I Comandi di base

Samuele Cacchiarelli

19 ottobre 2004

Sommario

Appunti della seconda e terza lezione del CORSO LINUX BASE organizzato dal GLM.

Parte I
Console e Terminali

In ogni sistema operativo, solitamente l'interfaccia principale che un utente utilizza per accedervi è costituita dalla *console*.

La CONSOLE è la coppia tastiera/video collegata direttamente alla macchina.

In un sistema mono-utente la *console* spesso identifica il computer che abbiamo avanti a noi. In un ambiente multi-utente come Linux , ciò non è sempre vero.

Visto infatti che ad una stessa macchina Linux possono accedere contemporaneamente più utenti, deve essere possibile il collegamento di più tastiere e video allo stesso computer. Tale collegamento avviene attraverso i *terminali*.

> "Un TERMINALE [REALE] è costituito da una tastiera, un video, ed una piccolissima unità di elaborazione locale, che si occupa esclusivamente di gestire la comunicazione tra il terminale stesso e l'elaboratore a cui è collegato. In sostanza il terminale si limita a visualizzare sullo schermo i messaggi del sistema e ad inviare i comandi digitati dall'utente sulla tastiera.
>
> Spesso come terminali vengono usati dei normali personal computer, dotati di un opportuno software di emulazione di terminale . In questo caso è del tutto ininfluente la potenza del personal utilizzato come terminale, visto che verrà usato esclusivamente per introdurre l'input e visualizzare l'output, mentre ogni elaborazione verrà eseguita dal computer a cui il terminale è collegato.

> Esistono dei terminali più evoluti, i cosiddetti terminali grafici (in ambiente UNIX si parla spesso di *X Terminal*) che permettono di utilizzare un'interfaccia grafica (GUI) per eseguire le operazioni di input/output e quindi consentono anche di visualizzare un output di tipo grafico (immagini, disegni, grafici).[1]"

La presenza di più utenti in una stessa macchina può avvenire anche attraverso i cosiddetti *terminali virtuali*.

In Linux si parla di console o terminale virtuale, intendendo degli speciali device identificati dal sistema come *tty*, che danno la possibilità di effettuare il login di sistema dalla stessa console reale, ovvero direttamente dalla stessa macchina. Al boot del sistema, solitamente vengono infatti inizializzate più di una console virtuale. Tali console sono raggiungibili dall'utente attraverso la combinazione di tasti ALT+F1, ALT+F2, ALT+F3. ecc ecc. Su ognuna di queste è possibile effettuare un login ed avere pertanto un accesso al sistema.

Per fare un esempio, supponendo che si stia lavorando sul PC come utente pippo e si abbia bisogno di effettuare alcune operazioni come utente root; si può tranquillamente passare ad una seconda console virtuale, semplicemente premendo la combinazione di tasti ALT+F2 o ALT+F3 a seconda che si voglia passare alla tty2 o alla tty3 e riloggarsi come root; a seguito di tale operazione, ci saranno due utenti collegati contemporaneamente sulla stessa macchina, uno sulla console virtuale iniziale (solitamente tty1 o tty7 se si è acceduti al sistema direttamente in modalità grafica) un altro sulla tty2 o tty3. Si potrà cioè lavorare su due console virtuali attraverso una sola console reale, *switchando* dall'una all'altra semplicemente attraverso la combinazione di tasti sopra indicata.

Prompt dei comandi

Linux, a differenza di un sistema Windows non necessita di una interfaccia grafica per interagire con l'utente. Una gran parte dei comandi Linux, infatti, vengono (o possono essere) dati direttamente da riga di comando o meglio dal *prompt dei comandi*.

Il prompt dei comandi in Linux è spesso identificato da un programma particolare, ovvero dall'interprete *bash*. Sebbene in Linux esistano diversi interpreti di comandi (detti anche *shell*), la bash è senz'altro la più diffusa.

