

# Dischi e Filesystem

Radu Andries

24 Maggio 2011

## Introduzione

Se non si capisce qualcosa subito, il concetto è spiegato più avanti. Ho spiegato tutti i concetti che ho usato, ma data l'interdipendenza tra le varie parti ho dovuto fare un ordine particolare, quindi consiglio di leggere questo documento 2 volte, almeno la prima parte

## Filesystem hierarchy standard (FHS)

Il FHS è stato ideato per rendere simile la struttura delle directory sotto le varie distribuzioni Linux. Non tutte le distribuzioni lo seguono alla lettera ma le differenze di solito sono minime. Ecco a voi un elenco di alcune directory importanti sotto Linux.

- /bin - Comandi/eseguibili essenziali per il sistema.
- /boot - Cartella contenente i file necessari al boot del sistema. Qui ci finiranno il kernel ed altri file necessari (file di configurazione del bootloader ad esempio). Quando il sistema è acceso non è necessaria la presenza di questa directory. Al riavvio però potrebbero capitare brutte cose(tm).
- /dev - In questa directory ci sono \*tutti\* i device visti da linux. Sono sotto forma di file (ricordate che in unix/linux tutto è un file?). Vedere Udev.
- /etc - file configurazione sistema.
- /home - Cartella che contiene le directory di ogni utente. Le cartelle utente (di tipo /home/\$utente) contengono anche la configurazione delle applicazioni usate. La configurazione relativa all'utente ha priorità su quella di sistema. Ma non dimentichiamo i permessi...
- /lib - Librerie di sistema. Qui ci sono i firmware per i dispositivi, i moduli del kernel e le librerie necessarie per i programmi in /bin

- /proc - procs. Prossima lezione... (per i curiosi, file relativi ad ogni processo in esecuzione)
- /media - I dischi montati dal sistema(chiavette usb,dischi esterni, memory card) sono accessibili qui in directory del tipo /media/usbkey, /media/c-drom ecc.
- /mnt - punto temporaneo di mount.
- /sbin - vedere bin, solo che questi di solito sono di amministrazione.
- /tmp - file temporanei. Vengono zappati ogni avvio. Una pratica comune per migliorare le prestazioni è montare questa dir in ram (tmpfs)
- /usr - Gerarchia secondaria per dati utenti di sola lettura (programmi)
  - /usr/bin - binari dei programmi
  - /usr/include - headers dei programmi e di libstdc ecc
  - /usr/lib - librerie (\*.so) richieste dai progammi
  - /usr/share - Dati programmi
  - /usr/src - Sorgenti dei programmi, di solito già scompattati. La più importante è /usr/src/linux, che è un collegamento simbolico (symlink) alla cartella dei sorgenti del kernel che è in esecuzione
- /var - dati dinamici delle applicazioni (ma che sono persistenti tra riavvii)

## Kernel e Dischi

### Udev

Udev è il gestore delle periferiche di linux. È stato introdotto dalla versione 2.5 di Linux a sostituire devfs. Si occupa principalmente di gestire l'aggiunta e rimozione di periferiche in /dev in risposta anche alle azioni degli utenti.

Dopo che il Linux ha avviato init , init provvede a chiamare udev che provvede a popolare la cartella /dev con i device collegati al sistema. Ci sono 2 binari importanti:

- udevd - Daemon di udev. deve essere in esecuzione sempre
- udevadm - comandi a udev. Ad esempio con “udevadm trigger” viene detto ad udev di rielaborare immediatamente gli eventi relativi al hardware. “man udevadm” per più infomazioni.

## Tipi di device

Ci sono 2 tipi di device, a caratteri e a blocchi.

- a caratteri, l'I/O avviene carattere per carattere - esempio porta seriale (/dev/tty\*)
- a blocchi, DMA - i dischi (/dev/sd\*)

## Tipi di Filesystem

Linux ha supporto per molti filesystem. Ve ne elencherò qualcuno che incontrerete.

### **ext3**

Il filesystem più usato dai computer con su linux. Si consiglia di usare il più nuovo ext4.

### **ext4**

ext4 è un filesystem ad alte prestazioni dotato di journaling. Questo vuol dire che è capace di recuperare dopo un arresto improvviso senza sincronizzazione del disco.

### **reiserfs**

Un altro filesystem con journaling.

### **btrfs**

È la risposta di linux a zfs. È un ottimo filesystem, ma purtroppo è ancora in sviluppo.

### **tmpfs**

Usa la ram per i file scritti su di esso.

### **devtmpfs**

alternativa a udev che dovrebbe rendere l'avvio di linux più veloce, ma perdendo alcune funzionalità

### **sysfs,procfs**

servono al sistema per interfacciarsi con il kernel linux

## unionfs,aufs

Servono per creare livelli di filesystem uno sopra l'altro, con solo un livello scrivibile.

