettura: se nel frattempo l'utente decidesse di recuperare parte delle informazioni ancora in memoria, potrebbe accedervi direttamente e senza passare attraverso le prestazioni limitate del disco.

Ogni bella medaglia ha però il suo rovescio e in questo caso esso è rappresentato dalla possibilità di perdere i dati parcheggiati in memoria in caso di mancanza della tensione di alimentazione o di spegnimento della macchina senza utilizzare l'apposita procedura di reboot o shutdown.

Per minimizzare tale possibilità, ogni 30 secondi un apposito programma (in realtà nelle nuove versioni di Linux si tratta di un processo generato automaticamente dal kernel) si occupa di scrivere su disco i buffer parcheggiati in memoria, tentando di rallentare il meno possibile il lavoro degli utenti. In caso di spegnimento brutale del sistema vi è anche la possibilità di danneggiare seriamente il filesystem.

Per questo ad ogni riavvio del sistema un apposito programma (fsck) si occupa di riparare eventuali filesystem danneggiati.

I file che rappresentano periferiche hardware

Ne abbiamo già visto alcuni esempi, ma può valere la pena scrivere una tabella dei file che permettono di accedere alle periferiche hardware più spesso utilizzate in Linux. La lista completa si trova all'interno dei sorgenti del kernel in `/usr/src/linux/Documentation/devices.txt`.

/dev/console	console di sistema
/dev/ttyn	console virtuali (ALT+Fn)
/dev/ttySn	porte seriali (/dev/ttyS0 corrisponde a COM1: in DOS)
/dev/lpn	porte parallele (/dev/lp0 corrisponde a LPT1: in DOS)
/dev/fdn	floppy disk (/dev/fd0 corrisponde ad A: in DOS; /dev/fd0H1440 per specificare capacità)
/dev/hdmn	hard disk (/dev/hda corri sponde all'intero disco, /dev/hdan ad una partizione)
/dev/sdmn	dischi SCSI
/dev/audio	scheda audio
/dev/null	accetta qualunque input
/dev/zero	genera infiniti caratteri con codice ASCII zero
/dev/cdrom	link simbolico al CD-ROM
/dev/mouse	link simbolico al mouse
/dev/modem	link simbolico al modem

Come si nota dalla tabella, i programmi di installazione delle varie distribuzioni di Linux creano in /dev dei link simbolici (collegamenti fra file) ai file speciali dove si trovano CD-ROM, mouse e modem. In questo modo i programmi utente per lavorare correttamente non hanno bisogno di conoscere dove realmente si trovano tali periferiche.

Ad esempio avendo un CD-ROM SCSI esso sarebbe accessibile come /dev/scd0, mentre disponendo di una periferica IDE/ATAPI si dovrebbe utilizzare /dev/hdd. L'utilizzo del link risolve una volta per tutte tale problema.

È interessante anche notare come non tutte le periferiche hardware possano essere utilizzate per contenere un filesystem, ma solamente quelle gestibili "a blocchi" (hard disk, floppy disk, ...) e non quelle gestite "a carattere" (terminali, porte seriali e parallele, ...).

Creazione di un filesystem

Per creare un filesystem è necessario utilizzare il comando mkfs. Il supporto deve già essere formattato a basso livello, al contrario che in DOS e Windows, dove entrambe le operazioni di formattazione fisica e creazione del filesystem vengono svolte dallo stesso comando. Ad esempio per creare un filesystem di tipo ext2 nel dischetto è necessario dare i seguenti comandi:

```
fdformat /dev/fd0H1440
mkfs -t ext2 /dev/fd0
```

Il parametro che segue -t indica il tipo di filesystem da creare.

A questo punto il filesystem può essere montato come abbiamo visto in precedenza.

Si noti che per accedere al floppy durante la formattazione fisica non è stato utilizzato il file speciale `/dev/fd0`, bensì la sua variazione `/dev/fd0H1440` che fornisce al kernel informazioni sulla capacità a cui si vuole utilizzare il dischetto, in caso contrario `fdformat` non avrebbe saputo quale capacità utilizzare.

Una volta formattato fisicamente il disco, Linux è in grado di riconoscerne automaticamente il formato e perciò si può utilizzare semplicemente `/dev/fd0`.

Il caso degli hard disk è leggermente più complesso, in quanto è necessario prima utilizzare il comando `fdisk` (o una delle tante variazioni, ad esempio `cfdisk`) per partizionare il disco. Una volta partizionato il disco, si può creare un filesystem per ognuna delle partizioni create.

