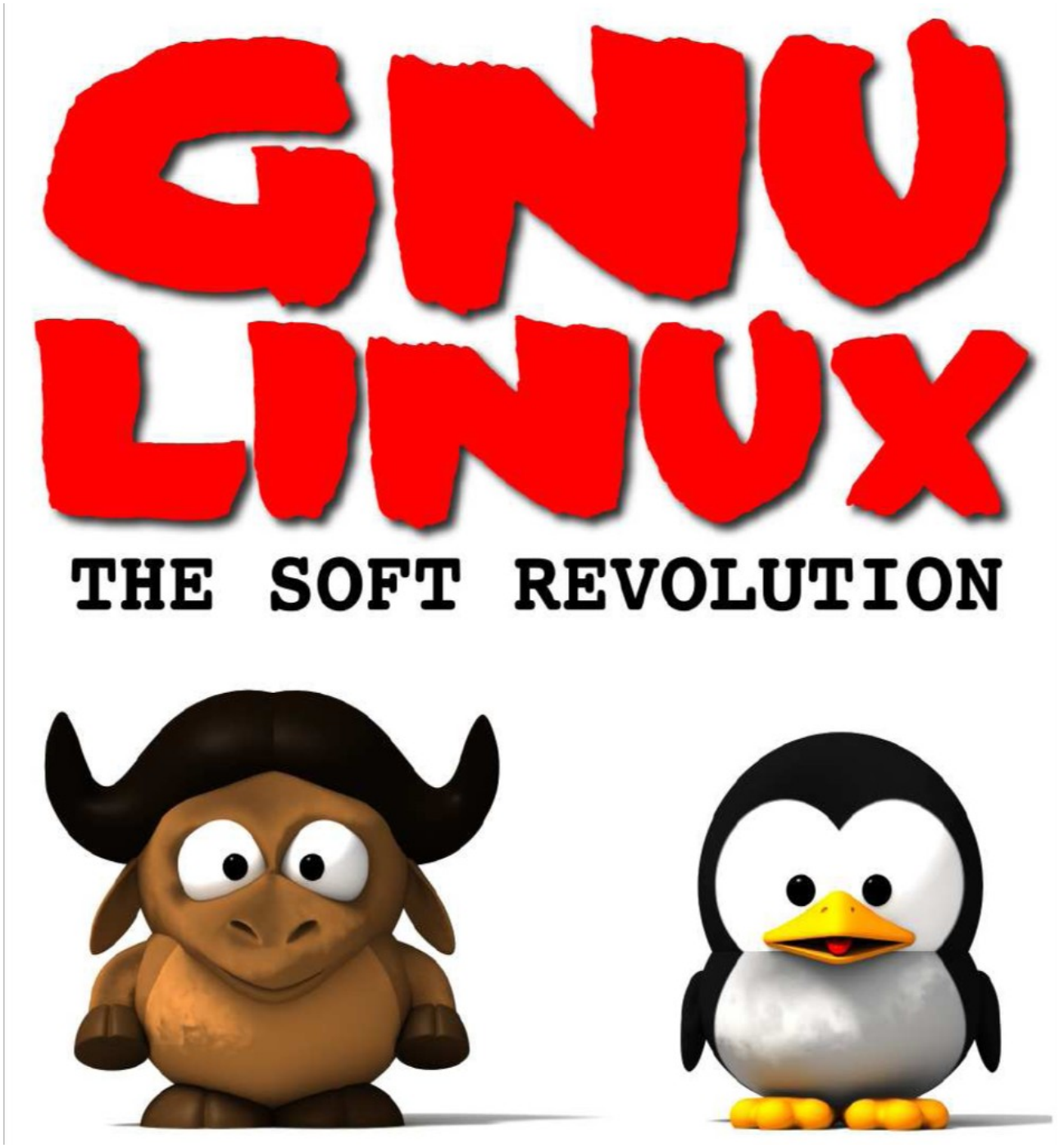


Prontuario dei principali comandi

Questa guida è distribuita durante i corsi GNU/Linux gratuiti tenuti al Politecnico di Torino

http://netstudent.polito.it

Introduzione

Questo prontuario ha come scopo la presentazione schematica dei **comandi base** della *shell* GNU/Linux. Per tale ragione costituisce più un riferimento per l'identificazione dei comandi che una guida introduttiva alla shell.