## Dischi in /dev

In linux i dischi vengono rappresentati in /dev come sdXY dove X(numero disco) è una lettera da "a" a "z" e Y(numero partizione) un numero intero partendo da 1. Quindi il disco 1 del sistema sarà /dev/sda e la partizione 3 (se esiste) è /dev/sda3.

Il sistema usa /dev/sdX per leggere e modificare MBR e tabella di partizioni...

## I file

Tutto in Linux è un file. Anche le directory sono file. Di tipo directory. I device sono file (guardare /dev).

Ok ma tutto questo come si gestisce?

## I permessi UNIX

Ogni file ha 3 tipi di permessi: proprietario, gruppo e tutti gli altri. Quando si fa "ls -l" compare una lista dei file con i loro attributi. Ora spiegherò cosa rappresenta ogni colonna.

```
totale 36
drwxr-xr-x 2 root      root   4096 22 mag 15.15 cron.t8VqrG/
-rw-rw-r-- 1 admiral0 users    5 22 mag 15.24 file
drwx----- 2 admiral0 users  4096 22 mag 15.17 kde-admiral0/
drwx----- 2 admiral0 users  4096 22 mag 15.17 ksocket-admiral0/
drwx----- 2 root      root   4096 22 mag 15.18 libgksu-m3qts4/
drwx----- 2 admiral0 users  4096 22 mag 15.18 orbit-admiral0/
drwx----- 2 admiral0 users  4096 22 mag 15.16 pulse-PKdhtXMmr18n/
-rw-rwxr-- 1 admiral0 users    5 22 mag 15.24 script*
drwxrwxr-x 2 admiral0 audio  4096 22 mag 15.23 temp/
```

Ecco cosa vogliono dire le colonne:

1. I permessi e tipo di file. d vuol dire che è una directory. Il resto sono permessi in formato rwx per il proprietario, il gruppo e per tutti gli altri. (r=readable→lettura permessa, w=writable→scrittura permessa, x=eXecution→esecuzione permessa)
2. Numero di link (non ci interessa più di tanto)

3. Proprietario
4. Gruppo
5. Dimensione in byte
6. Data creazione
7. Nome

## Masks

Per definire i permessi unix invece di rwx si può usare la notazione in ottale. Sono 4 cifre. Gli zeri iniziali si possono anche omettere.

1. 4→setuid, 2→setgid, 1→sticky
2. Diritti proprietario 4→read, 2→write, 1→execute
3. Diritti gruppo.
4. Diritti altri

Non spiegherò la prima cifra in quanto viene usata raramente da un utente. Basta dire che se vedete dei binari che non sono del sistema che sono setuid o setgid non va per niente bene(tm)

Per ottenere la “maschera” in ottale basta sommare i diritti, quindi rwxr-xr- viene tradotto in 0754 o più corto 754.

## Gestione permessi

La gestione permessi su linux può essere effettuata da qualsiasi file manager sotto linux. Ci sono però dei comandi dedicati per gestirli da linea di comando.

### chown

Serve per cambiare proprietario e/o gruppo di un file/directory. Ecco le opzioni

```
chown proprietario : gruppo nomefile
```

Può essere usato “-R” per rendere ricorsivo il comando ed applicarlo su tutte le directory e tutti i file sottostanti (se “nomefile” è una directory)

### chmod

Cambia permessi d’accesso al file.

```
chmod [-R] [u|g|o]+|-r|w|x nomefile
```

Dove u=proprietario, g=gruppo, o=altri. I permessi possono essere specificati anche in ottale.

Esempio:

```
chmod u-x nomefile
```

Rimuove diritti di esecuzione per il proprietario di nomefile

```
chmod -R go+w nomedirectory
```

Aggiunge i diritti di scrittura per il gruppo e per gli altri della directory “nomedirectory” e tutti i suoi contenuti.

## **dd**

Serve per copiare file, ma dato che tutto è un file è un comando tuttofare, permette di clonare l'intero hard disk, blocco per blocco.

```
dd if=input of=output [bs=dimensione] [count=numero]
```

bs serve a dire quanti bytes copiare alla volta e count quante volte ripetere. input e output possono essere qualsiasi file che non sia una directory

## **Permessi ACL**

Acl è un modo diverso di avere i permessi su Linux. E' molto meno usato, ed è molto complesso. Permette di specificare diritti personalizzati per molti più utenti e gruppi, non solo utente e gruppo dell'appartenenza al file. Per info leggere la seguente pagina:

<http://en.wikipedia.org/wiki/Richacls>

# **Amministrazione dei dischi**

Ci sono vari programmi per amministrare i dischi. Vi mostrerò i più comuni e facili da usare.

