

CAPITOLO 7

ESPLORIAMO IL FILESYSTEM

Pietro Buffa

La prima cosa che bisogna imparare bene è come spostarsi all'interno di tutte quelle directory che compongono l'apparentemente intricato filesystem.

Occorrono tre cose:

1. Memorizzare alcuni comandi di fondamentale importanza.
2. Ricordare sempre che Linux è *case sensitive*, ossia distingue i caratteri maiuscoli da quelli minuscoli.
3. Fare pratica.

7.1 IL COMANDO `cd`

Il comando `cd` (change directory), consente di spostarsi tra le directory. Il suo utilizzo è semplice e la sintassi è la seguente:

```
> cd [percorso]
```

Ma cosa intendiamo per percorso?

Come abbiamo visto nel precedente capitolo, in GNU/Linux, tutti i file sono memorizzati all'interno di specifiche directory del filesystem, a partire dalla directory root “/”. Per spostarci da una directory all'altra, bisogna digitare lo specifico percorso (path), seguendo i rami del filesystem ed utilizzando una barra trasversale “/” (slash), per separare i nomi delle directory.

Linux distingue due tipi fondamentali di percorso:

- **Percorso assoluto**, quello che inizia a livello della directory base root “/”, ed elenca tutte le sottodirectory fino ad arrivare a quella di destinazione. Ogni volta che il simbolo “/” è il primo carattere di un percorso, siamo di fronte ad un percorso di tipo assoluto.

Vediamo alcuni esempi:

```
bash:/home/bios/immagini > cd /usr/local
```

Con tale comando l'ipotetico utente “bios” si vuole spostare dalla directory “immagini”, in cui si trovava, alla directory “local” contenuta all'interno della directory “usr” che a sua volta è contenuta all'interno della directory base root “/”. Adesso il prompt dei comandi visualizza il nuovo percorso e quindi la nuova directory di lavoro corrente:

```
bash:/usr/local >
```

Nel caso in cui ci trovassimo di fronte ad un terminale in cui il prompt non visualizzi la directory di lavoro corrente, possiamo utilizzare il comando `pwd`, che già conosciamo, per sapere sempre dove ci troviamo all'interno del filesystem.

- **Percorso relativo**, richiede un viaggio più breve, potete usare il percorso relativo per iniziare a spostarvi dalla directory di lavoro in cui siete e procedere da lì fino ad arrivare alla directory di destinazione. Un percorso relativo comincia sempre con un nome di directory e mai con il simbolo “/”.

Vediamo alcuni esempi:

```
bash:/home > cd bios/immagini
```

Il prompt evidenzierà questo spostamento:

```
bash:/home/bios/immagini >
```

ATTENZIONE: Esistono poi due “directory speciali”, esse sono: “.” e “..”. Linux interpreta il puntino singolo come la directory di lavoro corrente (quella in cui ci troviamo), mentre i due puntini come la directory superiore.

L’inserimento del comando:

```
> cd .          vi fa rimanere nella directory corrente.
```

Mentre:

```
> cd ..         vi fa risalire alla directory superiore nel filesystem.
```

Supponete adesso di essere l’ipotetico utente bios nella vostra home directory e di voler passare alla directory immagini dell’utente tux (sempre che vi sia consentito), possiamo scrivere:

```
bash:~ > cd ../tux/immagini
```

I due puntini indicano alla shell di passare alla directory superiore, che è */home*, poi dico di andare avanti per giungere alla sottodirectory tux e quindi alla sottosottodirectory immagini, che è la destinazione finale.

Scrivendo invece soltanto il comando:

```
> cd
```

Si torna automaticamente alla propria directory home.

Il simbolo “~” (tilde), rappresenta la vostra personale directory home e può essere quindi utilizzato come notazione breve.

7.2 IL COMANDO **ls**

Il comando **ls** (list) è l’equivalente del comando “dir” del vecchio Ms-DOS. Serve a visualizzare il contenuto delle directory in ordine alfabetico. E’ sicuramente il comando più usato, del resto, che senso avrebbe spostarsi in una directory se poi non se ne può vedere il contenuto?

Come tutti i comandi ha tantissime opzioni che ne modificano il comportamento in base alle nostre esigenze, ci riserviamo dunque di ricordare all’utente che volesse approfondire la conoscenza di un determinato comando, di consultare le relative pagine man.

