Guida ai comandi base della shell in GNU/Linux

Da Linux@P2P Forum Italia.

Vai a: navigazione, ricerca

La **Bash** (**B**ourne **A**gain **Sh**ell) è la più diffusa shell (linea di comando) nei sistemi GNU/Linux, è uno strumento tanto potente quanto pericoloso se usato in modo improprio. Vediamo di dare qualche indicazione sull'utilizzo dei principali comandi. Per avere una breve introduzione su cosa è una linea di comando, come accedervi e quali le principali convenzioni (fondamentali da conoscere se vogliamo usarla) fare riferimento alla guida d'introduzione all'uso della shell.

Premessa: molti comandi per essere eseguiti necessitano dei privilegi di amministratore (utente root), pertanto prima bisogna:

- Se si usa Ubuntu/Kubuntu/Xubuntu: anteporre ai comandi **sudo**, nel seguente modo:

```
sudo comando
(inserire la password dell'utente che non apparirà a video neanche come asterischi!)
```

- Se si usa una qualsiasi altra distro: loggarsi come root

```
su
(inserire la password di root che non apparirà a video neanche come asterischi!)
```

Indice

[nascondi]

Operazioni comuni

Muoversi fra le directory

Il comando **cd** ci permette di muoversi fra le directory, il uso utilizzo è molto semplice, volendo accedere ad una directory contenuta nella cartella in cui siamo:

```
cd Desktop
```

Volendosi spostare direttamente in una sua sottodirectory (anche infiniti livelli di sottodirectory, non c'è un limite) basta dare:

```
cd Desktop/emilio/amule/
```

Possiamo sia inserire il percorso relativo alla nostra posizione (come abbiamo fatto prima, eravano nella nostra home e siamo entrati in una directory in essa contenuta) oppure possiamo muoverci attraverso percorsi assoluti, esempio la directory Desktop, puo anche essere raggiunta così:

```
cd /home/ercoppa/Desktop
```

Ossia partendo a calcolare il percorso fin dalla / (radice di sistema)

Visualizzare il contenuto di una directory

Per visualizzare il contenuto di una directory (e non solo) utilizziamo il comando **ls**, l'uso più semplice è:

```
ls
```

che restiruirà il contenuto (directory, file e link) della directory in cui ci troviamo. Ma **ls** ci puo fornire molte più informazioni infatti associando al comando una o più delle sue opzioni (consultabili con *man ls*) è possibile avere molti più dettagli. Partiamo con le opzioni **-al**

```
ercoppa@gentoo ~/Desktop $ ls -al
 drwxr-xr-x  2 tizio ufficio      80 2006-01-24 14:20 sottodirectory
 -rw-r--r--  1 tizio ufficio    3655 2006-01-07 18:51 questo.txt
 -rw-rw-r--  3 tizio ufficio    5482 2005-11-15 12:41 quello.ods
 -rw-r--r--  4 caio  ufficio    1224 2005-06-27 18:15 quell'altro.jpeg
 -rwxr-x---  3 root  ufficio     458 2006-04-12 10:38 eseguibile
 lrw-rw-r--  3 tizio ufficio      19 2006-04-12 21:52 Pubblic ->
/home/samba/pubblic
```

Analizziamo questo output (partendo da destra):

```
drwxr-xr-x
```

Il primo carattere (o meglio byte) da destra puo indicare:

- **d** ossia che è una directory
- **-** ossia che è un file
- **l** ossia che link (simbolico)
- **b** per un dispositivo speciale a blocchi
- **c** per un dispositivo speciale a caratteri

- **p** per un FIFO
- **s** per un socket

Gli altri 9 caratteri che seguono, indicano i permessi assegnati su quel file rispettivamente:

- i primi tre (in rosso) indicano i permessi dell'utente proprietario (user) di quel file
- i successivi tre (in blu) indicano i permessi assegnati al gruppo (group) (ossia tutti gli utenti appartenenti a quel gruppo)
- gli ultimi 3 caratteri (in verde) indicano i permessi assegnati a tutti gli altri utenti (others)

I permessi per ogni soggetto (user, group, others) possono essere di tre tipi:

- **r** per indicare la lettura (in caso di directory è possibile vedere la lista dei files contenuti)
- **w** per indicare la modifica (o eliminazione)
- **x** per indicare il permesso di eseguire il file

La presenza del segno **-** indica l'assenza di quel permesso

Successivamente abbiamo ad esempio

```
2 tizio ufficio      80 2006-01-24 14:20
```

- **2** rappresenta il numero di collegamenti (link) verso quel file
- tizio è il proprietario (user) del file
- ufficio è il gruppo a cui appartiene il file
- **80** la dimensione in byte
- **2006-01-24** è la data (e eventualmente anche l'ora) di creazione o ultima modifica del file

L'ultima indicazione è riguardo al nome del file o della directory. Se un link simbolico, viene indicato anche la cartella o file a cui è collegato.

```
Pubblic -> /home/samba/pubblic
```

Altre opzioni interessanti per ls sono:

- **-lt** che mostra secondo la data di ultima modifica (dal più recente al più vecchio):

```
ercoppa@gentoo ~ $ ls -lt
totale 18652
drwxr-xr-x  5 ercoppa ercoppa    4096  6 mag 22:58 Desktop
drwxr-xr-x  9 ercoppa ercoppa    4096  6 mag 22:46 emesene
-rw-r--r--  1 ercoppa ercoppa      65  4 mag 01:54 svn
-rw-r--r--  1 ercoppa ercoppa 1150562 26 apr 18:40 prova
-rw-r--r--  1 ercoppa ercoppa  793869 25 apr 21:46 the free software song2.mp3
drwxr-xr-x 34 root    root       4096 12 apr 21:54 winex
```

- **-lR** che elenca ricorsivamente tutte le sottodirectory incontrate:

```
ercoppa@gentoo ~/Desktop/amule $ ls -lR
.:
-rw-r----- 1 ercoppa ercoppa    1343614 27 mar 16:39 Manuale Del Linguaggio C.pdf
./[ebook - ITA] Manuale del Linguaggio C.zip_FILES:
totale 5692
-rw-rw-rw- 1 ercoppa ercoppa 2612224 20 set  1999 trickyc.doc
-rw------- 1 ercoppa ercoppa  679716 31 ott  2001 trickyc.doc.zip
```

```
-rw-rw-rw- 1 ercoppa ercoppa 1358512  8 mag  1999 trickyc.pdf
-rw------- 1 ercoppa ercoppa 1156807 31 ott  2001 trickyc.pdf.zip
```

Copiare un file

Per copiare file (comprese le directory) possiamo usare il comando **cp**, il suo uso più elementare è:

```
cp file_da_copiare destinazione
```

Dobbiamo subito far notare che cp non avverte in caso esists già un file con quel nome, quindi è consigliabile usare l'opzione **-i** che abilita la richiesta di conferma (viene posta una domanda) in caso di sovrascrittura.

```
ercoppa@gentoo ~ $ cp -i file_uno Desktop/
cp: sovrascrivo `Desktop/file_uno'? s
```

Un'altra opzione molto utile è **-a**, che conserva nella copia quanto è possibile della struttura e degli attributi dei file originali. Molte utile se si vogliono preservare gli stessi permessi ad esempio durante un backup di sistema.

Da tenere in considerazione è anche l'opzione **-R** che permette la copia ricursiva anche di tutte le sotto-directory (NOTA: **-a** contempla anche **-R**)

Spostare un file

Per spostare i file possiamo usare **mv**, il suo uso più elementare è:

```
mv file_da_spostare destinazione
```

La destinazione ovviamente deve essere una directory (altrimenti il file viene rinominato). Le opzioni da tenere in considerazione principalmente sono:

- **-i**: chiede una conferma per la sovrascrittura nel caso in cui nella destinazione da noi specificata esista già un file (o directory) con quel nome, vediamo un esempio pratico:

```
ercoppa@gentoo ~ $ mv -i file_uno Desktop/
mv: sovrascrivo `Desktop/file_uno'? s
ercoppa@gentoo ~ $
```

- **-v**: stampa il nome di ogni file prima di spostarlo

```
ercoppa@gentoo ~ $ mv -v  Desktop/file_uno ./
`Desktop/file_uno' -> `./file_uno'
ercoppa@gentoo ~ $
```

Rinominare un file

Per rinominare un file (o directory) usiamo il già noto **mv**, la sintassi più elementare è:

```
mv nome_file nuovo_nome_file
```

Per tutelarci dalla sovrascrittura di un file con lo stesso nome possiamo usare l'opzione **-i**:

```
ercoppa@gentoo ~ $ mv -i file_uno file_due
mv: sovrascrivo `file_due'? s
ercoppa@gentoo ~ $
```

Eliminare un file

Per eliminare un file usiamo il comando **rm**, il suo uso più elementare è:

```
rm file
```

Nel caso in cui stessimo tentando di cancellare directory abbiamo due possibilità:

- usare **rm** con le opzioni **-rf**:

```
ercoppa@gentoo ~ $ rm cartella
rm: impossibile rimuovere `cartella': Is a directory
ercoppa@gentoo ~ $ rm -rf cartella
```

- se la directory è vuota, usare rmdir

```
ercoppa@gentoo ~ $ rmdir cartella
```

Come per **cp** e **mv** è consigliabile usare un opzione di sicurezza come **-i** che chiede conferma per ogni file da cancellare:

```
ercoppa@gentoo ~ $ rm -i file_uno
rm: rimuovere regular empty file `file_uno'? s
```

o nel caso di directory (non vuota):

```
ercoppa@gentoo ~ $ rm -rfi cartella
rm: entrare nella directory `cartella'? s
rm: rimuovere regular empty file `cartella/file_uno'? s
rm: rimuovere regular empty file `cartella/file_due'? s
rm: rimuovere directory `cartella'? s
```

Come per **cp** e **mv** è disponibile l'opzione **-i** che ci tutela dalla eliminazione dei nostri file chiedendo un ulteriore conferma alla nostra azione:

```
ercoppa@gentoo ~ $ rm -i processo.txt
rm: rimuovere regular file `processo.txt'? y
```