## **Amministrazione ordinaria**

### **Archiviazione**

Sotto linux è possibile comprimere e decomprimere quasi qualsiasi archivio, infatti per rar e zip bastano i programmi unrar e unzip. Nel mondo linux però vengono usati i tarball. Cosa sono i tarball? Archivi tar compressi. (tar semplicemente crea l'archivio, non comprime). Eccone un paio di molto usati e come comprimere e decomprimere.

### **tar.gz**

Il più comune, ma anche il meno compresso

```
#decomprimere
tar xzf archivio.tar.gz
#oppure
gunzip archivio.tar.gz
tar xf archivio.tar
#comprimere
tar czf file.tar.gz cartella altracartellasevoglio
#oppure
tar cf file.tar cartella altracartellasevoglio
gzip file.tar
```

### **tar.bz2**

Più compresso del gz, ma richiede più CPU per compressione e decompressione

```
#decomprimere
tar xjf archivio.tar.bz2
#oppure
bunzip2 archivio.tar.bz2
tar xf archivio.tar
#comprimere
tar cjf file.tar.bz2 cartella altracartellasevoglio
#oppure
tar cf file.tar cartella altracartellasevoglio
bzip2 file.tar
```

### **tar.xz**

L'algoritmo lzma, il migliore ed il più nuovo che ci sia in giro (usato anche per i \*.7z)

```
#decomprimere
tar xJf archivio.tar.xz
#oppure
unxz archivio.tar.xz
tar xf archivio.tar
#comprimere
tar cJf file.tar.xz cartella altracartellasevoglio
#oppure
tar cf file.tar cartella altracartellasevoglio
xz file.tar
```

### **Quali fs ho montati sul sistema?**

Niente di più facile. Basta eseguire "mount" oppure "df -h".

```

$ mount
proc on /proc type proc (rw,nosuid,nodev,noexec,relatime)
sys on /sys type sysfs (rw,nosuid,nodev,noexec,relatime)
run on /run type tmpfs (rw,nosuid,nodev,noexec,relatime,size=10240k)
/dev/mapper/snowflakehdd-root on / type ext4 (rw,commit=600)
devpts on /dev/pts type devpts (rw,relatime,mode=600,ptmxmode=000)
shm on /dev/shm type tmpfs (rw,nosuid,nodev,relatime)
debugfs on /proc/sys/debug type debugfs (rw)
/dev/mapper/snowflakehdd-home on /home type ext4 (rw,commit=600)
fusectl on /sys/fs/fuse/connections type fusectl (rw)

```

Tra parentesi sono le opzioni con cui sono montati i filesystem. Per df il parametro h serve per dare un output “umano” specificando GB o Mb invece di byte.

```

$ df -h
File system                Dim. Usati Dispon.  Uso% Montato su
udev                      10M    0      10M   0% /dev
run                       10M 496K    9,6M   5% /run
/dev/mapper/snowflakehdd-root 31G  20G    10G  66% /
shm                       1,8G 708K    1,8G   1% /dev/shm
/dev/mapper/snowflakehdd-home 207G 168G    29G  86% /home

```

## mount

Mount serve per “agganciare” un filesystem ad una directory nella /. Il sistema all’avvio legge il file /etc/fstab e monta tutti i filesystem che sono descritti.

Il formato è

```
mount [-t tipofs] device directory [-o opzioni_fs]
```

nel caso di tmpfs,sysfs,proc il device è none

## umount

Serve per smontare un filesystem

```
umount directory_o_device
```

## fsck

Serve per verificare la consistenza di un filesystem. Il filesystem deve essere smontato, altrimenti possono capitare danni al filesystem.

```
fsck device
```

## losetup

crea e amministra device di loop. I device di loop servono a montare immagini ISO, immagini di hard disk ed altro.

## Amministrazione straordinaria

### mkfs.<fs>

Crea un filesystem su una partizione. Dopo il punto va il tipo di filesystem che si vuole creare. Per esempio mkfs.ext4 crea un filesystem ext4.

```
$ mkfs.ext4 /dev/sdd1
```

**Attenzione:** Se sbagliate device potreste formattare la vostra partizione di sistema e quindi dire addio al sistema in un attimo.

### cgdisk

Serve per partizionare. È un po' schizzinoso per le tabelle di partizioni, ma è molto facile da usare. Nel caso si rifiuti di modificare la tabella di partizioni si deve usare fdisk

### fdisk

Il programma più usato per partizionare. È difficile da usare.

### parted

Un'alternativa ai due di sopra. Ha una riga di comando e dei comandi tutti suoi.

### photorec, dd\_rescue

Servono rispettivamente a recuperare file cancellati e clonare/copiare hard disk difettosi. Consultare le manpage per dettagli.