I comandi all'interno di questo prontuario devono essere eseguiti dalla riga di comando.

In ambiente GNU/Linux esistono le GUI (Graphical User Interfaces), dove è possibile puntare, cliccare, trascinare e svolgere il proprio lavoro senza prima aver letto una montagna di documentazione. In ogni caso l'ambiente di lavoro tradizionale di Unix è la CLI (Command Line Interface), dove si scrivono le istruzioni che il computer dovrà eseguire. Questo metodo è molto più veloce e potente, ma richiede una certa conoscenza riguardo i comandi da utilizzare.

GNU/Linux è case sensitive. «User», «user», e «USER» sono tutte cose diverse.

Ottenere maggiore aiuto

Per ottenere maggiore aiuto o informazioni riguardo un determinato comando, esiste il comando **man** che serve per visualizzare il manuale di un determinato comando.

La sintassi del comando **man** è la seguente:

```
man [comando]
```

Ad esempio, per visualizzare la pagina di manuale dello stesso comando **man** è sufficiente digitare il seguente comando:

```
man man
```

Una volta all'interno del manuale, per poter spostarsi al suo interno, basta utilizzare le frecce direzionali. Per uscire dal manuale premere il tasto «**q**».

Quasi tutti i comandi accettano anche l'opzione **-h** (o **--help**) che fornisce una breve descrizione sull'utilizzo del comando e delle sue opzioni.

Gestione di file e directory

pwd serve per mostrare la directory in cui ci si trova.

La sintassi del comando è la seguente:

```
pwd [opzioni]
```

ls serve per elencare il contenuto di una directory.

La sintassi del comando è la seguente:

```
ls [opzione] [directory]
```

Alcune opzioni da utilizzare con il comando **ls** ed il loro effetto.

Opzione	Risultato
[directory]	elenca il contenuto della directory specificata, se non specificata viene considerata la directory corrente
-a	elenca anche i file nascosti
-l	elenco dettagliato di file e sotto directory con i loro attributi
-R	elenca ricorsivamente i file nella directory indicata e in tutte le sottodirectory
-s	mostra la dimensione dei file

-S	ordina i file per dimensione partendo dal più grande
-u	ordina i file per data e ora di accesso partendo dal più recente
-X	ordina i file per estensione e ordine alfabetico
-r	elenca i file invertendone l'ordine
--color	mostra i file con colori differenti

cd serve per spostarsi all'interno delle directory del filesystem.

La sintassi del comando è la seguente:

```
cd [directory]
```

Alcuni esempi di uso del comando:

- `cd ..`

  Serve per spostarsi alla directory superiore.

- `cd`

  Serve per spostarsi, da qualsiasi punto, alla propria directory home. È equivalente a:

  `cd ~`

- `cd /etc`

  Serve per spostarsi nella directory `/etc`.

mkdir serve per creare directory all'interno del filesystem.

La sintassi del comando è:

```
mkdir [opzioni] directory
```

Alcuni esempi di uso del comando **mkdir**:

- `mkdir prova`

  Verrà creata la directory `prova/` all'interno della directory corrente.

- `mkdir ~/prova`

  Verrà creata la directory `prova` all'interno della propria home directory, qualunque sia la directory in cui ci si trova al momento.

- `mkdir -p prova1/prova2/prova3/bin`

  Qualora non esistessero, verranno create anche tutte le directory intermedie, a partire dalla directory corrente.

cp serve per:

- copiare un file in un altro file;
- copiare un file in un'altra directory;
- copiare più file in un'altra directory;
- copiare directory.