La sua sintassi è la seguente:

```
> ls [opzioni] [argomento]
```

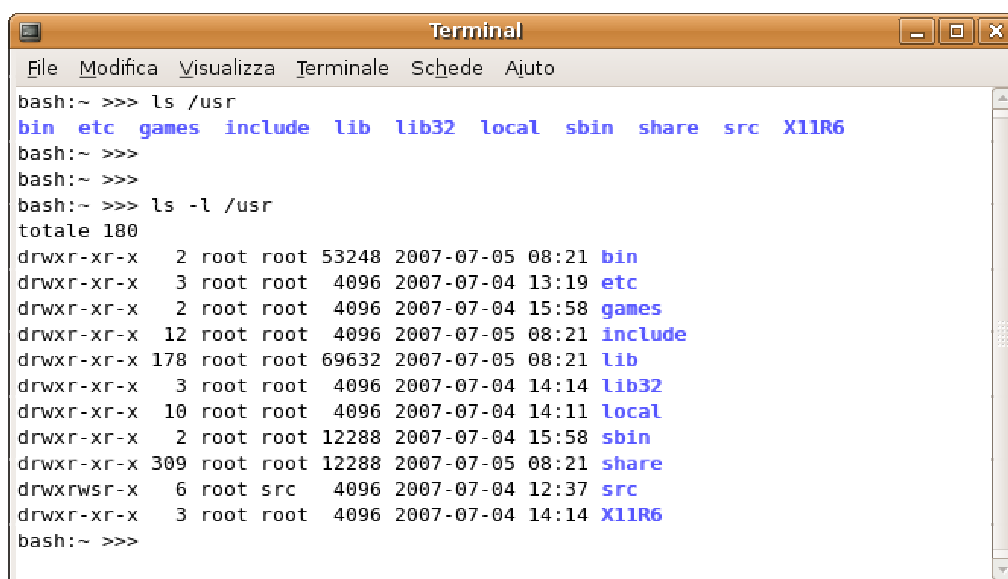
Utilizzando il comando, senza specificare alcuna opzione o argomento, esso visualizza i nomi di tutti i files e di tutte le directory presenti all’interno della directory di lavoro corrente. Utilizzando qualche opzione possiamo rendere ancora più utile questo comando.

OPZIONI:

Le opzioni permettono di visualizzare una notevole varietà di informazioni in differenti formati, non dimenticate che possono essere adoperate anche più opzioni insieme in base a ciò che si vuole ottenere.

- l E’ l’opzione più utilizzata, stampa l’output nel formato “lungo” includendo moltissime informazioni importanti e disponendo files e directory in colonna.

Ecco ad esempio come si presenta il contenuto della directory `/usr` senza l'uso di opzioni ed utilizzando l'opzione `-l`, Fig. 7.1 in basso.



```
bash:~ >>> ls /usr
bin etc games include lib lib32 local sbin share src X11R6
bash:~ >>>
bash:~ >>>
bash:~ >>> ls -l /usr
totale 180
drwxr-xr-x  2 root root 53248 2007-07-05 08:21 bin
drwxr-xr-x  3 root root  4096 2007-07-04 13:19 etc
drwxr-xr-x  2 root root  4096 2007-07-04 15:58 games
drwxr-xr-x 12 root root  4096 2007-07-05 08:21 include
drwxr-xr-x 178 root root 69632 2007-07-05 08:21 lib
drwxr-xr-x  3 root root  4096 2007-07-04 14:14 lib32
drwxr-xr-x 10 root root  4096 2007-07-04 14:11 local
drwxr-xr-x  2 root root 12288 2007-07-04 15:58 sbin
drwxr-xr-x 309 root root 12288 2007-07-05 08:21 share
drwxrwsr-x  6 root src   4096 2007-07-04 12:37 src
drwxr-xr-x  3 root root  4096 2007-07-04 14:14 X11R6
bash:~ >>>
```

Fig. 7.1 Esecuzione del comando `ls` sulla directory `/usr`.