## Extra

### LVM2 - Logical Volume Manager

Lvm2 è un modo alternativo per gestire le partizioni su Linux. Permette di fare backup dell'intero disco, ridimensionare le partizioni, fare snapshot, aggiungere dischi muovere partizioni ecc , tutto senza dover spegnere il computer anche se è il filesystem /. LVM2 è diviso in 3 unità logiche

- physical volume (pv) - Sono i volumi fisici, lo spazio fisico disponibile.
- volume group (vg) - Sono raggruppamenti di volumi fisici.
- logical volume (lv) - Sono le partizioni logiche. I filesystem si creano sui lv.

Quindi per creare un setup lvm servono dei moduli del kernel abilitati (device-mapper) e dei hard disk liberi :). Ecco a voi un esempio semplice.

```
$ #Si crea una sola partizione sda1
$ cfdisk /dev/sda
$ #eventuali altri dischi
$ cfdisk /dev/sdb
$ #si creano i physical volume
$ pvcreate /dev/sda1 /dev/sdb1
$ #si crea il vg
$ vgcreate ilmiovg /dev/sda1 /dev/sdb1
$ #Si crea un lv di dimensione di 1 Gb
$ lvcreate -n lamiapartizione -L 1G ilmiovg
$ #ecc ecc
$ lvcreate -n altrolv -L 512M ilmiovg
$ #Formatto il primo lv con ext4
$ mkfs.ext4 /dev/mapper/ilmiovg-lamiapartizione
$ mkfs.btrfs /dev/mapper/ilmiovg-altrolv
$ #nessuna differenza per montare
$ mount /dev/mapper/ilmiovg-altrolv /mnt
```

## Amministrazione base di un lv

Ecco i comandi per gestire in modo base un lv.

```
$ #estendiamo un lv di 5 Gb
$ lvextend -L +5G /dev/mapper/ilmiovg-lamiapartizione
$ #ATTENZIONE: non dimentichiamoci del filesystem.
$ #          non tutti supportano l'estensione se sono montati.
$ #          ext4 sì :D
$ resize2fs /dev/mapper/ilmiovg-lamiapartizione
$ # back to normal!
$ umount /mnt
$ resize2fs /dev/mapper/ilmiovg-lamiapartizione 1G
$ lvreduce -L 1G /dev/mapper/ilmiovg-lamiapartizione
```

## Dire addio ad un lv

```
$ lvremove /dev/mapper/ilmiovg-lamiapartizione
```

## LUKS - Linux Unified Key Setup

LUKS serve per crittografare interi device o partizioni. Di solito si usa insieme ad lvm per creare più partizioni su un unico device cryptato. Ecco come si crea e si opera con un device LUKS.

```
$ #formattiamo un device e prepariamolo per LUKS
$ dd if=/dev/urandom of=/dev/sdd1
$ cryptsetup luksFormat /dev/sdd1
$ # Montiamo il device dandogli un nome
$ cryptsetup luksOpen /dev/sdd1 ilmiodrive
$ #formattiamolo se serve (solo la prima volta di solito)
$ mkfs.ext3 /dev/mapper/ilmiodrive
$ #montiamo
$ mount /dev/mapper/ilmiodrive /mnt
$ #tutto quello che mettiamo in mnt viene cryptato
$ #chiudiamo baracca
$ umount /mnt
$ cryptsetup luksClose ilmiodrive
```

## RAID - Redundant Array of Independent Disks

Il RAID software serve per quelli che hanno bisogno di RAID, ma dove non è possibile usare RAID fisico (a livello chipset). Vi consiglio caldamente di usare RAID fisico in quanto rende la CPU più libera. Ci sono molti tipi di RAID, ve ne spiegherò solo quelli più rilevanti come newbie sysadmin. Per ulteriori info wikipedia è un'ottima fonte.

- RAID-0 o striping - Si fa con un minimo di 2 dischi identici. Si ha la capacità sommata dei 2 device. Il kernel scrive metà dei dati su ciascuno di dischi. Si raddoppia la velocità di trasferimento. Se però uno dei dischi muore e non si ha un backup, condoglianze.
- RAID-1 o mirroring - Si fa con un minimo di 2 dischi identici. Si ha la capacità di un singolo device. Il kernel scrive i dati su entrambi gli hard disk.
- RAID-5 - Si fa con un minimo di 3 dischi identici. Si ha la capacità sommata di 2 dei 3 device. Il kernel scrive metà sul primo hd, metà sul secondo e sul terzo scrive il bit di parità. Se muore un hd si è tranquilli, ma si deve sostituire asap, perché se muore un'altro dei 2 rimasti, di nuovo condoglianze. Si hanno prestazioni quasi a livello di RAID-0.

Vi spiegherò come “assemblare” un device RAID software per poi lavorarci su.

```
$ #questo assembla i RAID noti al kernel
$ mdadm Auto-detect
```

```
$ #et voilà. Inizia sempre con md, il numero può essere diverso  
$ mount /dev/md0 /mnt
```