



## Mini guida

Il primo passo che un utente deve compiere dopo il boot del sistema è il login, cosa che chi utilizza windows NT conoscerà ma alla quale gli utenti windows 9x sono estranei. Questo perchè Linux, a differenza dei vari windows 9x è un sistema multiutente: anche windows 9x, da un certo punto di vista, è multiutente, visto che è possibile creare due o più "utenti" nello stesso sistema ognuno dei quali con le proprie impostazioni e propri file (sebbene non esista il concetto di ownership dei file): il fatto è che la multiutenza di windows 9x è detta "multiutenza seriale", mentre linux è un sistema a multiutenza parallela. Cosa significano queste due (apparentemente assurde) definizioni? Nella multiutenza seriale, possono esistere più utenti che possono però lavorare solamente uno alla volta sulla macchina: se ad esempio l'utente "pippo" sta utilizzando la macchina, l'utente "pluto" dovrà aspettare che il primo si disconnetta per poter lavorare (il famigerato "Disconnetti Utente" del menu di avvio). Nella multiutenza parallela, invece, due o più utenti possono lavorare in simultanea sulla stessa macchina senza che il lavoro dell'uno sia bloccato o intralciato dal lavoro dell'altro. Addirittura, i due utenti si potranno anche parlare l'uno con l'altro, ma vedremo più avanti questi giochetti!

Tutta questa prefazione per introdurre un programma che vedremo ogni qualvolta avvieremo il sistema: `login`

Login altro non è che un programma che ci presenta una richiesta di username e password per poterci dare l'autorizzazione a lavorare sulla macchina: una volta che avremo digitato una coppia username-password corretta, potremo iniziare il nostro colloquio con il sistema o, in caso opposto, non potremo fare alcunchè (ecco perchè non si dovrebbero mai dimenticare le password!).

Il login avviene solitamente in due modi: testuale (in console) o grafico (sotto X). Il primo è il più classico ed in stile *unix*, mentre il secondo è più "evoluto" e ci permette di lavorare direttamente nell'ambiente grafico, grazie a programmi come *xdm*, *gdm*, *wdm*, *qdm* e simili.

Tornando al login, siamo rimasti alla richiesta di uno username: digitate il nome dell'utente che l'installazione vi ha chiesto di creare (sperando che l'abbiate fatto, visto che utilizzare l'account di root può essere molto pericoloso, soprattutto quando non si conosce bene il sistema, e ancora di più quando lo si conosca, dicono alcuni!), dopodichè inserite la password: se non avete fatto errori, vi vedrete catapultati nella vostra home directory [1] con il prompt pronto ad eseguire ogni vostro ordine, buono o cattivo che sia! A questo punto, il programma di login fa la sua uscita di scena dalla console sulla quale stiamo lavorando (ma non dalle altre: provate a digitare Alt+FX dove alla X potete sostituire un numero normalmente da uno a sei, e vedrete altre console pronte a dare accesso ad altri utenti, o un ulteriore accesso a voi stessi) ed entra in gioco la shell. La shell altro non è che il ponte che unisce voi al sistema: tramite essa potrete impartire comandi e vederne l'output; senza una shell il sistema sarebbe lo stesso "vitale" ma voi avreste qualche problema in più a comandarlo! Le shell nel mondo unix sono moltissime (le trovate elencate nel file `/etc/shells`), sebbene la più utilizzata sia la bash: per vedere qual'è la shell che state utilizzando digitate

```
echo $SHELL
```

Prima di iniziare la carrellata di comandi, ricordiamo una cosa: noi saremo quanto più precisi possibile, ma non ci sarà possibile per evidenti motivi spiegare tutte le opzioni per ogni singolo comando: per ogni informazione aggiuntiva che vi interessi avere rivolgetevi alla pagine di manuale digitando

```
man comando
```

come abbiamo visto due capitoli fa! Ed è finalmente ora di iniziare con l'esplorazione dei più comuni comandi che il nostro Linux ci mette a disposizione. Il primo comando che vedremo, essenziale per muoversi fra le directory e certamente noto a molti, è `cd`. Senza argomenti, `cd` ci riporta alla nostra home directory; con un argomento, che dovrà essere un nome di directory, `cd` ci porterà all'interno della directory specificata. Utilizzato con l'argomento `-`, `cd` ci riporta alla directory di provenienza. Facciamo un esempio per chiarirne l'uso: digitiamo

```
cd /tmp
```

per entrare nella directory `/tmp`. A questo punto digitiamo

```
cd
```

per tornare nella nostra home directory.