Ognuna di queste shell ha delle proprie caratteristiche e peculiarità ma, volendo semplificare, le prime differenze che balsano subito all'occhio di chi è stato abituato solo ad una shell tipo DOS sono le seguenti:

- Le più comuni shell linux dispongono di un notevole set di comandi propri (*built-in*);

[1] da UNIX: introduzione elementare - Guida introduttiva al sistema operativo UNIX per principianti di M.Liverani, 1995 - http://www.isinet.it/~marco/unix/index.html

- supportano la colorazione del prompt e dell'output;
- permettono di default la memorizzazione dei comandi;
- permettono il completamento automatico dei comandi e dei file;
- non eseguono i comandi semplicemente indicando il loro nome se questi non si trovano nel cosiddetto *path dei comandi.*

La shell BASH

Come anticipato sopra, la shell è semplicemente un mezzo attraverso il quale dare dei comandi al computer . La shell più comune nei sistemi Linux è senza dubbio la shell "Bash". Vediamone alcune caratterstiche.

Il file .bashrc

È convenzione in un sistema linux chiamare i file di configurazione di un determinato programma con il nome stesso del programma più il suffisso "rc" e aggiungere cone prefisso il "." qualora si tratti di file di configurazione degli utenti presenti nelle rispettive *home directory.* La shell Bash, segue questa convenzione, pertanto è possibile personalizzare la configurazione a livello utente andando a modificare o creare il file ".bashrc" nella propria *home directory.*

Gli Alias

Una delle principali personalizzazioni che è possibile ed utile creare nel proprio file di configurazione, sono gli alias dei comandi. Questi come dice la parola stessa, rappresentano una modalità attraverso la quale è possibile dare un nome diverso (o anche lo stesso) ad un comando.

Ad esempio il comando *ls* semplicemente indica i file e directory presenti nella directory. Le opzioni *–color* e *-F* conferiscono all'output del comando ls una maggiore leggibilità. Può essere opportuno per tanto rinominare il comando ls con un suo alias in modo da poter ottenere sempre un output più comprensibile.

Il modo per farlo e dare i seguenti comandi o aggiungerli al file di configurazione:

```
alias l='ls --color -F'
```

ma volendo possiamo anche dare lo stesso nome del comando all'alias

```
alias ls='ls --color -F'
```

Il comando alias senza opzioni restituisce la lista di tutti gli alias già impostati.

Per annullare un alias si utilizza il comando unalias.

Es:

```
unalias ls
```

Parte II
Manuali online

Prima di elencare una serie di comandi, di cui si ha normalmente bisogno in un ambiente linux vogliamo indicarne due che possono essere di aiuto per tutti gli altri. Ci riferiamo ai comandi per accedere ai manuali on line.

1 man

man è il comando per visualizzare le informazioni sintetiche ma complete sull'utilizzo di un determinato programma, comando o più in generale su di un determinato argomento.

Ad esempio è possibile avere informazioni sul comando stesso *man* digitando:

```
man man
```

oppure informazioni sull'argomento console con il comando:

```
man console
```

L'output dello stesso generalmente segue una struttura logica standard così composta:

> "viene sempre riportato il nome del comando illustrato seguito da una brevissima descrizione del comando stesso; viene poi descritto in modo sintetico la sintassi del comando (synopsis) ed infine vengono elencate e descritte dettagliatamente le opzioni che è possibile specificare insieme al comando. Di solito alla fine della pagina è riportato l'elenco dei files di configurazione del programma stesso (files), l'elenco di eventuali comandi correlati (see also), eventuali problemi noti, riscontrati in situazioni particolari, nell'uso del programma (bugs) ed il nome dell'autore del programma (authors)[2]".

[2]da UNIX: introduzione elementare - Guida introduttiva al sistema operativo UNIX per principianti di M.Liverani, 1995 - http://www.isinet.it/~marco/unix/index.html

Le pagine di manuale richiamate automaticamente dal comando man, sono collocate solitamente in apposite directory man e suddivise in base alle seguenti categorie:

1. Comandi utenti
2. Chiamate di sistema
3. Subroutines
4. Devices, dove sono riportate le descrizioni dettagliate dei devices installati sul sistema
5. Formati dei file
6. Giochi
7. Miscellaneous, altre descrizioni che non trovano collocazione migliore nelle altre sezioni
8. Comandi di amministrazione di sistema

Alcune pagine di manuale possono avere lo stesso nome e riferirsi però a due categorie diverse. Ad esempio esiste la pagina di manuale per il comando *mount* ed è collocata nella sezione 8 (Comndi di amministrazione), ma esiste anche la pagina di manuale per la chiamata di sistema *mount* ed è collocata nella sezione 2. Per accedere all'una o all'altra pagina occorre in questo caso specificare anche il numero di pagina. Ad esempio, un semplice utente è solitamente interessato a conoscere la sintassi e il funzionamento del comando mount.