Creare un link (simbolico)

Prima di capire come fare un collegamento (d'ora in poi link), bisogna fare una precisazione: in GNU/Linux (e non solo) esistono due tipologie di link:

- simbolico (soft link): è quello che siamo soliti incontrare anche in ambiente windows ed è quello che continueremo ad usare, salvo specifiche necessità. Per una definizione più ampia leggere qui
- hard link: è qualcosa di più complesso, che lega il link al contenuto stesso (inode) del file collegato. Per una definizione più corretta e ampia leggere qui

Per creare un link simbolico dobbiamo usare il comando **ln** con l'opzione **-s**, il suo utilizzo è abbastanza semplice:

```
ln -s file nome_collegamento
```

Ecco un esempio:

```
ercoppa@gentoo ~ $ ln -s file_due gianni
ercoppa@gentoo ~ $ ls -l
lrwxrwxrwx  1 ercoppa ercoppa       8  8 mag 00:12 gianni -> file_due
```

Creare una directory

Per creare una directory usiamo il comando **mkdir**, il suo uso più elementare è:

```
mkdir nome_cartella
```

Montare una partizione (renderla accessibile)

Prima di tutto dobbiamo identificare la partizione, per avere delle indicazioni utili fare riferimento a questa guida. Il comando che ci permette di montare una partizione è **mount**, di norma è soddisfacente usarlo nella sua forma più elementare:

```
mount /dev/partizione /media/nome_punto_di_montaggio
```

Ovviamente "/media/nome_punto_di_montaggio" è una cartella (detta punto di montaggio) che abbiamo precedentemente creato con:

```
mkdir /media/nome_punto_di_montaggio
```

Vediamo le opzioni più utili da associare al comando **mount**:

- **-t** (type) che permette di specificare (se il filesystem non venisse automaticamente riconosciuto) il fs della partizione e puo assumere i seguenti valori: adfs, affs, autofs, coda, coherent, cramfs, devpts, efs, ext, **ext2**, **ext3**, hfs, hpfs, iso9660, jfs, minix, msdos, ncpfs, nfs, **ntfs**, proc, qnx4, ramfs, **reiserfs**, romfs, **smbfs**, sysv, tmpfs, **udf** (fs dei cd/dvd multisessione), ufs, umsdos, usbfs, **vfat** (valido per fat, fat16, fat32), xenix, **xfs**, xiafs.

Ecco un esempio del comando con l'opzione **-t**:

```
mount -t vfat /dev/sda1 /media/penna_usb
```

- **-o** (options) che permette di specificare opzioni utili, i volori indicabili sono (citiamo i più comuni):
  - **async**: I/O in modo asincrono
- **atime**: Aggiorna la data/ora di accesso agli inode ad ogni accesso
- **defaults**: usa le impostazioni di default (rw, suid, dev, exec, auto, nouser e async)
- **dev**: Interpreta i device speciali a caratteri o a blocchi del file system
- **exec**: permette l'esecuzione di binari
- **ro**: read-only ossia solo lettura
- **rw**: read-write ossia permessa la lettura e scrittura dei dati
- **sync**: I/O in modo sincrono
- **suid**: Abilita le funzionalità dei bit set-user-identifier o set-group-identifier.
- **user**: abilita l'utente comune a montare la partizione (a senso specificarla solo nel /etc/fstab)
- **users**: abilita l'utente comune a montare e/o smontare la partizione

- **noexec**: non permette l'esecuzione di file binari
- **nodev**: Non interpreta i device speciali a caratteri o a blocchi del file system
- **noatime**: Non aggiorna la data/ora di accesso agli inode ad ogni accesso
- **nosuid**: Disabilita le funzionalità dei bit set-user-identifier o set-group-identifier.
- **nouser**: Non permette allo user di montare la partizione (a senso specificarla solo nel /etc/fstab)
- **umask=valore**: imposta la maschera di permessi, attraverso un valore ottale, guardare qui per maggiori info
- **nls=valore**: impostiamo il "Native Language Support", molto utile per i fs NTFS usare il valore **utf8**

Ecco un esempio dell'uso di **-o**:

```
mount -o exec,async /dev/sda1 /mnt/penna_mia
```

Oppure combinando con l'opzione **-t**:

```
mount -o exec,async -t vfat /dev/sda1 /mnt/penna_mia
```

NOTA IMPORTANTE N°1: Per montare fin dall'avvio del sistema una partizione bisogna agire sul file /etc/fstab, fare riferimento alla Guida a FSTAB per questo.

NOTA IMPORTANTE N°2: Il punto di montaggio puo essere creato in qualunque punto del filesystem, comunque per convenzioni le distro sono solite montare in /media o in alternativa /mnt

Smontare una partizione

Per smontare una partizione possiamo fare in due modi:

- **umount** e l'identificativo della partizione, ad esempio:

```
umount /dev/sda1
```

- **umount** e il punto di montaggio:

```
umount /media/penna_mia
```

Capire quali partizioni sono montate

Il comando **mount** senza nessun altra opzione e argomento, puo fornire dettagli su quali partizioni sono montate, i relativi fs e le opzioni applicate. Quindi basterà dare:

```
mount
```

E avere un output del tipo:

```
/dev/hda5 on / type ext3 (rw,noatime,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw,nosuid,nodev,noexec)
udev on /dev type tmpfs (rw,nosuid)
devpts on /dev/pts type devpts (rw,nosuid,noexec)
none on /var/tmp/portage type tmpfs (rw,size=1000)
/dev/hda3 on /mnt/ubuntu type reiserfs (rw,noexec,nosuid,nodev,user_xattr)
/dev/hda2 on /mnt/gentoo64 type xfs (rw,noatime)
```

```
shm on /dev/shm type tmpfs (rw,noexec,nosuid,nodev)
usbfs on /proc/bus/usb type usbfs (rw,noexec,nosuid,devmode=0664,devgid=85)
```

Leggere un file

Per leggere il contenuto di un file principalmente vengono usati tre comandi:

- **cat** che ci mostra semplicemente il contenuto del file, il suo uso è elementare:

```
cat nomefile
```

- **less** che ci legge il documento pagina per pagina, (è possibile scorrere il testo usando le freccette e uscire dalla letture premendo il tasto q)il suo utilizzo è elementare:

```
less nomefile
```

Ecco alcuni tips per less (usato di default per leggere i man dei comandi):

- **/stringa** permette di ricercare la **stringa** (insieme di caratteri) dal cursore in poi
- **?stringa** permette di ricercare la **stringa** dal cursore in su (fino ad inizio pagina)
- premendo la barra spaziatrice (space) ci spostiamo di una pagina (corrisponde alle linee mostrate dalla schermata)
- premere PagUp e PagDwon per spostarsi di una pagina (come space) indietro o avanti

Molto simile, ma con meno funzionalità è il comando **more**

- **tail** che visualizza l'ultima parte di un file. Molte utile se vogliamo leggere solo le ultime righe di un log (come **dmesg**).Il suo utilizzo è elementare:

```
tail nomefile
```

- **head** che visualizza le prime 10 righe di un file. Molte utile se vogliamo leggere solo le prime righe di un log (come **dmesg**).Il suo utilizzo è elementare:

```
head nomefile
```

se affiancato dall opzione -N° viene incrementata la visualizzazione delle righe in base al numero scritto ad esempio:

```
head -15 nomefile
```

che visualizzera la prime 15 righe del file

Vedere i processi in esecuzione

Per vedere i processi possiamo usare il semplice **ps** con la sua utile opzione **aux**:

```
ps aux
```

che produrra la lista completa dei processi in esecuzione, con relativi dettaggi su PID (Process IDentifier), utente che che ha avviato il processo, percentuale usata, memoria usata,e altre info. Ecco uno spezzone come esempio:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
ercoppa   6492 13.3 10.8 218844 112632 ?       Sl   21:46  17:11
/opt/firefox/firefox-bin
ercoppa   6604  0.0  0.0      0     0 ?        Z    22:10   0:00 [gnome-open]
<defunct>
ercoppa   6813  0.1  1.6  48264 16648 ?        Sl   22:50   0:06 gnome-terminal
ercoppa   6815  0.0  0.0   2332   708 ?        S    22:50   0:00 gnome-pty-helper
ercoppa   6816  0.0  0.1   4216  1744 pts/0    Ss   22:50   0:00 bash
ercoppa   6853  0.0  0.0      0     0 ?        Z    23:08   0:00 [play] <defunct>
```

Volendo avere una lista dinamica dei processi, possiamo usare il tool **top**, che fornirà praticamente le stesse info di **ps aux** (e non solo), ma aggiornate in tempo reale.

Scompattare un archivio compresso

Se si sta cercando di installare un sorgente potrebbe essere utile vedere "Come installare software in GNU/Linux"

.tar

Usiamo il comando **tar** con l'opzione **x**(extract) e **vf** (per avere un output delle operazioni)

```
tar xvf pacchetto.tar
```

.tar.gz e .gz

Stesso comando che per il .tar, solo con l'aggiunta di **z** (indica a tar che stiamo operando su .tar.gz o .gz)

```
tar xzvf pacchetto.tar.gz
```

.tar.bz2 e .bz2

Stesso comando che per il .tar, solo con l'aggiunta di **j** (indica a tar che stiamo operando su .tar.bz2 o .bz2)

```
tar xjvf pacchetto.tar.bz2
```

.zip

Come da logica usiamo **unzip**

```
unzip file.zip
```

.rar

Anche qui, il comando è abbastanza logico: **unrar**

```
unrar x file.rar
```

.7z

Per scompattare un file .7z, basta usare **7z**:

```
7z x file.7z
```

Avere informazione sul proprio hardware (motherboard, cpu, scheda grafica, ram, chipset ecc)

Principalmente per avere informazione sul proprio hardware possiamo usare tre comandi: **lspci**, **lsusb**, **lshw**.

Con **lspci** possiamo avere informazione su i device PCI connessi, schede AGP e chipset della scheda madre (ad esempio controller SATA). Il suo uso è abbastanza elementare:

```
lspci
```

Che produrra un output del genere:

```
gentoo@ercoppa # lspci
00:00.0 Host bridge: VIA Technologies, Inc. K8T800Pro Host Bridge
00:00.1 Host bridge: VIA Technologies, Inc. K8T800Pro Host Bridge
00:00.2 Host bridge: VIA Technologies, Inc. K8T800Pro Host Bridge
00:00.3 Host bridge: VIA Technologies, Inc. K8T800Pro Host Bridge
00:00.4 Host bridge: VIA Technologies, Inc. K8T800Pro Host Bridge
00:00.7 Host bridge: VIA Technologies, Inc. K8T800Pro Host Bridge
00:01.0 PCI bridge: VIA Technologies, Inc. VT8237 PCI bridge [K8T800/K8T890
South]
00:07.0 FireWire (IEEE 1394): VIA Technologies, Inc. IEEE 1394 Host Controller
(rev 80)
00:08.0 RAID bus controller: Promise Technology, Inc. PDC20378 (FastTrak 378/SATA
378) (rev 02)
00:0a.0 Ethernet controller: Marvell Technology Group Ltd. 88E8001 Gigabit
Ethernet Controller (rev 13)
00:0e.0 Multimedia audio controller: Creative Labs SB Live! EMU10k1 (rev 0a)
00:0e.1 Input device controller: Creative Labs SB Live! Game Port (rev 0a)
00:0f.0 RAID bus controller: VIA Technologies, Inc. VIA VT6420 SATA RAID
Controller (rev 80)
00:0f.1 IDE interface: VIA Technologies, Inc. VT82C586A/B/VT82C686/A/B/VT823x/A/C
PIPC Bus Master IDE (rev 06)
00:10.0 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller
(rev 81)
00:10.1 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller
(rev 81)
00:10.2 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller
(rev 81)
00:10.3 USB Controller: VIA Technologies, Inc. VT82xxxxx UHCI USB 1.1 Controller
(rev 81)
00:10.4 USB Controller: VIA Technologies, Inc. USB 2.0 (rev 86)
00:11.0 ISA bridge: VIA Technologies, Inc. VT8237 ISA bridge [KT600/K8T800/K8T890
South]
00:11.5 Multimedia audio controller: VIA Technologies, Inc. VT8233/A/8235/8237
AC97 Audio Controller (rev 60)
00:11.6 Communication controller: VIA Technologies, Inc. AC'97 Modem Controller
(rev 80)
00:18.0 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron]
HyperTransport Technology Configuration
```

```
00:18.1 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] Address
Map
00:18.2 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron] DRAM
Controller
00:18.3 Host bridge: Advanced Micro Devices [AMD] K8 [Athlon64/Opteron]
Miscellaneous Control
01:00.0 VGA compatible controller: ATI Technologies Inc RV350 AP [Radeon 9600]
01:00.1 Display controller: ATI Technologies Inc RV350 AP [Radeon 9600]
(Secondary)
```

In maniera analoga **lsusb** ci da informazione sui device USB connessi al computer. Il suo uso elementare è:

```
lsusb
```

Che produrrà un output del genere:

```
gentoo@ercoppa # lsusb
Bus 002 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 004 Device 001: ID 0000:0000
Bus 005 Device 001: ID 0000:0000
Bus 001 Device 002: ID 03f0:4911 Hewlett-Packard
Bus 001 Device 001: ID 0000:0000
```

Il comando che però ci fornisce più informazioni sul nostro hardware è **lshw** (quasi mai preinstallato nelle distro), anche lui è molto semplice da utilizzare:

```
lshw
```

Visto l'output molto lungo è consigliabile far generare un report in una pagina html e visualizzarla comodamente con un browser (nell'esempio firefox):

```
lshw -html > info.html && firefox info.html
```

Avanzate

Cambiare i permessi

Introduzione ai permessi sotto GNU/Linux

In GNU/Linux i permessi su un file possono essere di tre tipi:

- **r** -> lettura
- **w** -> scrittura
- **x** -> esecuzione

E sono assegnati principalmente a tre soggetti:

- Proprietario del file (**u**ser)
- Gruppo ossia gli utenti dello stesso gruppo del proprietario (**g**roup)
- Altri (**o**thers)

Il comando per cambiare i permessi è **chmod**, il suo uso è molto elementare:

```
chmod valore file
```

Dove il **valore** dei permessi puo essere espresso in due modi:

- con numero ottale
- con caratteri (u,g,o,a,r,w,x)

Cambiare i permessi esprimendoli in numero ottale

Esprimere i permessi nel sistema ottale è molto semplice e diretto, una volta capita la logica che vi è dietro. Il numero che dobbiamo ricavarci è composto da **tre cifre**, ognuna per esprimere rispettivamente:

- i permessi del proprietario (la prima cifra)
- i permessi del gruppo (la seconda)
- i permessi a tutti gli altri utenti del sistema (la terza)

E ogni cifra ricaverà il valore (da 0 a 7) in base a queste regole:

- nessun permesso ---> valore **0**
- permesso lettura ---> valore **4**
- permesso scrittura ---> valore **2**
- permesso esecuzione ---> valore **1**

Per ricavarci quindi la cifra complessiva di un soggetto (esempio il proprietario) basterà fare la somma dei valori dei permessi che vogliamo assegnargli, ad esempio se desideriamo dare il permesso di letture e scrittura, la cifrà sarà:

4 (lettura) + **2** (scrittura) = **6**

Se invece vogliamo concedere il permesso di lettura e esecuzione:

4 (lettura) + **1** (esecuzione) = **5**

Esempi pratici

Facciamo qualche altro esempio pratico:

- Permesso di lettura, scrittura e esecuzione al proprietario, permesso di sola letture agli altri soggetti:

```
ercoppa@gentoo ~ $ chmod 744 file_uno
ercoppa@gentoo ~ $ ls -l
-rwxr--r--  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Tutti i permessi al proprietario, agli utenti del gruppo sola esecuzione, agli altri nessun permesso

```
ercoppa@gentoo ~ $ chmod 710 file_uno
ercoppa@gentoo ~ $ ls -l
-rwx--x---  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Permesso di lettura e scrittura al proprietario, scrittura al gruppo, esecuzione agli altri

```
ercoppa@gentoo ~ $ chmod 621 file_uno
ercoppa@gentoo ~ $ ls -l
-rw--w---x  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Nessun permesso a tutti

```
ercoppa@gentoo ~ $ chmod 000 file_uno
ercoppa@gentoo ~ $ ls -l
```

```
----------  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

Cambiare i permessi esprimendoli in caratteri

Cambiare i permessi esprimendoli in caratteri, è molto semplice, ma meno diretto in confronto ad un valore ottale. Le regole da comprendere sono molto semplici:

- **u** identifica il proprietario (user)
- **g** identifica gli utenti dello stesso gruppo del proprietario (group)
- **o** identifica tutti gli altri utenti del sistema (others)
- **a** (all) identifica tutti gli utenti del sistema (dal proprietario fino agli utenti estranei)

Con:

- **-** si toglie un permesso
- **+** si concede un permesso

I permessi ovviamente possono essere di tre tipi:

- **r** lettura
- **w** scrittura
- **x** esecuzione

Quindi per assegnare dei permessi ai file basta usare **chmod** con questa sintassi

```
chmod soggetto=simbolotipo_di_permesso file
```

Dove a **soggetto** va sostituito **u** / **g** / **o** / **a** (o più soggetti contemporaneamente esempio: ug), poi far seguire **=**, il **simbolo** **+** / **-** a seconda se si vuole concedere o negare il permesso e infine il **tipo di permesso** **r** / **w** / **x** (o più permessi contemporaneamente, esempio rw). E' possibile anche omettere il simbolo di uguaglianza **=**

Esempi pratici

- Con questo comando daremo i permessi di scrittura a tutti gli utenti del sistema

```
ercoppa@gentoo ~ $ chmod a=+w file_uno
ercoppa@gentoo ~ $ ls -l
--w--w--w-  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Con questo comando daremo tutti i permessi all'utente e nessuno agli altri soggetti

```
ercoppa@gentoo ~ $ chmod u=+rwx,go=-rwx file_uno
ercoppa@gentoo ~ $ ls -l
-rwx------  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Con questo comando daremo il permesso di scrittura solo al proprietario del file

```
ercoppa@gentoo ~ $ chmod u+w file_uno
ercoppa@gentoo ~ $ ls -l
--w-------  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Con questo comando daremo il permesso di lettura a tutti gli utenti del

sistema

```
ercoppa@gentoo ~ $ chmod a+r file_uno
ercoppa@gentoo ~ $ ls -l
-r--r--r--  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- con questo comando daremo il permesso di lettura al gruppo

```
ercoppa@gentoo ~ $ chmod g+r file_uno
ercoppa@gentoo ~ $ ls -l
----r-----  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Con questo comando daremo il permesso di esecuzione agli altri utenti

```
ercoppa@gentoo ~ $ chmod o+x file_uno
ercoppa@gentoo ~ $ ls -l
---------x  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

- Con questo comando daremo il permesso a tutti su tutto

```
ercoppa@gentoo ~ $ chmod ugo=+rwx file_uno
ercoppa@gentoo ~ $ ls -l
-rwxrwxrwx  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
```

Cambiare il proprietario di un file

Per cambiare il proprietario di un file (puo farlo solo root) basta usare **chown**:

```
chown nuovo_proprietario file
```

Ad esempio

```
gentoo@ercoppa # ls -l
-rwxrwxrwx  1 ercoppa ercoppa       0 12 mag 21:04 file_uno
gentoo@ercoppa # chown root file_uno
gentoo@ercoppa # ls -l
-rwxrwxrwx  1 root    ercoppa       0 12 mag 21:04 file_uno
```

Se stiamo operando con una directory è consigliabile usare l'opzione **-R**, per agire ricursivamente anche sul suo contenuto:

```
ercoppa@gentoo ~ $ ls -l
drwxr-xr-x  2 ercoppa ercoppa    4096 13 mag 21:25 prova
ercoppa@gentoo ~ $ ls -l prova/
-rw-r--r-- 1 ercoppa ercoppa 0 13 mag 21:25 file_uno
gentoo@ercoppa # chown -R root prova
gentoo@ercoppa # ls -l
drwxr-xr-x  2 root    ercoppa    4096 13 mag 21:25 prova
gentoo@ercoppa # ls -l prova/
-rw-r--r-- 1 root ercoppa 0 13 mag 21:25 file_uno
```

Cambiare il gruppo di appartenza di un file

Per cambiare il gruppo proprietario di un file usiamo **chgrp**, il suo uso è molto semplice:

```
chgrp gruppo file
```

UN opzione interessante è **-R**, che come logico aspettarsi, cambia il gruppo

proprietario ricursivamente per tutti i file contenuti in una directory:

```
chgrp -R gruppo cartella
```

Ecco qualche esempio:

```
ercoppa@gentoo ~ $ ls -l
drwxr-xr-x  2 ercoppa ercoppa    4096 21 mag 18:08 prova
ercoppa@gentoo ~ $ ls -l prova/
-rw-r--r-- 1 ercoppa ercoppa 0 21 mag 18:08 file_uno
gentoo@ercoppa # chgrp -R root prova
gentoo@ercoppa # ls -l
drwxr-xr-x  2 ercoppa root       4096 21 mag 18:08 prova
gentoo@ercoppa # ls -l prova/
-rw-r--r-- 1 ercoppa root 0 21 mag 18:08 file_uno
```

Uccidere (killare) un processo

Per terminare un processo usiamo **kill** fornendo come indicazione sul processo il **PID** (Process IDentifier), facilmente ricavabile osservando output di **top** o **ps aux**. Il suo uso è semplice

```
kill PID
```

Esempio pratico:

```
kill 5926
```

Un opzione per forzare l'uccisione di un processo, nel caso questo non rispondesse al classico segnale, è **-9**:

```
kill -9 PID
```

Esempio pratico

```
kill -9 5926
```

Caricare un modulo del kernel

Per caricare un modulo del nostro kernel possiamo usare **modprobe**:

```
modprobe nome_modulo
```

Se vogliamo avere una lista dei moduli già caricati usiamo **lsmod**:

```
ercoppa@gentoo ~ $ lsmod
Module                  Size  Used by
nfnetlink_queue         8512  1
nfnetlink               5016  2 nfnetlink_queue
xt_tcpudp               2752  4
xt_multiport            3136  2
iptable_filter          2368  1
ip_tables               9800  1 iptable_filter
xt_state                1856  3
ip_conntrack           37868  1 xt_state
xt_NFQUEUE              1792  3
x_tables               11268  5
xt_tcpudp,xt_multiport,ip_tables,xt_state,xt_NFQUEUE
```

Per avere una lista completa dei moduli compilati nel nostro kernel ,che è possibile caricare con **modprobe**, possiamo usare quest'ultimo con l'opzione **-l** (elle):

```
gentoo@ercoppa # modprobe -l
/lib/modules/2.6.19-gentoo-r5/alsa-driver/synth/snd-util-mem.ko
/lib/modules/2.6.19-gentoo-r5/alsa-driver/synth/emux/snd-emux-synth.ko
/lib/modules/2.6.19-gentoo-r5/alsa-driver/pci/ac97/snd-ac97-codec.ko
/lib/modules/2.6.19-gentoo-r5/alsa-driver/pci/ac97/snd-ac97-bus.ko
/lib/modules/2.6.19-gentoo-r5/alsa-driver/pci/emu10k1/snd-emu10k1.ko
/lib/modules/2.6.19-gentoo-r5/alsa-driver/pci/emu10k1/snd-emu10k1-synth.ko
/lib/modules/2.6.19-gentoo-r5/alsa-driver/misc/ac97_bus.ko
/lib/modules/2.6.19-gentoo-r5/alsa-driver/acore/snd-rawmidi.ko
/lib/modules/2.6.19-gentoo-r5/misc/vboxdrv.ko
/lib/modules/2.6.19-gentoo-r5/x11-drm/radeon.ko
/lib/modules/2.6.19-gentoo-r5/x11-drm/drm.ko
```

Per ricavare il nome del modulo basterà quindi togliere il percorso del file (nel nostro caso /lib/modules/2.6.19-gentoo-r5/cartella/) e non considerare l'estensione (.ko), esempio:

```
gentoo@ercoppa # modprobe drm
```

Visualizzare informazioni su un modulo

per avere qualche informazione in piu' sui moduli compilati nel kernel usiamo **modinfo**:

```
cydonia@cydonia ~$ modinfo drm
filename:       /lib/modules/2.6.20-16-386/kernel/drivers/char/drm/drm.ko
license:        GPL and additional rights
description:    DRM shared core routines
author:         Gareth Hughes, Leif Delgass, José Fonseca, Jon Smirl
srcversion:     1B6FCEBBDA6FEECCBD70ADC
depends:        agpgart
vermagic:       2.6.20-16-386 mod_unload 486
parm:           cards_limit:Maximum number of graphics cards (int)
parm:           debug:Enable debug output (int)
```

Creare un utente

Per creare un nuovo utente nel nostro sistema usiamo **useradd**, vediamo come usarlo accompagnato dalle opzioni piu utili:

```
useradd nome_nuovo_utente -m -G gruppo1,gruppo2,gruppo3 -s /bin/bash
```

Analizziamo subito le opzioni usate:

- **-m** indica al sistema di creare per il nuovo utente una nuova home (/home/utente) con tutti i file base predefiniti (disponibili sotto /etc/skel). Nel caso in cui ci serva un utente solo per far girare un servizio (tipico esempio è un utente creato ad hoc per far girare un demone p2p) possiamo evitare questa opzione
- **-G gruppo1,gruppo2** Con questa opzione indichiamo i gruppi a cui l'utente apparterrà fin da subito (ovviamente poi si potranno aggiungerne altri). I gruppi comuni a quasi tutti i sistema GNU/Linux sono:
    - **users**: gruppo in generale degli utenti

- **audio**: abilita l'accesso ai dispositivi audio
- **cdrom**: abilita l'accesso diretto ai dispositivi ottici
- **floppy**: abilita l'accesso diretto ai floppy
- **games**: abilita il gioco
- **usb**: abilita l'accesso ai dispositivi USB
- **plugdev**: concede la possibilità di monstare ed utilizzre unità rimovibili quali memorie USB o macchine fotografiche.
- **video**: abilita l'accesso all'hardware e all'accelerazione
- **wheel**: abilita l'utilizzo di **su**
- **haldaemon**: come per plugdev, serve per l'automount
- **-s /bin/bash**: indica la shell predefinita dell'utente

Altre opzioni utili sono:

- **-e MM/GG/AA**: la data in cui l'account dell'utente verrà disabilitato
- **-u VALORE_UID**: opzione per assegnare uno specifico UID all'utente (vedere man useradd per avere maggiori info)

Per assegnare una password all'utente (necessaria per il login) vedere la sezione più sotto

Eliminare un utente

Per eliminare un utente dal nostro sistema possiamo usare il comando **userdel**, l'uso più semplice è:

```
userdel utente
```

Nel caso in cui volessimo cancellare anche la sua cartella sotto /home, aggiungiamo l'opzione **-r**

```
userdel -r utente
```

Aggiungere o rimuovere un utente ad un gruppo

Per aggiungere un utente al gruppo usiamo **gpasswd**, con l'opzione **-a** in questo modo

```
gpasswd -a nomeutente gruppo
```

In modo analogo per rimuovere un utente da un gruppo usiamo l'opzione **-d** operando nel seguente modo

```
gpasswd -d utente gruppo
```

Cambiare password ad un utente

Il modo più semplice per cambiare la password di un utente è con **passwd**:

```
passwd utente
```

Oppure se sta cambiando la proprio password si puo omettere l'utente

```
passwd
```

Redirigere l'output di un comando

Prima di tutto bisogna fare una premessa concettuale

Standard input, output ed error

Ad ogni processo GNU/Linux assegna implicitamente tre "descrittori" ("file descriptor":)

- Standard input (**/dev/stdin**) ossia il "descrittore" da dove riceve gli input ed associato al valore 0 (di norma la tastiera)
- Standard output (**/dev/stdout**) ossia il "descrittore" dove produce i propri output ed associato al valore 1 (di norma lo schermo)
- Standard error (**/dev/stderr**) ossia il "descrittore" dove vengono spediti i messaggi di errore ed associato al valore 2 (di norma lo schermo)

Con la Bash possiamo facilmente "gestire" e monovrare questo flusso di dati secondo le nostre esigenze.

Redirigere l'output

Per redirigere l'output di un processo possiamo usare **>**. Un esempio classico dell'uso è quando vogliamo redirigere l'ouput di un comando su un file di testo:

```
ercoppa@gentoo ~ $ ps aux > processo.txt
```

In questo caso ho inviato il risultato del comando **ps aux** al file processo.txt (che potrò leggere con tutta comodità in un secondo momento).

Concatenare più comandi con | , && e ;

Il carattere **|** (premere shit+\ per produrlo) puo essere definito come un condotto (pipeline) che permette di incanalare lo standard output del comando che lo precede, nello standard input del comando che lo segue. Maggiori info qui paragrafo "139.3.2 Condotto"

```
comando1 | comando2
```

All'atto pratico l'output del comando1 verrò "processato/elaborato" dal comando2. Vediamo un esempio molto banale:

```
gentoo ercoppa # lsusb
Bus 004 Device 001: ID 0000:0000
Bus 005 Device 001: ID 0000:0000
Bus 003 Device 001: ID 0000:0000
Bus 002 Device 001: ID 0000:0000
Bus 001 Device 002: ID 03f0:4911 Hewlett-Packard
Bus 001 Device 001: ID 0000:0000
gentoo ercoppa # lsusb | grep "Packard"
Bus 001 Device 002: ID 03f0:4911 Hewlett-Packard
```

Come possiamo vedere l'output di lsusb (che ha il compito di fornire informazioni sui device usb connessi al computer) viene elaborato da grep

La sequenza **&&** permette di far eseguire più comandi alla shell nell'ordine da noi specificato:

```
comando1 && comando2
```

Il comando2 viene eseguito solo, e solo se, il comando1 viene terminato/eseguito positivamente (exit code 0). Facciamo qualche breve esempio:

```
ercoppa@gentoo ~ $ echo ciao && echo pino
ciao
pino
```

In questo caso, il primo comando (echo stampa a schermo semplicente l'argomento fornito) viene eseguito senza problemi e pertanto la bash esegue anche il secondo comando. Vediamo un esempio negativo:

```
ercoppa@gentoo ~ $ cd ... && echo fatto
bash: cd: ...: No such file or directory
```

La bash non puo andare in ... perchè non esiste una tale directory, pertanto il secondo comando non viene eseguito.

Il carattere ";" permette di concatenare quanti comandi si vuole: la shell tenterà di eseguire il comando seguente solo dopo che il precedente sia stato portato a termine ed indipendentemente dall'esito di quest'ultimo; il codice di uscita della sequenza sarà quello dell'ultimo comando eseguito. La sintassi é questa:

```
comando1; comando2
```

esempio:

```
ercoppa@gentoo ~ $ cd ...; echo fatto
bash: cd: ...: No such file or directory
fatto
```

Possiamo vedere che il secondo comando viene avviato nonostante il fallimento del primo.

Manipolare l'output di un comando con grep e tail

Un comando molto utile è **grep**, il suo compito è quello di stampare le linee in cui compaia una certa stringa, per comprenderlo meglio passiamo subito ad un esempio:

```
ercoppa@gentoo ~ $ cat ciao.txt
ciao1
ciao2
ciao3
ciao1 ciao2
ercoppa@gentoo ~ $ cat ciao.txt | grep ciao2
ciao2
ciao1 ciao2
```

Come potete vedere il file ciao.txt contiene varie linee, ma grazie a grep possiamo estrapolare dal nostro file solo le linee che contengono quella specifica stringa. Il comando grep è fondatamentale quando stiamo cercando qualcosa in file molto grossi come un log:

```
ercoppa@gentoo ~ $ dmesg | grep hda
Kernel command line: root=/dev/hda5 1
ide0: BM-DMA at 0xfc00-0xfc07, BIOS settings: hda:DMA, hdb:pio
hda: Maxtor 6B200P0, ATA DISK drive
```

```
hda: max request size: 512KiB
hda: 398297088 sectors (203928 MB) w/8192KiB Cache, CHS=24792/255/63, UDMA(133)
hda: cache flushes supported
hda: hda1 hda2 hda3 hda4 < hda5 hda6 >
EXT3-fs: hda5: orphan cleanup on readonly fs
EXT3-fs: hda5: 1 orphan inode deleted
EXT3 FS on hda5, internal journal
ReiserFS: hda3: found reiserfs format "3.6" with standard journal
ReiserFS: hda3: using ordered data mode
ReiserFS: hda3: journal params: device hda3, size 8192, journal first block 18,
max trans len 1024, max batch 900,
ReiserFS: hda3: checking transaction log (hda3)
ReiserFS: hda3: Using r5 hash to sort names
XFS mounting filesystem hda2
Ending clean XFS mount for filesystem: hda2
Adding 1647292k swap on /dev/hda6.  Priority:-1 extents:1 across:1647292k
```

Come è possibile vedere ho facilmente estrapolato tutti i riferimenti a hda (un disco IDE) dentro a dmesg.

A questo punto é bene soffermarsi un attimo per puntualizzare alcuni problemi in cui prima o poi incorrerà con grep (e non solo) l'utente alle prime armi. Creiamo un file di testo che chiamiamo testo.txt con due righe; "3.4" la prima, "3A4" la seconda:

```
$ echo -e "3.4\n3A4" > testo.txt
$ cat testo.txt
3.4
3A4
```

Se ora diamo grep per selezionare la riga contenente l'epressione "3.4" come abbiamo fatto finora, otterremo questo output:

```
$ grep 3.4 testo.txt
3.4
3A4
```

Vediamo che non é stata selezionata la sola riga prevista. La ragione di questo fatto é che grep fa uso di un sistema di modelli chiamato "espressioni regolari" (spesso abbreviato in "RegExp" od altro, dall'inglese "Regular Expressions"). In questo tipo di modelli alcuni caratteri sono "interpretabili", cioé il comando può sostituire ad esso altri caratteri. Il carattere punto (".") del nostro esempio, in questo contesto, può essere sotituito da un altro qualsivoglia singolo carattere (oltre che da se stesso), ad esempio anche da "A"; quindi la striga "3A4" sarà ritenuta valida da grep. Per ovviare a questo inconveniente, il sistema delle espressioni regolari ammette dei caratteri detti di "fuga" ("escape", in inglese) che servono a dire al comando che uno o più caratteri interpretabili devono essere considerati nel loro significato letterale. Uno di questi é la controbarra ("\", "backslash" in inglese) che impone il significato appunto letterale al singolo carattere successivo. Proviamo, dunque, ad anteporre "\" alla nostra espressione:

```
$ grep 3\.4 testo.txt
3.4
3A4
```

Che dire: benvenuti nel mondo Unix! Naturalmente c'é una ragione al fatto che otteniamo lo stesso precedente output: la backslash é un carattere di escape anche per bash (mentre il punto ha solo il suo valore letterale). Bash, prima di

avviare un comando effettua una quantità di operazioni, tra le quali la risoluzione dei caratteri di escape per lui validi; una volta risolta la controbarra, restituirà a grep l'espressione "3.4" e, quindi, grep interpreterà a sua volta il carattere ".", ormai non più protetto, esattamente come nel caso precedente. Per nostra fortuna bash effettua tale operazione una sola volta, questo significa che potremo proteggere da interpretazioni, questa volta di bash, anche il nostro stesso carattere di escape:

```
$ grep 3\\.4 testo.txt
3.4
```

Bash risolverà la prima controbarra restituendo a grep la striga "3\.4" che ora sarà interpretata come vogliamo. Va detto che la soluzione vista, seppure efficace, non viene generalmente utilizzata, in quanto con espressioni più complesse si rischia la confusione di controbarre. Siccome, ad esempio, bash evita di interpretare il carattere "\" (oltre ad alcuni altri) nelle stringhe delimitate da ", possiamo ricorrere a quest'ultimo:

```
$ grep "3\.4" testo.txt
3.4
```

che dovrebbe apparire più chiaro. Questa é la ragione per cui sarebbe bene abituarsi a delimitare sempre con " le espressioni per grep. Un'altra possibilità é di utilizzare l'opzione "--fixed-strings" di grep che evita l'interpretazione dei caratteri dell'intera stringa:

```
$ grep --fixed-strings 3.4 testo.txt
3.4
```

Esistono parecchie altre fantasie sul tema.

Anche **tail** ci puo aiutare in molte situazioni, il suo scopo è quello di mostrare solo l'ultima parte (di default 10 linee) di un output. Ecco un esempio:

```
ercoppa@gentoo ~ $ dmesg | tail
mtrr: 0xd0000000,0x10000000 overlaps existing 0xd0000000,0x1000000
mtrr: 0xd0000000,0x10000000 overlaps existing 0xd0000000,0x1000000
mtrr: 0xd0000000,0x10000000 overlaps existing 0xd0000000,0x1000000
mtrr: 0xd0000000,0x10000000 overlaps existing 0xd0000000,0x1000000
agpgart: Found an AGP 3.0 compliant device at 0000:00:00.0.
agpgart: Putting AGP V3 device at 0000:00:00.0 into 4x mode
agpgart: Putting AGP V3 device at 0000:01:00.0 into 4x mode
[drm] Setting GART location based on new memory map
[drm] Loading R300 Microcode
[drm] writeback test succeeded in 1 usecs
```

E' possibile specificare la "quantita" di un file da mostrare sia in linee che in bytes, per info su come usare a fondo tail consultare il man con **man tail**

Mandare in background un processo (e recuperarlo)

Per mandare in background un processo usiamo il carattere **&**, in questo modo

```
comando &
```

Ecco un esempio

```
ercoppa@gentoo ~ $ mioscript &
[1] 3924
ercoppa@gentoo ~ $
```

Faccio notare che viene fornito il numero di job (in questo caso 1).A questo punto il comando puo essere "recuperato" in foreground con il comando **fg**

```
fg %n
```

Dove n è il numero di job, quindi nel nostro esempio ci basta fare

```
fg %1
```

Per ritrovarci di nuovo lo script in "primo piano".

Un altro sistema per poter mandare in background un processo: mandiamo in esecuzione un processo

```
ercoppa@gentoo ~ $ ./mio_script
```

Premiamo Ctrl+z in modo tale da fermare (ma non terminare) il processo ed ottenere:

```
[1]+  Stopped                 mio_script
```

E poter mandare in background il processo attraverso in numero di job

```
ercoppa@gentoo ~ $ bg %1
[1]+ mio_script &
```

e semmai recuperarlo con

```
ercoppa@gentoo ~ $ fg %1
```

Creare un archivio compresso

tar

```
tar -cf nome.tar cartella_comprimere/
```

tar.gz

```
tar -czf nome.tar.gz cartella_comprimere/
```

tar.bz2

```
tar -cjf nome.tar.bz2 cartella_comprimere/
```

zip

```
zip -r nome.zip cartella_comprimere/
```

rar

```
rar a nome.rar cartella_comprimere
```

Dubbi? leggi il manuale

man

La prima fonte che dobbiamo consultare in caso di dubbi su un comando è il **man** di quest'ultimo: con buona probabilità è la documentazione più completa e aggiornata che si possa trovare in giro e, anche se in inglese, è scritta nel modo più comprendibile possibile. Per visualizzare il manuale di un comando usiamo **man** seguito dal comando:

```
man comando
```

Vediamone un esempio:

```
man mkdir
```

L'output che riceviamo è (ad oggi) così:

```
MKDIR(Novembre 1998)
MKDIR(Novembre 1998)



NOME
      mkdir - crea directory

SINTASSI
      mkdir [opzioni] directory...