Per finire, digitiamo

```
cd -
```

e ci vedremo ancora in `/tmp` !

Il secondo comando, anch'esso noto a chi abbia lavorato con un sistema DOS, è `dir`, che elenca file e directory presenti nella directory corrente, se non altrimenti specificato. Molto spesso, per non dire sempre, al posto di `dir` si usa il comando `ls` (abbreviazione per `list`): essi hanno non solo la stessa funzione, ma anche la stessa identica sintassi: lo potete constatare confrontando le pagine di manuale relative all'uno ed all'altro comando.

Normalmente, `ls` (o `dir`) visualizza i file in ordine alfabetico e non visualizza i file nascosti, quelli cioè il cui nome inizia con un punto: per vedere anche questi file, dobbiamo utilizzare l'opzione `-a` (`all`) che ci svelerà la presenza anche di due directory: `"."` e `".."`; la prima è la rappresentazione della directory corrente, la seconda della directory precedente nell'albero delle directory: così, con

```
ls .
```

vedremo la lista dei file e delle directory presenti nella directory corrente, con

```
ls ..
```

vedremo invece la lista dei file e delle directory (d'ora in poi generalizzeremo dicendo solo "file") presenti nella directory "madre" della directory corrente.

Inoltre, per vedere il contenuto di una directory senza doverci spostare, è ovviamente possibile digitare

```
ls /nome/della/directory
```

Ma torniamo alle opzioni di ls: con -l potremo utilizzare il "long listing format", che ci darà maggiori informazioni sui file: un output potrebbe essere questo:

```
-rw-r--r--    1 edo      edo          9306 mag 28 18:41 prova.txt
drwxr-sr-x    5 edo      edo          4096 gen 13 2001 GNUstep
drwx--S---    2 edo      edo          4096 apr 18 02:08 Mail
drwxr-sr-x    2 edo      edo          4096 lug 21 17:23 mirrored
drwx--S---    2 edo      edo          4096 gen  8 2001 nsmail
-rw-r-----   1 edo      edo        155403 ago 12 11:42 out.jpeg
drwxr-sr-x    2 edo      edo          4096 apr  9 01:56 prove
drwxr-sr-x    2 edo      edo          4096 mar  8 15:38 public_html
```

Qui notiamo molte informazioni in più su ogni file, non solo il suo nome! Ad esempio, vediamo a chi appartiene, la sua grandezza e la data di creazione. Per il blocco di informazioni riportate sulla sinistra non abbiate fretta: lo esamineremo in seguito; per il momento, sappiate solo che è un insieme di caratteri che definiscono le caratteristiche di ogni file.

Altre opzioni, forse meno utilizzate ma sempre utili sono:

- -h che esprime la grandezza del file in formato più facilmente leggibile (ad esempio, 2K, 32M, 4G ecc); non ha senso se utilizzato da solo, ma può essere utile accoppiato con -l;
- --color che distingue tramite colori i i tipi di file;
- -S per ordinare i file in ordine di grandezza discendente;
- -t per ordinarare i file in ordine di creazione discendente (i più recenti in alto)

Per avere tutte le possibili opzioni relative al comando ls, vi rimandiamo alla pagina di manuale del comando stesso.

---

Ora che abbiamo visto come muoverci nel filesystem e come vedere il contenuto di una directory, concentriamoci sui singoli file e sulle operazioni che possiamo fare su di essi: vedremo quindi i comandi per copiare, spostare ed eliminare i file presenti sul disco; inoltre, vedremo come eseguire le stesse operazioni sulle directory, imparando anche come crearle.

Il primo comando è cp, utilizzato per copiare file e directory. La sua sintassi è molto semplice:

```
cp file_sorgente destinazione
```

oppure, per la copia multipla di file, si usa:

```
cp file1 file2 file3 ... fileN destinazione
```

che permetterà di copiare i file file1 .. fileN nella directory di destinazione. Vediamo un esempio:

```
cp miofile /tmp
```

che copierà 'miofile' nella directory /tmp. Nel caso in /tmp esista già un file chiamato miofile, questo verrà sovrascritto dal nuovo file. Vediamo ora le opzioni più importanti di cp:

- -b Esegue automaticamente una copia di backup di ogni file di destinazione esistente.
- -f Forza la sovrascrittura dei file, senza richiedere interventi da parte dell'utente.
- -i Attiva la modalità interattiva, che chiede conferma prima dell'eventuale sovrascrittura di file di destinazione preesistenti; il suffisso usato è il classico simbolo della tilde, ~.
- -p Mantiene, se possibile, gli attributi (permessi, ownership ecc.) del file.
- -r Permette di attivare la modalità ricorsiva, che permette la copia di directory.
- -v Attiva la modalità verbose, che visualizza in output quello che il sistema ha fatto in seguito al nostro comando.