Potrà digitare

```
man 8 mount
```

per avere il manuale del comando mount.

Il programmatore invece può avere bisogno di informazioni più specifiche inerenti al comando mount.

Potrà digitare

```
man 2 mount
```

Per avere la pagina della chiamata di sistema mount.

Oppure

```
man -a mount
```

per vedere in sequenza tutte le pagine relative a mount presenti in tutte le eventuali sezioni.

Bisogna aggiungere che non tutti i comandi dispongono di una pagina man che ne spieghi l'utilizzo. Un comodo programma per vedere quali comandi o argomenti dispongono di un manuale man è *xman*. Come si può intuire dalla presenza della x all'inizio del comando, xman, sebbene sia alquanto spartano, va eseguito in ambiente grafico. Un ulteriore comando per trovare quali manuali vi sono per un determinato comando o argomento è *apropos*.

Es:

```
apropos xterm
```

il cui risultato può assomigliare al seguente

```
resize (1x) - set TERMCAP and terminal.....
xterm (1x) - terminal emulator for X
terms (5) - database of blessed terminals for...
xtermset (1) - change settings of an xterm
```

2 info

Un altro modo, meno utilizzato e apparentemente più complicato, con cui vengono rilasciati alcuni manuali, in un sistema linux, è costituito dal formato info. Tale formato permette la navigazione in maniera pseudo-ipertestuale all'interno delle pagine di manuale. Risulta pertanto maggiormante indicato per la spiegazione di comandi o argomenti più complessi, articolabili in diverse sezioni.

Parte III
Gestione filesystem

Di seguito riportiamo un rapido elenco dei comandi mostrati durante la seconda e terza lezione del CORSO LINUX del GLM. Per una spiegazione più accurata degli stessi si rimanda alla lettura dei rispettivi manuali attraverso il comando *man* e delle fonti indicate nelle note.

In particolare si consiglia la lettura del CAPITOLO 7 "ESERCIZI PRATICI" dell'opera Appunti di Informatica Libera reperibile al seuente indirizzo internet:

http://a2.swlibero.org/a21.html

3 Partizionamento di un disco fisso

```
cfdisk /dev/hda
```

dove, in questo caso, */dev/hda* è il nome del file di dispositivo rappresentante il primo disco fisso (Master) attacato al primo canale ide della scheda madre del computer.

Questo comando consente di creare sul disco le partizioni primarie (*/dev/hda1, /dev/hda2, /dev/hda3, /dev/hda4*) o logiche (*/dev/hda5, /dev/hda6..* e così via) sulle quali poi verranno creati i rispettivi filesystem.

Un comando più spartano ma più potente, per effettuare la stessa operazione è fdisk.

```
fdisk /dev/hda
```

4 creazione di un filesystem

Vi sono diversi comandi per creare un filesystem in Linux ma i più comuni sono:

mkfs.etx2 per creare un filesystem ext2

mkfs.ext3 per creare un filesystem ext3

mkfs.*altri_ tipi* per creare un filesystem del tipo specificato dall'estenzione

In alternativa a tali comandi specifici è possibile usare il semplice *mkfs* seguito dall'opzione *-t tipo_filesystem*.

Nell'esempio seguente è possibile creare un filesystem ext3 sulla seconda partizione primaria del primo disco fisso.

```
mkfs.ext3 /dev/hda2
```

oppure

```
mkfs -t ext3 /dev/hda2
tune2fs (per i filesystem etx2/ext3)
```

5 mounting di un filesystem

Una volta creato il filesystem è possibile montare la partizione su una directory vuota di un filesystem già montato (solitamente viene montata in appositi punti di mount del filesystem principale) in modo da poter rendere accessibili i suoi dati al sistema.

Il comando è

```
mount /file/di/dispositivo /punto/di/mount
es:
mount /dev/hda2 /mnt/disco2
```

Da notare che qualora si voglia montare un dispositivo quale il cdrom, bisognerà indicare come dispositivo il disco intero e non una sua partizione

Ad esempio se il cdrom è attaccato come disco secondario (*slave*) del primo canale ide:

```
mount -o ro /dev/hdb /mnt/cdrom
```

L'opzione *-o ro* imposta il mounting del dispositivo in sola lettura.