La sintassi del comando è la seguente:

```
cp [opzioni] origine dest.
```

Alcune opzioni da utilizzare con il comando **cp**:

Opzione	Risultato
-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 preesistenti
-p	mantiene, se possibile, gli attributi del file
-r	permette di attivare la modalità ricorsiva, consentendo la copia di directory

Alcuni esempi di uso del comando **cp**:

- `cp /prova/miofile /prova1`

  Copia il file `miofile` della directory `prova` nella directory `/prova1`.

- `cp /prova/miofile /prova1/nuovofile`

  Copia il file `miofile` della directory `/prova` nella directory `/prova1` dandogli il nome `nuovofile`.

- `cp -r /prova /prova_copia`

  Copia la cartella `/prova`, e tutto il suo contenuto, nella cartella `/prova_copia`.

mv serve per spostare, o rinominare, file e directory. La sintassi del comando è la seguente:

- `mv [opzioni] origine dest.`

Le opzioni sono le stesse del comando cp.

Alcuni esempi di uso del comando **mv**:

- `mv miofile nuovofile`

  Cambierà il nome al file `miofile` in `nuovofile`.

- `mv miofile /prova`

  Sposterà il file `miofile` nella directory `/prova` sovrascrivendo un eventuale file con lo stesso nome.
- `mv /prova /prova_nuova`

  Cambierà il nome alla directory `/prova` in `/prova_nuova`.

rm serve per cancellare file o directory dal file system. La sintassi del comando è la seguente:

```
rm [opzioni] file …
```

Alcune opzioni da utilizzare con il comando **rm**:

Opzione	Risultato
-i	chiede conferma prima di cancellare
-f	forza la cancellazione del file senza chiedere conferma
-r	abilita la modalità ricorsiva usata per la cancellazione delle directory

rmdir serve per cancellare directory dal file system.

La sintassi del comando è la seguente:

```
rmdir directory
```

Alcuni esempi di uso dei comandi **rm** e **rmdir**:

- `rm miofile`

  Cancella il file `miofile`.
- `rm -rf prova/`

  Cancella la directory `prova/` e tutto il suo contenuto.
- `rmdir prova/`

  Cancella la directory `prova/` solo se questa non contiene alcun file all'interno.

touch serve per aggiornare la data dell'ultimo accesso o quello dell'ultima modifica di un file.

La sintassi del comando è la seguente:

```
touch [opzioni] file
```

Alcune opzioni per il comando **touch**:

Opzione	Risultato
-a	cambia solo la data dell'ultimo accesso
-c	non creare il file
-m	cambia solo la data dell'ultima modifica
-t STAMP	specifica la data nel formato «[[CC]YY]MMDDhhmm[.ss]»

Alcuni esempi di uso del comando:

- `touch miofile`

  Nel caso esista un file di nome `./miofile` la data e l'ora di ultima modifica verranno impostate a quelle correnti. In caso contrario verrà creato un nuovo file.
- `touch -t 0702211100 miofile`

  Imposta come data e ora di ultima modifica del file `./miofile` alle ore 11.00 del 21 febbraio 2007.

ln serve a creare un collegamento (o *link*) ad un file o una directory.

Un collegamento è un file speciale che non contiene dati, ma solo un riferimento ad un altro file: ogni operazione effettuata sul collegamento viene in realtà eseguita sul file a cui punta.

La sintassi del comando è la seguente:

```
ln -s /path_file/linked_file
/path_link/link_name
```

L'opzione **-s** specifica che verrà creato un collegamento simbolico.

chmod è il comando da utilizzare per la modifica dei permessi.

La sintassi del comando è la seguente:

```
chmod [OPZIONI] permessi nomefile
```

Ci sono due metodi per modificare i permessi, attraverso l'uso dei numeri o delle lettere. Non è consigliabile modificare i permessi ai file di sistema, alcuni file hanno dei permessi molto restrittivi per scongiurare accessi non autorizzati e problemi di sicurezza. Ad esempio, il file `/etc/shadow`, che contiene le password

utente, non ha impostato alcun permesso per gli utenti.

chmod con i letterali

Quello che segue è il dizionario dei letterali da usare con i permessi:

Opzioni	Definizione
u	proprietario
g	gruppo
o	altri
x	esecuzione
w	scrittura
r	lettura
+	aggiungi permesso
-	annulla permesso
=	imposta permesso