Dopo aver visto come copiare file e directory, dedichiamoci allo spostamento ed al cambio di nome dei file; tutto questo è fatto con un solo comando: mv, la cui sintassi è:

```
mv SORGENTE DESTINAZIONE (per rinominare)
mv SORGENTE DIRECTORY    (per spostare un file)
mv SORGENTE1 .. SORGENTE-N DESTINAZIONE (per spostare più file)
```

Tramite mv, oltre a cambiare il nome di un file, è anche possibile rinominare una directory o muovere più directory in un'altra directory. Le opzioni di mv sono molto simili a quelle di cp, sebbene in numero minore. Quelle che a noi interessano sono le stesse viste per il comando cp, con l'esclusione di -p che per mv non esiste. Come ultimo, ci rimane il comando rm, che permette di eliminare file o directory. La sua sintassi è abbastanza semplice:

```
rm FILE (per eliminare un file)
rm FILE1 .. FILE-N (per eliminare più file)
```

Le opzioni di rm sono anch'esse simili a quelle di mv; le più utili sono le stesse che sono state viste per cp, con l'esclusione di "-p" che qui non ha alcun senso pratico. Deve qui essere aperta una parentesi relativa alle directory: di norma, se siete sicuri di poter eliminare una directory non vuota, potete utilizzare il comando

```
rm -rf directory
```

che forza l'eliminazione della directory 'directory' e dei suoi contenuti. Per eliminare una directory vuota, si usa invece il comando:

```
rmdir directory
```

A volte, però, tale comando non sembra sortire gli effetti desiderati: nonostante la directory sembri vuota, quanto scritto sopra ci restituisce un errore, dove il sistema si lamenta perchè esistono dei file all'interno della directory che vogliamo eliminare. Accade spesso, infatti, che per controllare il contenuto di una directory si usi solamente "ls" che, per sua natura, non visualizza i file nascosti! A questo punto si può procedere in due modi: entrare nella directory e cancellare tali file oppure, se si è sicuri che i file nascosti all'interno di tale directory non sono importanti, si può procedere con rm -rf. Come ultima cosa, vediamo come creare le directory: il comando è mkdir, la cui sintassi è:

```
mkdir DIRECTORY (per creare una directory)
mkdir DIRECTORY1 .. DIRECTORY-N (per creare più directory).
```

Vediamo un esempio: nella nostra home directory, supponiamo /home/utente, è presente la directory "dir" e vogliamo creare "sottodir" come sottodirectory della precedente. La prima cosa che viene in mente di fare è di entrare in "dir" e lanciare il comando:

```
mkdir sottodir
```

È però possibile, e anche molto più comodo, utilizzare direttamente il comando:

```
mkdir dir/sottodir
```

con una sola attenzione: il comando

```
mkdir dir/sottodir sottodir2 sottodir3
```

non fa quello che ci si potrebbe aspettare, ossia creare sottodir, sottodir2 e sottodir3 come sottodirectory di "dir"; quello che qui accade, invece, è la creazione di "sottodir" come sottodirectory di "dir" e la creazione di "sottodir2" e "sottodir3" allo stesso livello di "dir"! Quindi, per non dover scrivere:

```
mkdir dir/sottodir dir/sottodir2 dir/sottodir3
```

in questo caso è più comodo entrare in dir e lanciare:

```
mkdir sottodir sottodir2 sottodir3
```

Con queste nozioni, potrete certamente iniziare a fare gli esperimenti sulla creazione, rimozione e spostamento di file e directory. Permetteteci però qualche consiglio:

- utilizzate touch per creare file vuoti con cui operare; il comando "touch file" crea il file vuoto "file"; utilizzate invece il comando "mkdir" (mkdir nomedirectory) per la creazione delle directory;
- spostatevi in /tmp per le prove; è più sicuro e vi eviterà di lasciare file e directory inutili sparsi per il filesystem o, peggio, di eliminare file importanti o di sistema per qualche disattenzione;
- non, ripeto, NON utilizzate l'utente root per queste prove: uno stupido errore di digitazione può, come si dice in gergo, "rasare" un intero filesystem!