Per evitare ogni volta di specificare sia il dispositivo che le eventuali opzioni ed ovviamente il punto di mount è possibile fissare tali parametri direttamente nel file di configurazione */etc/fstab* (si veda "man mount" per maggiori informazioni)

Una possibile riga da aggiungere al file /etc/fstab relativa al cdrom può essere questa:

```
/dev/hdb  /mnt/cdrom  iso9660  ro,user,noauto 0 0
```

Se presente in /etc/fstab sarà dunque sufficiente eseguire il comando

```
mount /mnt/cdrom
```

per montare ad esempio il cdrom.

Per smontare un dispositivo in ogni caso è sufficiente indicare semplicemente il comando umount seguito dal punto di mount da "sganciare" dal sistema

```
umount /mnt/disco2
```

nota

- non è possibile montare una partizione se non è stata inizializzata con un filesystem (in gergo *formattata*)
- non è possibile montare un filesystem non supportato dal kernel (in realtà non esiste un filesystem che linux non supporti ;-)

6 Inizializzazione e attivazione di una partizione di swap

La partizione di swap in Linux serve a creare una zona nel disco fisso dedicata alla cosidetta memoria virtuale, ovvero quella zona di memoria supplementare utilizzabile dal sistema qualora la memoria fisica (rappresentata dalla RAM) sia insufficiente.

Il comando per conoscere l'ammontare di memoria disponibile ed utilizzata è *free*

```
free
      total  used   free  shared buffers cached
Mem:  377172 341264 35908 0      26984   160904
+/- b/c:     153376 223796
Swap: 313228 0      313228
```

A differenza dei normali filesystem, lo speciale filesystem di swap non va montato.

Più precisamente, la procedura per creare e attivare una partizione di swap è la seguente:

- creare preliminarmente una partizione con cfdisk o fdisk e segnarla come swap
- inizializzare la partizione con il comando mkswap

  ```
  mkswap /dev/hda3
  ```

- attivare la partizione con il comando

  ```
  swapon /dev/hda3
  ```

Data l'importanza di avere una partizione di swap, in un sistema Linux, è necessario accertarsi che la stessa esista e che venga attivata automaticamente in fase di boot. Pertanto è importante che in /etc/fstab siano specificati i parametri e le opzioni corrette al riguardo.

La seguente riga nel file /etc/fstab, ad esempio, fa si che la partizione /dev/hda3 venga attivata come swap in fase di boot.

```
/dev/hda3   none   swap  sw  0 0
```

Parte IV
Operazioni su File e Directory[3]

7 creare una directory

Il comando per creare una directory è mkdir. Ad es:

```
mkdir nomedirectory
```

[3]da "LINUX GUIDA PRATICA" di M. Stutz, ed. Mondadori Informatica 2003

Per creare un intero ramo di directory in una volta sola si può usare mkdirhier:

```
mkdirhier directory/sottodirectory/altradirectory
```

oppure mkdir con l'opzione -p:

```
mkdir -p directory/sottodirectory/altradirectory
```

8 visualizzare le directory

```
ls [nomedirectory]
```
[4]

tale comando visualizza il contenuto della directory corrente o delle directory indicate come argomento.

Alcune ozioni utili sono *-a* e *-l*:

```
ls -a [nomedirectory]
```

visualizza anche i file o directory nascosti.

```
ls -l [nomedirectory]
```

visualizza maggiori infirmazioni su file e directory.

Per visualizzare l'albero delle directory intero:

```
tree  [nomedirectory]
```

9 creare un file

La creazione di un file può avvenire in diversi modi. Molti programmi o comandi generano o danno la possibilità di generare file. Uno di questi è *touch*.

```
touch nomefile
```

Un altro modo di generare un file è attraverso la redirezione dell'output di un comando in un file. Ad esempio:

```
echo 'Mettiamo questa frase in un file' > nuovofile
```

[4]Per convenzione indichiamo tra parentesi quadre gli argomenti o le opzioni che sono opzionali. La stessa convenzione è utilizzata anche nelle pagine man.

il carattere speciale "> " redirige lo *standard-output* del comando precedente in nuovofile.

Il comando echo avrebbe semplicemente stampato a schermo (*standard-output*) la frase indicata. La presenza del carattere di redirezione

Altri comuni caratteri di redirezione sono "> >", "2>", "2>&1" :

> > concatena lo standard-output di un comando, ad esempio, in un file senza cancellare il contenuto eventualmente già presente nel file;

2> redirige lo *standard-error* di un comando;

2>&1 redirige sia lo *standard-output* che lo *standard-error* di un comando.