Quelli che seguono sono degli esempi di utilizzo del comando **chmod** con i letterali: per prima cosa è utile creare alcuni file vuoti. Digitare il seguente comando:

```
touch file1 file2 file3 file4
```

Al fine di rendersi conto di come variano i permessi si può utilizzare il comando **ls -l** dopo ogni applicazione del comando **chmod**. Appena creati i file, l'output del comando **ls** sarà simile al seguente:

```
total 0
-rw-r--r--  1 user user 0 Nov 19 20:13 file1
-rw-r--r--  1 user user 0 Nov 19 20:13 file2
-rw-r--r--  1 user user 0 Nov 19 20:13 file3
-rw-r--r--  1 user user 0 Nov 19 20:13 file4
```

Aggiungere il bit di esecuzione al «proprietario»:

```
chmod u+x file1
```

Aggiungere agli «altri» i bit di scrittura ed esecuzione:

```
chmod o+wx file2
```

Negare al «gruppo» il bit di lettura:

```
chmod g-r file3
```

Aggiungere i bit di lettura, scrittura ed esecuzione a tutti gli utenti:

```
chmod ugo+rwx file4
```

chmod con i numeri

Opzioni	Definizione
#--	proprietario
-#-	gruppo
--#	altri
1	esecuzione

«Proprietario», «gruppo» e «altri» sono rappresentati da tre numeri. Per ottenere il valore da impostare, è sufficiente determinare la tipologia di accesso e poi fare la somma.

Per esempio, se si desidera un file con i permessi «-rw-rw-rwx» è necessario utilizzare la seguente combinazione:

Proprietario	Gruppo	Altri
lettura e scrittura	lettura e scrittura	lettura, scrittura ed esecuzione
4+2=6	4+2=6	4+2+1=7

Dunque, il comando da digitare sarà il seguente:

```
chmod 667 nomefile
```

Se invece si desidera un file con i permessi «--w-r-x--x» è necessario utilizzare la seguente combinazione:

Proprietario	Gruppo	Altri
scrittura	lettura ed	esecuzione
2	4+1=5	1

Per applicare tali permessi sarà necessario digitare il seguente comando:

```
chmod 251 nomefile
```

Quelli che seguono sono degli esempi di utilizzo del comando **chmod** con i numeri: per prima cosa creare alcuni file vuoti con il seguente comando:

```
touch file1 file2 file3 file4
```

I permessi di tali file, visualizzabili con **ls**, corrispondono ai seguenti:

```
total 0
-rw-r--r--  1 user user 0 Nov 19 20:13 file1
-rw-r--r--  1 user user 0 Nov 19 20:13 file2
-rw-r--r--  1 user user 0 Nov 19 20:13 file3
-rw-r--r--  1 user user 0 Nov 19 20:13 file4
```

Aggiungere il bit di esecuzione al «proprietario»:

```
chmod 744 file1
```

Aggiungere ad «altri» i bit lettura ed esecuzione:

```
chmod 647 file2
```

Negare a «gruppo» il bit lettura:

```
chmod 604 file3
```

Aggiungere i bit lettura, scrittura ed esecuzione a tutti:

```
chmod 777 file4
```

Funzioni di ricerca

find serve per cercare all'interno di una directory e delle sue sottodirectory i file che soddisfano i criteri stabiliti dall'utente.

La sintassi del comando è la seguente:

```
find [directory] [espressione]
```

Alcuni esempi di usi del comando:

- `find . -name '*.mp3'`

  Cerca all'interno della directory corrente `./` tutti i file con estensione `.mp3`.

- `find . -perm 664`

  Cerca all'interno della directory corrente tutti i file con permessi di lettura e scrittura per il proprietario e il gruppo, ma solo di lettura per gli altri utenti.

- `find . -name '*.tmp' -exec rm {} \;`

  Cerca ed elimina tutti i file temporanei all'interno della directory corrente.