      Opzioni POSIX: [-p] [-m mode] [--]

      Opzioni GNU (forma breve): [-p] [-m modo] [--verbose] [--help] [--version]
[--]

DESCRIZIONE
      mkdir crea directory con i nomi specificati.

      Di natura, i permessi delle directory create sono 0777 (<<a+rwx>>) meno i
bit impostati nella umask.


OPZIONI
      -m modo, --mode=modo
             Imposta  a  modo  i  permessi, che possono essere simbolici come in
chmod(1), e poi usa i permessi
             predefiniti come punto di partenza.

      -p, --parents
             Crea ogni directory genitrice mancante per ogni argomento directory
.  I permessi delle  directory
             genitrici  sono  impostati  al  valore di umask modificato con
u+wx.    Ignora gli argomenti cor-
             rispondenti a directory esistenti (per cui, se una directory /a
esiste  gia,  <<mkdir  /a>>  e  un
             errore, mentre <<mkdir -p /a>> non lo e).

      --verbose
             Visualizza un messaggio per ogni directory creata. Cio e utile
soprattutto con --parents.
```

```
      --     Termina l'elenco delle opzioni.

OPZIONI GNU STANDARD
      --help Stampa un messaggio di spiegazione sullo standard output ed esce
(con successo).

      --version
             Stampa informazioni sulla versione sullo standard output ed esce
(con successo).

      --     Termina l'elenco delle opzioni.

AMBIENTE
      Le variabili  LANG, LC_ALL, LC_CTYPE e LC_MESSAGES hanno il solito
significato.

CONFORME A
      POSIX.2

NOTE
      Questa  pagina  descrive  mkdir come si trova nel pacchetto fileutils-4.0;
altre versioni potrebbero dif-
      ferire leggermente.



                                                  GNU fileutils 4.0
MKDIR(Novembre 1998)
```

Analizziamo le vari parti:

- le prime righe, "NOME" e "DESCRIZIONE" ci aiutano a comprendere la funzione del programma.
- "SINTASSI": illustra come va usato il comando, quali argomenti accetta e come usare le varie opzioni
- "OPZIONI" e "OPZIONI GNU STANDARD": illustrano le principali opzione del comando, con annessa ad ogni opzione una breve ma esauriente spiegazione.
- "AMBIENTE", "CONFORME A", "NOTE" e l'ultima riga: ci vengono fornite alcune informazione di contorno, ma non per questo meno utili

Per i comandi più comuni e/o complessi potrebbero essere presenti nel man numerosi esempi pratici sull'uso delle opzioni.

comando --help

Per avere un sunto, semplice e ben fatto, di tutte le opzioni (e non solo) che il comando contempla possiamo usare l'opzione (che ogni comando dovrebbe avere) **--help** (o a volte **-h**):

```
comando --help
```

Vediamone un esempio:

```
ercoppa@gentoo ~ $ mkdir --help
Uso: mkdir [OPZIONE]... DIRECTORY...
Crea la/le DIRECTORY, se non esistono già.
```

```
Mandatory arguments to long options are mandatory for short options too.
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as needed
  -v, --verbose     print a message for each created directory
      --help     display this help and exit
      --version  stampa le informazioni sulla versione ed esce

Report bugs to <bug-coreutils@gnu.org>.
```

Come possiamo vedere in poche righe è riassunto tutto il man (o almeno la parte più utile) semplice da essere consultate, anche per i meno esperti

help comando

Bash é dotata di un insieme di comandi interni detti "builtin". Alcuni di essi hanno lo stesso nome di comandi i cui eseguibili si possono trovare in /bin o in /usr/bin, hanno la stessa funzione e vengono di norma avviati in luogo di questi. Per quanto riguarda l'uso elementare dei comandi interni, la loro sintassi non si discosta da quella dei corrispondenti esterni, mentre ci possono essere differenze quando si utilizzano opzioni più ricercate. Per visualizzare le caratteristiche di un comando interno si deve consultare la pagina "man" di bash, oppure si può utilizzare il comando "help" "builtin" di bash con la sintassi **help comando**. Esempio:

```
$ help echo
echo: echo [-neE] [arg ...]
   Output the ARGs.  If -n is specified, the trailing newline is
   suppressed.  If the -e option is given, interpretation of the
   following backslash-escaped characters is turned on:
        \a      alert (bell)
        \b      backspace
        \c      suppress trailing newline
        \E      escape character
        \f      form feed
        \n      new line
        \r      carriage return
        \t      horizontal tab
        \v      vertical tab
        \\      backslash
        \num    the character whose ASCII code is NUM (octal).

   You can explicitly turn off the interpretation of the above characters
   with the -E option.
```

Alcune distribuzioni hanno una specifica pagina "man" dedicata ai comandi interni.

Configurazione sistema

i comandi fondamentali (newbies)

Di Anonimi
11-Jul-2006 16:37

Altri articoli di

- Linux in pillole - Tips-Trics
  05-Oct-2007 14:22
- (S)Vista-Linux multiboot (JanHus)
  24-May-2007 15:38
- Semplice guida alla compilazione del kernel
  29-Dec-2006 16:35
- Creare un server FTP con Mepis
  17-Dec-2006 15:37
- creare boot-disk da floppy Secondo metodo
  05-Oct-2006 19:53
- creare boot-disk da flppy avviabile
  05-Oct-2006 14:01
- Installazione fglrx 8.28.8 ATI (9200;9250)
  04-Oct-2006 22:02
- Dual-Boot da windows con WinGrub2006
  01-Oct-2006 16:22
- 3D Xgl Compiz - Ubuntu Dapper - Simply
  17-Sep-2006 02:53
- Installiamo KDE 3.5 RC1 su SimplyMepis
  27-Nov-2005 16:18

Sfoglia tutti gli articoli [11]

un piccolo elenco dei comandi coi qli noi utOnti dovremo - o almeno dovremmo - entrare in confidenza:

Ottenere ulteriori informazioni:
man : formatta e mostra le pagine della guida in linea.
info : sistema di consultazione dei manuali tramite ipertesti.
help : richiama l'help per i comandi builtin della shell.

Lavorare con file e directory:
cd : cambia la directory corrente.
ls : mostra il contenuto di una directory.
cp : copia file e directory.
mv : muove o rinomina un file o una directory.
rm : cancella file e directory.
mkdir : crea una directory.
ln : crea link a file e directory.
pwd : mostra la directory corrente.
chmod : modifica i permessi di accesso di un file.
chown : cambia il proprietario di un file.

cat : mostra il contenuto di un file.
find : cerca un file tra le directory.
vi : l'editor di testo. l'unico ed il solo.

Filesystem e processi
df : mostra lo spazio libero sul disco fisso.
free : mostra lo stato della memoria.
mount : monta un filesystem.
umount : disattiva un filesystem.
ps : visualizza un elenco dei processi correnti.
kill : invia un messaggio (TERM di default) ad un processo.

Sistema:
halt : chiude il sistema.
reboot : riavvia il sistema.
hostname : mostra e cambia il nome dell'host.

Vari:
startx : avvia l'ambiente grafico (X Window System).
date : mostra la data.
gzip : comprime e decomprime file .gz.
tar : crea backup di file (file .tar).
more : separa l'output in piu' pagine (anche less).
reset : resetta il terminale alle impostazioni iniziali.
lynx : browser web solo testo.
zip e unzip : comprime e decomprime file .zip.

##
##############################
Shell builtin: comandi interni alla shell bash
non sono considerati i comandi specifici della programmazione.
per maggiori informazioni sui comandi builtin: help nome_comando
#

alias : definisce alias di comandi.

bg : manda un processo sospeso in background.

cd : cambia la directory corrente.

exec : sistituisce la shell corrente con un nuovo processo.

exit : chiude la shell.

export : esporta una variabile nelle shell figlie.

fg : porta in foreground un processo.

help : richiama l'help per i comandi builtin.

history : mostra l'history della shell.

jobs : mostra i processi fatti partire dalla shell corrente.

logout : esce da una shell di login.

set : setta una variabile.

type : mostra dove si trova l'eseguibile di un comando.

ulimit : controlla le risorse disponibili per la shell.

umask : setta i permessi di default per la creazione di nuovi file.

```
##########################################################################
##############################
# Di seguito vengono presentati i comandi piu' o meno standard di Linux
# divisi per directory.
# Ricordo brevemente che le directory /sbin/ e /usr/sbin/ contengono file
# di solito eseguibili solo dal superutente (root) e di norma non sono
# inserite nel PATH degli utenti normali (l'elenco di directory in cui
# si cercano gli eseguibili)
#