File e permessi

Ogni file presente nel sistema appartiene ad un particolare utente che l'ha creato e ad un gruppo ed ha associati tre gruppi di permessi (più uno speciale, vedere Figura 1), che si riferiscono rispettivamente all'utente proprietario (user o owner), agli utenti appartenenti al gruppo proprietario (group) ed agli altri utenti del sistema (other).

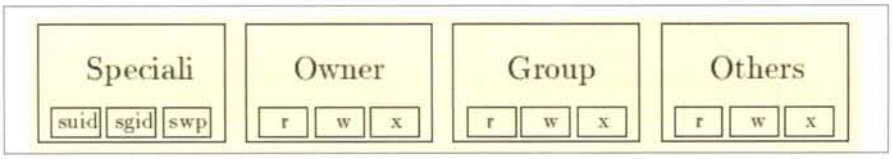

Figura 1 - I permessi associati ad un file.

Si possono vedere i permessi utilizzando l'opzione -l del comando ls:

```
# ls -l
drwxr-xr-x   2 beppe    users        1024 Dec 26   1996 documenti
drwxr-xr-x   2 beppe    users        1024 Dec 26   1996 esempi
lrwxrwxrwx   1 beppe    users           6 Nov  7   1998 doc -> documenti
-rw-r--r--   1 beppe    users       23512 Dec 11   1998 lettera.txt
-rw-r--r--   1 beppe    users      431607 Jul 15   1996 zimage.2.0
```

Il primo carattere della prima colonna di dati rappresenta il tipo del file:

-	file ordinario
d	directory
l	link simbolico
b	periferica gestita a blocchi
c	periferica gestita a carattere

Continuando con l'analisi della prima colonna si trovano i permessi, raggruppati a tre a tre (user, group, other). Essi indicano la possibilità di leggere (r), scrivere (w) ed eseguire (x) il file. Ovviamente questo ultimo permesso è inutile nel caso il file in questione non sia un programma.

Il simbolo - al posto delle lettere rwx indica che tale permesso non è abilitato.

La possibilità di creazione o cancellazione di un file è competenza del permesso w della directory in cui esso è contenuto. Nel caso delle directory il permesso x rappresenta la possibilità di attraversare ("fare cd") la directory. Il campo r corrisponde infine alla possibilità di listare il contenuto della directory.

Il numero nella seconda colonna indica il conto dei link non simbolici al file (ne parleremo fra poco), mentre la terza e quarta colonna contengono l'indicazione di username e gruppo proprietari. Seguono la dimensione in byte, la data di ultima modifica (con opportuni parametri di ls è possibile ottenere anche le date di creazione e ultimo accesso) ed infine il nome del file.

Nel caso del file `lettera.txt` dell'esempio, i permessi sono i seguenti: `-rw-r--r--` che vanno letti nel seguente modo: il simbolo - nella prima posizione indica che si tratta di un file ordinario (`-`),con permessi `rw-` (lettura e scrittura) per il proprietario (beppe), `rw-` per gli utenti appartenenti al gruppo (users) e `r--` (sola lettura) per tutti gli altri.

0	`---`	4	`r--`
1	`--x`	5	`r-x`
2	`-w-`	6	`rw-`
3	`-wx`	7	`rwx`

Figura 2 - Tabella di corrispondenza fra permessi e valori ottali.

Per modificare un permesso si utilizza il comando chmod, con la seguente sintassi:

```
chmod  permesso file
```

I tre gruppi di permessi vanno indicati con l'equivalente numero ottale, come spiegato nella tabella in Figura 2. Una regola semplice per ottenere il numero ottale corretto partendo dai permessi è quella, partendo da zero, di aggiungere 4 per il permesso di lettura, 2 per quello di scrittura e 1 per quello di eseguibilità.

Ad esempio per assegnare al file lettera.txt tutti i permessi all'utente proprietario, il solo permesso di lettura al gruppo e nessun permesso agli altri utenti, si utilizzerà:

```
chmod 740 lettera.txt
```

Il numero 7 corrisponde infatti a `rwx`, il 4 a `r--` e lo `0` a `---`.

Un metodo alternativo all'utilizzo dei numeri ottali è quello di specificare l'iniziale corrispondente al gruppo di permessi che si intende modificare (u=user, g=group, o=other), uno dei simboli + o - a seconda che si voglia concedere o togliere il permesso e l'iniziale corrispondente al nome del permesso (r=read, w=write, x=execute).