Se si esamina attentamente la figura in alto, nella zona relativa all'output del comando `ls -l`, si può notare come una tabella in cui ogni file e ogni directory sta su uno specifico record che a sua volta è composto da tante parti. Vediamo di analizzarle:

- La prima colonna è composta da una simbologia abbastanza strana, non preoccupatevi per il momento di capire cosa significhi.
- La seconda colonna riporta il numero di files che sono collegati simbolicamente a quel determinato file, non preoccupatevi di capire nemmeno questo, notate però che sono tutti 1. Nel caso di una directory, il numero si riferisce alle sottodirectory contenute (una directory contiene sempre almeno le due sottodirectory speciali, `."` e `.."`), infatti il numero di molte directory è 2.
- La terza colonna indica il proprietario del file, in questo caso sempre root.
- La quarta colonna riporta il gruppo proprietario del file.
- La quinta colonna riporta la dimensione del file in byte.
- La sesta e la settima colonna mostrano la data e l'ora della creazione o dell'ultima modifica del file o della directory.
- L'ottava colonna contiene infine il nome del file o della directory.

-a E' l'opzione che mostra i file nascosti.

Normalmente la shell è già impostata in maniera tale da visualizzare i files con colori differenti in base al loro significato:

- I files ordinari, vengono generalmente colorati in nero.
- I files eseguibili, vengono generalmente colorati in verde.
- Le directory, vengono generalmente colorate in blu.
- I collegamenti, vengono generalmente colorati in azzurro con vicino una piccola freccia che indica il file puntato.

-i E' l'opzione che elenca gli inode di ogni file, argomento tecnico che verrà trattato più avanti nel corso di questo manuale.

-h Vengono indicate le dimensioni dei files in Byte, KByte, MByte, GByte.

-F Contrassegna files, link e directory con dei simboletti, in modo da identificarli facilmente qualora la shell non dovesse essere abilitata per il colore.

Vediamo adesso di fare qualche esempio pratico per renderci conto di come funziona realmente il comando `ls`:

Immaginiamo di essere nuovamente l'ipotetico utente `bios` e di trovarci nella nostra personale `home directory`. Vogliamo vedere quanti utenti ha il sistema su cui stiamo lavorando, per far questo basta dare un'occhiata al contenuto della `directory /home`.

Possiamo farlo in due modi:

```
bash:~ > ls -l /home    ...utilizzando il percorso assoluto.
```

```
bash:~ > ls -l ..      ...utilizzando il percorso relativo (in questo caso basta salire alla
                        directory genitore, cioè alla directory home).
```

ATTENZIONE: Vi siete accorti che per vedere il contenuto della `directory home` non ci siamo prima dovuti spostare al suo interno? Il comando `ls` mostra il contenuto della `directory` che vogliamo, basta dargli, come argomento, il giusto percorso.

Adesso che abbiamo appreso come muoverci e come sbirciare il contenuto delle `directory` ci sentiamo un pò più padroni del nostro sistema Linux. Ma cosa succede se volessimo vedere il contenuto della `directory /dev`? Niente di più facile! Lancio il comando: `cd /dev` ed una volta dentro lancio un: `ls -l` per vederne il contenuto. Provate a farlo. Questa `directory` è talmente piena di files che non riuscirete a leggerne nemmeno uno e questo accade perché la lista richiede più spazio di quello che il monitor può offrire.

Come fare allora a leggere il contenuto di `directory` così piene?

Purtroppo non c'è modo di dire al comando `ls` di darvi il tempo di leggere tutto. Però si può aggirare il problema prendendo l'output di `ls` ed indirizzarlo ad un programma più educato. Fate particolare attenzione a questa procedura:

```
bash:~ > ls -l /dev | less
```

Adesso è il programma `less` ad occuparsi della visualizzazione della lista dei files contenuti all'interno di `/dev`. Questo magico effetto è stato ottenuto grazie al simbolo “|” (pipe), che crea una sorta di “canale di comunicazione” tra programmi diversi. E' come se l'output di `ls` sia stato incanalato attraverso un condotto e fatto arrivare fino al programma `less` che si occupa di visualizzare in maniera più comoda i files.

Questo è un esempio di “**comando complesso**” ossia fatto da più istruzioni legate insieme.

Impareremo più avanti ad apprezzare la reale importanza degli “operatori di reindirizzamento”, così come parleremo più approfonditamente del programma `less` e dei suoi fratelli minori, nel prossimo capitolo dedicato ai files.