In conclusione, ricordiamo che tutti i comandi fino a qui visti permettono di specificare caratteri jolly, quali ad esempio l'asterisco o il punto di domanda. Ad esempio:

```
rm ab*
```

rimuoverà tutti i file il cui nome inizia con "ab" seguiti da zero o più caratteri. Invece, il comando:

```
rm ab?
```

eliminerà tutti i file il cui nome inizia con ab seguiti da un qualsiasi carattere.

Nelle due precedenti lezioni abbiamo esaminato i comandi principali per muoverci all'interno del filesystem e per agire su file e directory. Dedicheremo questa pagina ad altri comandi non meno utili e legati a diverse funzioni relative al nostro sistema Linux.

Presenteremo in questa pagina una lista in ordine alfabetico, senza soffermarci troppo sulle opzioni di ognuno di essi; per ogni informazione, si rimanda alle pagine di manuale relative ad ogni comando.

### apropos

Visualizza una lista di elementi correlati a quello che gli si passa come argomento. Ad esempio, "apropos bash" mostrerà tutte le pagine di manuale relative a bash.

### bzip2, bunzip2

Permette di creare e decomprimere archivi in formato ".bz2", molto usati per il buon grado di compressione che tale formato offre. Maggiori informazioni le potrete reperire sul nostro articolo

### cat

Permette di leggere il contenuto di un file. Il comando cat non funziona da pager, ed è quindi molto spesso sostituito da comandi come less o more.

### chmod

Con chmod si possono modificare gli attributi di file e directory. E' ad esempio possibile, tramite questo comando, rendere eseguibile un file, negare a determinati utenti la sua lettura ecc.

### chown

Modifica l'owner di un file o di una directory.

### date

Senza argomenti visualizza la data corrente; può inoltre essere utilizzato per la modifica della data e dell'ora del sistema.

### fdformat

Formatta un floppy.

**find**

find è il comando utilizzato per la ricerca di file all'interno del filesystem.

**free**

Visualizza informazioni sulla memoria di sistema, che comprende sia la RAM che lo swap.

**grep**

Grep permette di ricercare determinati pattern all'interno di uno o più file. E' un comando molto utile, con il quale vi consigliamo di familiarizzare il prima possibile!

**gzip, gunzip**

Questi due comandi permettono di creare ed estrarre archivi in formato .gz.

**kill**

Invia un segnale di terminazione ad un processo. Si usa spesso per terminare (con l'opzione -9) processi che non rispondono più.

**less**

Permette di leggere il contenuto di un file ed esegue la paginazione sullo stesso (in pratica, ferma la visualizzazione dopo che una schermata è stata riempita con il contenuto del file ed aspetta a passare alla successiva finché non gli viene esplicitamente richiesto dall'utente).

**ln**

Crea link simbolici ed hard link fra file all'interno del filesystem.

**more**

Analogo di less, che però non permette di scorrere un file all'indietro.

**mount, umount**

Comandi utilizzati per montare e smontare devices (ad esempio partizioni, cdrom ecc.)

**pwd**

Visualizza la directory corrente.

**tail**

Visualizza le ultime linee di un file. Con l'opzione "-f" continua a visualizzare il file mano a mano che alla fine dello stesso vengono aggiunte linee. Spesso si usa per monitorare in realtime i file di log del sistema.

**tar**

Crea ed estrae archivi di file. Il comando tar non comprime i file, perciò spesso è utilizzato assieme a comandi come gzip e bzip2.

**uptime**

Visualizza da quanto tempo il sistema è in esecuzione, riportando anche il carico medio negli ultimi 1, 5 e 15 minuti.

**whereis**

Visualizza la locazione dell'eseguibile, della pagina di manuale ed altre informazioni relative al comando passatogli come argomento.

Questa è una lista, ovviamente non completa, dei comandi principali; non ha pretese di essere completa (un'installazione di base di linux comprende più di mezzo migliaio di eseguibile, ai quali se ne aggiungono altri mano a mano che si installano altri programmi).

**Ref:** <http://linux.html.it/>