##########################################################################
##############################
# Directory /bin/
#
```

arch : informazioni sull'architettura del sistema.

bash : la shell (interprete di comandi) normalmente usata.

cat : mostra il contenuto di un file.
-n numera le righe.
-b salta le righe vuote.

chgrp : cambia il gruppo di appartenenza di un file.

chmod : modifica i permessi di accesso di un file.
metodo numerico:
primo numero (opzionale):
4 : set user ID
2 : set group ID
1 : swappa la text image
secondo numero; permessi del proprietario:
4 : lettura
2 : scrittura
1 : esecuzione
terzo numero; permessi del gruppo, stessi valori.
quarto numero; permessi degli altri, stessi valori.
-R ricorsivo.

chown : cambia il proprietario di un file o directory.

user.group setta il proprietario a user.group
-R ricorsivo.

cp : copia file e directory.
-r ricorsivo.
-a mantiene gli attributi.
-f forza.

cpio : lavora su archivi di file (come i .tar).

cut : taglia un file di testo.
-b x-y mostra le colonne da x a y del file; conta i byte.
-c x-y mostra le colonne da x a y; conta i caratteri.
-f x mostra i campi x separati da TAB.
-d specifica un altro delimitatore al posto di TAB.

date : mostra la data.

dd : data duplicator, copia da un dispositivo di input su un output.
if=xxx nome del file o device di input.
of=yyy nome del file o device di output.

df : mostra lo spazio libero sul disco fisso.
-h usa un formato piu' "umano".

dmesg : riporta i messaggi mostrati durante il boot.

du : mostra lo spazio usato da file o directory.
-m dati in megabyte.

echo : stampa una stringa.

ed : editor di testo line-oriented.
red edita solo file nella directory corrente.

false : ritorna 1 come codice di uscita.

fuser : identifica i processi che stanno usando un file.

grep : trova testo all'interno di un file.
-i ignora Maiuscolo/minuscolo.
-v inverte il senso della ricerca.
-l elenca solo il nome del file.

gzip : comprime e decomprime file (estensione .gz).
-d decomprime.
-f forza.
-r ricorsivo.
-1 piu' veloce.
-9 miglior compressione.

hostname : mostra e cambia il nome dell'host.

-f mostra il nome completo (host.dominio).

kill : invia un messaggio (TERM di default) ad un processo.
-s specifica che deve inviare il segnale s.
-l lista dei segnali.

ln : crea link a file o directory.
-s crea un link simbolico.

loadkeys : carica un layout della tastiera.

ls : mostra il contenuto di una directory.
-a mostra anche i file nascosti (quelli che iniziano per '.').
-d mostra le directory (senza elencarne il contenuto).
-f disordinato.
-i mostra il numero di inode.
-k dimensione in Kb.
-l formato lungo.
color colora i file secondo il tipo.
-F classifica i file a seconda del tipo.

mkdir : crea una directory.

mknod : crea un device (file speciale) a caratteri o a blocchi.

more : separa l'output in piu' pagine.

mount : monta un filesystem.
-r monta un filesystem in sola lettura.
-w monta un filesystem in lettura/scrittura.
-t xxx monta un filesystem di tipo xxx (fat, vfat, ext2,...).
-a monta tutti i filesystem presenti in /etc/fstab.

mv : muove o rinomina un file o una directory.
-b crea copie di backup.
-i chiede conferma.

netstat : mostra informazioni sulle connessioni di rete.

ping : invia pacchetti ICMP ECHO_REQUEST ad un host.

ps : visualizza un elenco dei processi correnti.
l formato esteso.
u nome utente ed ora di avvio.
m informazioni sull'utilizzo della memoria.
a mostra anche i processi di altri utenti.
r mostra solo i processi attivi.
x mostra anche i processi che non controllano un terminale.

pwd : mostra la directory corrente.

rm : cancella file e directory (nota bene: NON esisteundelete!!!).

-d anche directory.
-i chiede conferma.
-f forza.
-r ricorsivo.

rmdir : rimuove una directory.

sed : legge un file e lo processa con determinati comandi.

setserial : setta la porta seriale.

sh : la shell base di unix.

sleep : si blocca per x Secondi (s) Minuti (m) Ore (h) Giorni (d).

stty : setta il terminale.
Esempio: "stty sane < /dev/ttyX" reimposta al default il teminale X.

su : login come un altro utente (default root).
-p preserva l'ambiente.

sync : svuota la cache del disco.

tar : crea od estrae backup di file.
x estrae.
c archivia.
v verbose.
f file (in cui archiviare o da estrarre).
z processa prima con gzip (per file .tar.gz o .tgz).

touch : cambia la data di un file (se non esiste lo crea).
-a ora di accesso.
-d cambia la data.
-m cambia la data di modifica.

true : ritorna 0 come codice di uscita.

umount : smonta un filesystem.
-a smonta tutti i filesystem inclusi in fstab.
-t smonta solo i filesystem di un certo tipo.

uname : mostra informazioni sul computer.
-m tipo di macchina.
-n nome dell'host.
-r release dell'OS.
-s nome dell'OS.
-v versione dell'OS.
-a tutte le informazioni.

zcat : mostra il contenuto di un file compresso con gzip (file .gz).

---

linux comandi 2

##
##############################
Directory /sbin/
#

SVGATextMode : setta parametri avanzati dello schermo.

badblocks : controlla la superficie di un disco fisso.

chattr : cambia gli attributi di un file.
-R ricorsivo.
a in scrittura appende al file.
i il file non può essere modificato, spostato, eliminato, linkato.
s quando il file viene cancellato lo spazio su disco viene azzerato.
S il file viene sincronizzato immediatamente.

dosfsck : controlla un filesystem DOS.

dumpe2fs : stampa info sul super block e sui blocks del disco fisso.

e2fsck : controlla una partizione ext2fs.
-c controlla anche i badblocks.
-f forza (anche su filesystem "pulito").
-n controlla in modo read-only.

fdisk : manutenzione delle partizioni del disco fisso (anche cfdisk).

fsck : controlla una partizione.

fsck.minix : controlla una partizione minix.

getty : apre una porta tty con richiesta di login (anche agetty, mgetty o mingetty).

halt : ferma il sistema.

hwclock : setta il clock hardware.

ifconfig : configura una interfaccia di rete.

init : lancia i processi di inittab e cambia il runlevel (e' il primo processo eseguito dal sistema).

insmod : installa un modulo nel kernel.
-f forza anche se le versioni sono diverse.

ipfwadm : amministrazione del firewall IP.

kbdrate : cambia l'intervallo di ripetizione della tastiera.

kerneld : demone che rimuove/installa automaticamente i moduli non usati/richiesti.

ldconfig : aggiorna l'elenco delle librerie.

lilo : installa il boot loader che consente di selezionare il sistema operativo all'avvio.

losetup : associa dispositivi loop a file.

lsattr : elenco degli attributi dei file.
-R ricorsivo.
-a tutti i file.

lsmod : mostra informazioni sui moduli del kernel caricati.

mkdosfs : crea una partizione DOS.

mke2fs : crea una partizione ext2fs (il filesystem nativo di Linux).

mkfs : crea una partizione del tipo specificato.

mkfs.minix : crea una partizione minix.

mklost+found : crea una directory lost+found nella directory corrente.

mkswap : crea un dispositivo di swap.

pidof : mostra il PID di un processo.

runlevel : stampa il system runlevel corrente e precedente.

shutdown : chiude il sistema.
-t x aspetta x secondi.
-r dopo la chiusura effettua un riavvio (reboot).
-h blocca il sistema (halt).
-f effettua un reboot veloce.
-c blocca uno shutdown in corso.

swapon : attiva un dispositivo o una partizione di swap.

swapoff : disattiva un dispositivo o una partizione di swap.

tune2fs : setta una partizione ext2fs.
-c x nr. di reboot prima di un filesystem check.
-g setta il gruppo che puo' beneficiare dei blocchi riservati.
-l mostra le impostazioni correnti.
-r setta i blocchi riservati.
-u setta l'utente beneficiario dei blocchi riservati.

update : svuota periodicamente il buffer del filesystem.

-S usa il metodo tradizionale (Chiama sync ogni 30 sec.).
-s x chiama sync ogni x secondi.
-f y svuota il buffer senza chiamere sync ogni y sec (def: 5).