10 cambiare directory

```
cd directory
cd /usr/local/directory
```

Il comando cd senza argomenti porta l'utente nella propria *home directory*

I seguenti comandi per tanto sono equivalenti.

```
cd
cd /home/nomeutente
cd $HOME
```

11 copiare file e directory

```
cp file1 file2
```

copia file1 in file2.

```
cp -p file1 file2
```

come sopra ma preservando i permessi.

```
cp -R dir1 dir2
```

copia ricorsivamente il contenuto della directory dir1 in dir2 .

```
cp -a dir1 dir2
```

come sopra, preservando i permessi dei file, i link simbolici.

12 Dare più di un nome ad uno stesso file (hard link)

```
ln nome nuovo_nome
```

questo comando crea un cosiddetto *hard-link* ovvero assegna un nuovo nome allo stesso file. Con questo comando non viene creata quindi una copia del file ma solo un nuovo nome. I file *nome* e *nuovo_ nome* sono in realtà lo stesso file. A riprova di ciò è sufficiente osservare come gli attributi dei due file (visualizzabili con ls -l) siano e rimangano identici tra loro, anche dopo modifiche effettuate all'uno o all'altro file.

13 creare un link simbolico

```
ln -s nomefile filelink
```

crea un link simbolico (*filelink*) che punta al file *nomefile*.

Nota: A differenza del comando ln, con *ln -s* non è necessario che il file linkato (*nomefile*) esista realmente e non è necessario che il file linkato risieda nella stessa partizione (o filesystem) di *filelink*.

14 spostare un file o directory

```
mv dir1 /nuovo/percorso
```

se la directory */nuovo/percorso* esiste, il comando indicato sposta la directory *dir1* dentro alla directory */nuovo/percorso* creando quindi */nuovo/percorso/dir1*;

se la directory */nuovo/percorso* non esiste il comando sposterà la directory *dir1* nella cartella */nuovo* rinominandola con il nome *percorso*. Creerà quindi */nuovo/percorso*.

```
mv uno due tre quattro
```

In caso di più argomenti l'ultimo deve essere costituito da una directory. Nell'esempio precedente, il comando sposta i file o directory *uno*, *due* e *tre* nella directory *quattro*.

15 rinominare un file o directory

Come in parte visto sopra, mv si comporta diversamente a seconda della tipologia e del numero dei file che gli vengono passati come argomenti.

Ad esempio nel caso di un file (fileuno)

```
mv fileuno due
```

se *due* non esiste, il comando rinomiva il file *fileuno* in *due*; se *due* esiste ed è un file, il comando lo sovrascrive; se *due* esiste ed è una directory, il file uno viene spostato nella directory due.

Nel caso di una directory invece (dir1)

```
mv dir1 due
```

se due non esiste, il comando rinomina la directory dir1 in due; se due esiste ed è un file il comando restituisce un errore; se due esiste ed è una directory, dir1 viene spostata in tale directory.

16 cancellare un file o directory

```
rm file [file2 .. .. ]
```

cancella il file o i file indicati.

```
rm -r directory [directory2 .. ..]
```

cancella (ricorsivamente) la directory o le directory indicate e il loro contenuto.

```
rmdir directory [directory2 .. .. ]
```

cancella una o più directory vuote.

Il comano *rm*, di per sé, non chiede conferma per la cancellazione di un file o directory; una volta dato l'invio al comando rm semplicemente cancella quanto indicato. Affinché il comando chieda conferma della cancellazione dei file o directory indicati è necessario aggiungere l'opzione *-i* .

```
rm -i file
```

cancella il file chiedendo conferma.

Molte distribuzioni Linux proprio per evitare che l'utente inesperto possa cancellare erroneamente file importanti, impostano automaticamente un alias a tale comano.

```
alias rm=''rm -i''
```

In tale modo il comportamento di default del comando rm, sarà quello di chiedere conferma ogni volta. In questa circostanza per forzare la cancellazione senza richiesta di conferma di aggiunge l'opzione *-f*.

```
rm -f  file
```

cancella il file senza conferma.

```
rm -i -f  file
```

cancella il file senza conferma

```
rm -i -f -i  file
```

cancella il file chiedendo conferma

Dagli esempi riportati si capisce che l'ultima tra le opzioni -i o -f prevale sulle altre.