Con questa sintassi il comando

```
chmod g-w lettera.txt
```

toglie il permesso di scrittura agli utenti del gruppo users, mentre

```
chmod u+r lettera.txt
```

abilita il permesso di lettura all'utente proprietario.

Agendo opportunamente sui permessi si può rendere il file non leggibile o non scrivibile ad altri utenti, oppure limitarne l'accesso ad un gruppo determinato di persone.

I comandi chown e chgrp permettono di modificare rispettivamente l'utente proprietario e il gruppo di un file:

```
chown utente file
chgrp gruppo file
```

Ovviamente essi possono essere eseguiti solamente dal proprietario attuale del file o da root.

Utilizzo dei permessi

Per condividere fra un gruppo ristretto di utenti (beppe e lorenz) uno stesso file è necessario creare un gruppo (project x) ed inserirvi gli utenti desiderati:

```
groupadd -g 113 projectx
gpasswd -a beppe projectx
gpasswd -a lorenz projectx
```

Al termine di queste operazioni avremo in /etc/group la seguente linea:

```
projectx:*:113:beppe,lorenz
```

per accedere ai permessi di questo gruppo gli utenti beppe e lorenz dovranno utilizzare il comando:

```
newgrp projectx
```

Volendo è possibile anche assegnare una password al gruppo, in modo che anche eventuali altri utenti oltre ai due inseriti possano acquisirne i permessi usando newgrp projectx, ma solo dopo aver inserito la password corretta:

```
gpasswd projectx
```

Per fare in modo che un file sia accessibile solamente dagli utenti del gruppo projectx è sufficiente renderlo di proprietà di tale gruppo e settare opportunamente i permessi in modo che sia leggibile e scrivibile solo al proprietario ed agli utenti del gruppo:

```
chgrp projectx relazione.txt
chmod 660 relazione.txt
```

Lo stesso risultato può essere ottenuto assegnando opportunamente il gruppo di default degli utenti in /etc/passwd. In questo caso, trattandosi del gruppo di default, non servirà creare un nuovo gruppo e gli utenti non dovranno utilizzare il comando newgrp per ottenere i dovuti permessi.

Link e link simbolici

I link ordinari sono dei riferimenti ad altri file, che permettono di non fare copie inutili degli stessi dati. Poiché si duplica l'informazione relativa alla posizione del file sul medesimo disco, il link può essere effettuato solamente all'interno di uno stesso filesystem e non, ad esempio, fra due dischi diversi.

Nel caso dei link simbolici, o symlink, invece l'informazione che viene conservata è il percorso del file a cui puntano, che perciò può risiedere in qualunque punto del filesystem della macchina, anche su un disco diverso.

Nella lista prodotta da `ls -l`, i link simbolici si riconoscono perché vengono identificati come file di tipo "l" ed è presente, preceduta dal simbolo ->, l'indicazione del nome del file a cui puntano.

Nella lista seguente ci sono quattro file linkati fra loro ed un link simbolico. A causa dell'eccessiva larghezza della lista, ho riportato solamente le informazioni utili e sostituito le rimanenti con dei puntini.

```
-r-sr-xr-x 5 ... /usr/bin/lmail
-r-sr-xr-x 5 ... /usr/bin/mail
-r-sr-xr-x 5 ... /usr/bin/mailx
lrwxrwxrwx 1 ... /usr/bin/rmail -> /bin/rmail
-r-sr-xr-x 5 ... /usr/bin/wmail
```

Per realizzare un link si usa il comando:

```
ln file link
```

Per realizzare un link simbolico è necessario aggiungere al comando precedente l'opzione -s:

```
ln -s file symlink
```

Come per `cp` o `mv`, si deve specificare come primo parametro il nome del file che esiste già, poi il nome di quello che si vuole creare.

Cancellare un file a cui è linkato un link simbolico non modifica il link ma causa un errore quando si tenta di utilizzarlo, poiché il file corrispondente non viene trovato. Per cancellare un file linkato in modo non simbolico bisogna invece prima cancellare tutti i link ad esso relativi.

Conclusioni

Nelle prossime puntate analizzeremo altri aspetti fondamentali di Linux: la gestione dei processi e la shell, ovvero l'interfaccia con cui l'utente impartisce ordini al sistema. MC