- `find /tmp -user pippo`

  Cerca tutti i file appartenenti all'utente specificato.

grep sintassi generale:

```
grep [opzioni] [-e] modello1 [-e
      modello2 …] [--] [file1 [file2
      …] ]
```

I parametri facoltativi *file* indicano i nomi dei file su cui effettuare la ricerca. Se non specificati, la ricerca viene effettuata sui dati letti dallo standard input. Specificando più di un parametro *file*, ogni linea per cui è stata trovata una corrispondenza viene preceduta dal nome del file che la contiene e dal suo numero di linea; in caso di un solo parametro *file* (o nessuno) viene invece indicato solo il contenuto della linea stessa.

I parametri *modello* specificano il criterio di ricerca, ed il comportamento predefinito prevede che si tratti di espressioni regolari. Una linea trova corrispondenza se soddisfa almeno uno dei modelli.

Il doppio trattino -- (facoltativo) indica che i parametri successivi non sono da considerarsi opzioni.

Tra le opzioni principali vi sono:

- `-i` Ignora le differenze tra lettere maiuscole e minuscole.
- `-n` Precede ogni linea dei risultati con il numero di linea all'interno del file (partendo da 1).
- `-l` Indica solo i nomi dei file in cui è stata trovata almeno una corrispondenza (ciascun file è elencato una sola volta, indipendentemente dal numero di corrispondenze in esso trovate).
- `-v` Nega i modelli specificati, producendo un elenco delle linee che non soddisfano alcun modello.
- `-E` I modelli sono espressioni regolari estese invece che espressioni regolari di base.
- `-F` I modelli sono stringhe che vanno ricercate in maniera letterale.
- `-c` Produce per ciascun file solo il conteggio del numero di linee che corrispondono.

Gestione dei file compressi

tar serve per aprire degli archivi di file con estensione `.tar` o per creare degli archivi. Solitamente, un archivio contenente dei file viene successivamente compresso. L'estensione, in base al programma di compressione, risulta:

- **.tar.bz2** se compresso con bunzip2
- **.tar.gz** se compresso gzip

La sintassi del comando è la seguente:

```
tar [OPZIONE]... [FILE]...
```

Alcune opzioni da utilizzare con il comando **tar**:

Opzione	Risultato
-c	crea un nuovo archivio
-r	aggiunge i file all'archivio
-x	estrae i file da un archivio
-t	elenca tutti i file in un archivio
-f archivio.tar	utilizza come archivio il file archivio.tar
-v	elenca tutti i file processati

Alcuni esempi di uso del comando **tar**:

- `tar -cf archivio.tar miofile.txt miofile.bin`

  Crea l'archivio `archivio.tar` contenente i file `miofile.txt` e `miofile.bin`.
- `tar -xf archivio.tar`

  Estrae tutti i file dall'archivio `archivio.tar`.
- `tar -tvf archivio.tar`

  Visualizza tutti file contenuti nell'archivio `archivio.tar`.
- `tar -xvf archivio.tar.gz`

  Estrae tutti i file dall'archivio, indipendentemente dal formato di compressione (gzip o bzip2).
- `tar -xvfz archivio.tar.gz`

  Estrae tutti i file dall'archivio `archivio.tar.gz` compresso con **gzip**.
- `tar -xvfj archivio.tar.bz2`

  Estrae tutti i file dall'archivio `archivio.tar.bz2` compresso con **bunzip2**.

gzip e **gunzip** sono utili, rispettivamente, a comprimere e decomprimere i file nel formato `.gz`.

La sintassi dei comandi **gzip** e **gunzip** è la seguente:

```
gzip [OPZIONE]... [FILE]...
```

Alcuni esempi di uso del comando **gzip**:

- `gzip -r archivio.gz prova.txt /home/daniel/Doc`

  Inserisce nell'archivio `archivio.gz` il file `prova.txt` e la cartella `/home/daniel/Doc`.
- `gzip -d miofile.gzip`

  Decomprime il file `miofile.gz`.
- `gzip -9 miofile.txt`

  Comprime in modalità «**-9**» (miglior compressione) il file `miofile.txt` creando il file `miofile.txt.gz`.