```
##########################################################
##############################
# Directory /usr/bin/
#
```

alien : converte pacchetti da/a vari formati (debian deb, redhat rpm, tgz)

apropos : cerca tra i man un determinato argomento.

ar : crea, modifica ed estrae file da un archivio.

arj : comprime file con arj (file .arj).

as : assembler per Linux.

at : esegue un programma ad una determinata ora.

awk : linguaggio di ricerca ed elaborazione di testo (anche gawk, nawk o mawk).

basename : elimina directory e suffissi dai nomi dei file.

batch : identico ad at, ma viene eseguito solo se il sistema non e' troppo carico.

bc : una calcolatrice solo testo.

biff : avvisa dell'arrivo di posta.

bison : parser generator (anche yacc).

cal : mostra un calendario.

chfn : cambia le proprie finger information nel file /etc/passwd.

chsh : cambia la propria shell di login.

chvt : passa ad un altro terminale virtuale.

clear : pulisce lo schermo del terminale.

cmp : compara due file.

colrm : rimuove le colonne da un file.

column : crea delle colonne.

comm : compara due file ordinati linea per linea.

compress : comprime un file (estensione .Z).

cpp : preprocessore C.

crontab : avvia un processo ad una determinata ora.

csplit : spezza un file in sezioni predeterminate.

ddate : converte la data da gregoriana a discordian.

dialog : per creare finestre e dialog box da shell script.

diff : visualizza le differenze tra due file.
-b ignora gli spazi.
-B ignora le linee vuote.
-i ignora M/m.

diff3 : confronta 3 file.

dircolors : per settare il colore dei file mostrati da ls.

dirname : stampa solo la directory di un riferimento.

dos : lancia l'emulatore DOSemu.

dpkg : gestire i pacchetti Debian.
-i installa un pacchetto.
-r rimuove un pacchetto (purge rimuove anche i file di configurazione).
-s [info] stampa informazioni su un pacchetto [non] installato.
-L [contents] mostra i file contenuti in un pacchetto [non] installato.
-l mostra l'elenco dei pacchetti installati.

dselect : interfaccia per gestire i pacchetti Debian.

dumpkeys : stampa la mappa dei tasti.

emacs : editor di testo (e non solo!) anche in ambiente grafico.
Se ne esce con ctrl+x ctrl+c.

env : esegue un programma in un determinato ambiente.

expand : converte le tabulazioni in spazi.

expr : valuta espressioni (anche aritmetiche).

fdformat : formatta un dischetto.
-n non verifica la formattazione.

fdmount : monta un dischetto.

file : determina il tipo di file.
-z controlla all'interno dei file compressi.

filesize : stampa la dimensione di un file.

find : cerca un file tra le directory.
-name xxx cerca file di nome xxx.
-type X cerca file di tipo X (_d_irectory, _f_ile)

finger : mostra le finger information di un utente di un sistema.

flex : per creare analizzatori lessicali (anche lex).

free : mostra lo stato della memoria.
-m dati in megabyte.

fromdos : converte un testo dal formato DOS a quello Unix (anche dos2unix).

ftp : client ftp (anche ncftp).

funzip : filtro per utilizzare unzip in una pipe.

g++ : compilatore C++.

gcc : compilatore C.

gdb : debugger a riga di comando.

gpm : demone che controlla il mouse.

groff : interfaccia per la compilazione di manuali.

groups : stampa il nome del gruppo di un utente.

gzexe : crea eseguibili compressi che si decomprimono al volo.

head : stampa le prime 10 righe di un file.
-c x primi x byte.
-n y prime y righe.

hexdump : mostra un file in un determinato formato.

id : stampa l'ID e l'UID.

indent : indenta in vari modi un sorgente C.

info : sistema di consultazione dei manuali tramite ipertesti.

install : copia dei file ed assegna permessi e proprietario.

installpkg : installa un pacchetto Slackware.

irc : client irc (anche ircII).

ispell : controllo grammaticale su un file.

kbd_mode : setta la tastiera.

killall : invia un messaggio a tutti i processi con uguale nome.
-s specifica che deve inviare il segnale s.
-i chiede conferma per ogni processo.

jed : editor di testo con interfaccia.

joe : editor di testo con interfaccia.

join : unisce linee di due file in campi comuni.

last : stampa informazioni sull'ultimo login.

ld : linker.

ldd : stampa informazioni sulle librerie condivise.

less : visualizza file di testo (anche more).

locale : mostra e setta le informazioni sul LOCALE (settaggi internazionali).

locate : cerca un file nel database locale.

logger : scrive un messaggio nei log di sistema.

logname : stampa il nome di login di un utente.

look : mostra le linee che cominciano con una certa stringa.

lpr : stampa un file.

lpq : mostra la coda di stampa.

lprm : cancella un lavoro dalla coda di stampa.

lsdev : mostra informazioni sul proprio hardware.

lynx : browser web solo testo.

m4 : macro processor.

make : GNU make.

mail : semplice programma per gestire le email (utile negli script).

man : formatta e mostra le pagine di guida in linea.

mc : shell Norton Commander-like.

md5sum : crea cecksum CRC di un file.

mesg : mostra o meno messaggi da altri utenti inviati da talk o write.

mev : riporta i mouse-events.

mkfifo : crea un file FIFO.

mkisofs : crea un filesystem iso9660 (CD-ROM).

mtools : collezione di programmi per manipolare file DOS.
mattrib : setta gli attributi msdos di un file.
mbadblocks : controlla un filesystem FAT.
mcd : cambia la directory corrente.
mcopy : copia file e directory.
mdel : cancella file.
mdeltree : cancella directory.
mdir : visualizza il contenuto di una directory.
mformat : formatta un dischetto msdos.
mlabel : setta il label di un disco dos.
mmd : cre una directory.
mmount : monta un filesystem msdos.
mmove : sposta file e directory.
mrd : cancella una sottodirectory.
mren : rinomina file e directory.
mtest : visualizza i settaggi degli mtools.
mtype : stampa a schermo il contenuto di un file msdos.

---

linux comandi 3

newgrp : cambia il proprio gruppo di appartentenza.

nice : esegue un programma e ne modifica la priorita' d'esecuzione.
-20 : priorita' maggiore (solo root puo' settare valori negativi).
19 : minima priorita'.

nl : numera le righe di un file.

nohup : esegue un comandi immune da hangup: continua a girare anche dopo un logout.

od : dump di un file in ottale e altri formati.

open : apre un comando in un altro terminale.

passwd : cambia la password di un utente.

paste : unisce linee di files.

patch : applica il risultato di un diff (una patch) ad un file.

pathchk : controlla se un dato file_name e' valido.

perl : interprete per script in linguaggio perl.

pkgtool : gestire pacchetti Slackware.

printenv : stampa informazioni sull'ambiente.

procinfo : mostra informazioni tratte dal filesystem /proc.

pstree : mostra un albero di processi.

quota : mostra le percentuali di dischi utilizzabili dagli utenti.

removepkg : elimina un pacchetto Slackware.

renice : modifica la priorita' di un processo attivo.

reset : resetta il terminale alle impostazioni iniziali.

rev : inverte le linee di un file.

rpm : gestire pacchetti RedHat.
-i installa un pacchetto.
-e rimuove un pacchetto.
-qi [-qip] mostra informazioni su un pacchetto [non] installato.
-ql [-qlp] mostra i file contenuti in un pacchetto [non] installato.
-qa mostra l'elenco dei pacchetti installati.

script : stampa su un file tutto l'input da tastiera.

sdiff : confronta due file ed elimina le differenze.

setfont : seleziona il font da usare.

setleds : setta i led della tastiera.

setterm : setta gli attributi del terminale.

shar : crea shell archives.

showfont : mostra il font usato.

showkey : mostra i codici della tastiera.

size : mostra dimensione e info di un file obj (e chi ha generato un file core).

skill : invia un signal o cambia la priorita' di un programma.

sort : ordina linee di testo.
-b ignora gli spazi all'inizio di riga.

-f ignora Maiuscolo/minuscolo.
-i ignora i caratteri estesi.
-n segue l'ordine numerico.
-r inverte l'ordine.

split : spezza un file in blocchi di dimensioni date.
-l xxx ogni blocco contiente xxx linee.
-b xxx [b,k,m] ogni blocco e' di xxx byte, kilo, mega.

strings : trova linee stampabili in un file.

strip : elimina simboli da un obj.

sudo : esegue un programma come superutente (root).

sum : Crea un CRC di un file.

superformat : formatta floppy ad alte capacita'.

tail : mostra la fine di un file.
-s solo le ultime s righe.
-c x ultimi x byte.
-f continua a leggere un file, utile se questo viene modificato.

tee : legge da standard input e stampa su stdout e su file.
-a append sul file.
-i ignora segnali di interruzione.

telnet : apre una sessione telnet.

test : esegue dei test su file o condizioni.

time : esegue un programma e mostra informazioni sul tempo di esecuzione e carico.

tload : rappresenta graficamente il carico medio del sistema.

todos : converte un testo da Unix a MSDOS (anche unix2dos).

top : mostra i processi che usano piu' tempo CPU o memoria.

tr : converte o cancella caratteri.

troff : formatta i documenti come le manpage.

tty : stampa il nome del file terminale connesso allo standard input.

tunelp : setta vari parametri per la stampante.

unarj : decomprime file .arj.

uncompress : decomprime file compressi (file .Z).

unexpand : converte spazi in tabulazioni.

uniq : rimuove linee indentiche in file ordinati.

unzip : decomprime file .zip.

updatedb : update di un file database di nomi di file (usato da locate).

uptime : mostra da quanto il sistema e' attivo.

users : mostra il nome degli utenti correntemente collegati.

uudecode : decodifica un file ASCII in un binario.

uuencode : codifica un file binario in ASCII.

vi : l'editor di testo. l'unico ed il solo.
Se ne esce (senza salvare) con "ESC :q!" salvando con "ESC :wq".

vim : la migliore versione di vi .

vmstat : riporta statistiche sulla memoria virtuale.

w : mostra chi e' loggato e cosa sta facendo.

wall : invia un messaggio a tutti i terminali.

wc : conta il nr di byte, parole e linee di un file.
-c conta i byte.
-w conta le parole.
-l conta le newline.

whatis : ricerca nel database della guida per una parola.

whereis : indica dove si trova un determinato file.

which : indica il percorso completo di un file.

who : mostra chi e' loggato.

whoami : indica l'usemame con cui si e' attualmente loggati.

whois : interroga un database whois.

wish : interprete per script in linguaggio Tk.

workbone : interfaccia solo testo per ascoltare CD audio.

write : invia un messaggio ad un altro utente.

xargs : appende ad un comando opzioni prese dallo stdin.

yes : emette continuamente una stringa finche' non viene killato (default y).

zcmp : usa cmp su file compressi.

zdiff : usa diff su file compressi.

zgrep : esegue grep su file compressi.

zgv : visualizza in un terminale molti formati di immagini.

zip : comprime file con zip.

znew : ricomprime file da .Z a .gz.

```
##############################################################
##############################
# Directory /usr/sbin/
#
```

addgroup : aggiunge un gruppo (acnhe groupadd).

adduser : aggiunge un nuovo utente al sistema (anche useradd).

cfdisk : manipola le partizioni del disco fisso.

chroot : cambia la directory root e vi esegue un programma.

ctrlaltdel : setta la funzione della combinazione CRTL+ALT+DEL.

debugfs : debugga un filesystem ext2.

groupdel : elimina un gruppo di utenti.

hdparm : mostra e setta molti parametri del disco fisso.

inetd : gestore di molti servizi di rete.

liloconfig : script che aiuta a configurare lilo.

makewhatis : genera il database per il comando whatis.

psupdate : aggiorna il file /etc/psdatabase che rappresenta l'immagine della mappa di sistema attuale (dopo aver ricompilato il kernel).

rdev : setta i dispositivi di un kernel (HD, swap, video mode...).

traceroute : mostra la strada percorsa per raggiungere un dato host.

userdel : elimina un utente del sistema.

vipw : edita il file /etc/passwd.

visudo : edita il file sudoers.

```
#########################################################
#############################
# /usr/X11R6/bin/
#
```

SuperProbe : identifica la scheda video.

X : il sistema a finestra X Window (di norma un link ad un X Server o un wrapper).

bitmap : editor di bitmap.

clock : un orologio.

editres : edita le risorse delle applicazioni Xt.

ghostview : visualizza file postscript (anche gv).

startx : avvia il sistema a finestre.

xbiff : avvisa graficamente se e' arrivata posta.

xcalc : piccola calcolatrice.

xclipboard : visualizza gli appunti.

xconsole : visualizza i messaggi mandati a console.

xdm : gestisce il login grafico.

xedit : editor di testo base.

xev : mostra eventi del mouse e della tastiera.

xeyes : occhi che seguono il puntatore.

xf86config : configura l'ambiente grafico.

xfontsel : seleziona un font.

xhost : controlla l'accesso all'X Server da altri host.

xinit : inizializza il sistema grafico (chiamato da startx).

xkill : uccide una applicazione in finestra.

xman : consultare le pagine di manuale graficamente.

xpaint : semplice programma di paint.

xrdb : setta le risorse X.

xset : setta vari parametri (mouse, screensever,...).

xsetroot : setta il colore dello sfondo e il puntatore.

xterm : apre una finestra terminale.

xv : visualizza molti formati di immagini.

xvidtune : aggiusta i parametri dello schermo.