È possibile modificare il tipo di compressione, le qualità disponibili variano da **-1** (nessuna compressione, solo archiviazione) a **-9** (compressione massima, viene utilizzato uno spazio minimo sul disco).

bzip2 e **bunzip2** servono per comprimere e decomprimere file nel formato `.bz2`. La sintassi dei sopracitati comandi è la seguente:

```
bzip2 [OPZIONE]... [FILE]...
```

Alcuni esempi di uso dei comandi **bzip2** e **bunzip2**:

- `bzip miofile.txt`

  Comprime il file `miofile.txt` creando il file `miofile.txt.bz2`.
- `bunzip2 miofile.bz2`

  Decomprime il file `miofile.bz2`.

zip e **unzip** servono per comprimere e decomprimere file nel formato `.zip`.
La sintassi del comando **zip** è la seguente:

```
zip [-options] [-b path] [-t mmddyyyy]
      [-n suffixes] [zipfile list] [-
      xi list]
```

Alcuni esempi di uso del comando:

- `unzip archivio.zip`

  Decomprime il file `archivio.zip`.

- `zip archivio.zip file.txt img.png documento.doc`

  Crea un file compresso `archivio.zip` contenente i file `file.txt`, `img.png` e `documento.doc`.

- `zip -e esempio.zip file.txt`

  Crea un archivio `esempio.zip` criptato e protetto con una password a scelta dell'utente.

Gestione del filesystem

mount serve per effettuare il *montaggio* di un filesystem all'interno della gerarchia di file del sistema, rendendolo accessibile a partire da una specifica directory chiamata **punto di mount** (o di montaggio).

Alcuni esempi di uso del comando **mount**:

- `mount`

  Visualizza i dispositivi attualmente montati, il loro punto di montaggio e il tipo di filesystem utilizzato per gestirne i dati.

- `mount /media/cdrom`

  Monta in `/media/cdrom` il dispositivo CD-Rom.

- `mount -t ntfs /dev/sda1 /media/dati_windows`

  Monta la partizione identificata come `/dev/sda1` all'interno della directory `/media/dati_windows`, in modo che tutti i dati presenti in questa partizione diventano accessibili a partire dalla directory scelta.

umount serve per *smontare* un dispositivo precedentemente montato.

La sintassi del comando **umount** è la seguente:

```
umount [dispositivo]
```

Alcuni esempi di uso del comando **umount**:

- `umount /media/cdrom`

  Smonta il dispositivo CD-ROM.

Ottenere informazioni sul sistema

du visualizza lo spazio occupato sul disco da file o directory.

La sintassi è la seguente:

```
du [opzioni] [file...]
```

Alcune opzioni da utilizzare con il comando **du**:

Opzione	Risultato
-a	visualizza le informazioni sia sui file che sulle directory
-s	visualizza la dimensione totale complessiva
-x	esclude le sottodirectory che siano parte di un'altro filesystem

Alcuni esempi di uso del comando **du**:

- `du miofile`

  Visualizza la quantità di spazio occupata da `miofile`.

- `du -s ~`

  Visualizza la quantità di spazio complessiva occupata dalla propria directory home.

df visualizza lo spazio rimasto sulle partizioni e sui dischi del proprio sistema.

La sintassi del comando è la seguente:

```
df [opzioni] [file...]
```

Alcune opzioni da utilizzare con il comando **df**:

Opzione	Risultato
-a	include nell'elenco anche i filesystem con una dimensione di 0 blocchi, che sono di natura omessi. Normalmente questi filesystem sono pseudo-filesystem con scopi particolari, come le voci per l' *automounter*. Filesystem di tipo «ignore» o «auto», supportati da alcuni sistemi operativi, sono inclusi solo se quest'opzione è specificata
-h	aggiunge a ciascuna dimensione un suffisso, come «M» per megabyte, «G» per gigabyte, ecc
-H	ha lo stesso effetto di **-h**, ma usa le unità ufficiali SI (con potenze di 1000 piuttosto che di 1024, per cui M sta per 1000000 invece di 1048576)
-t tipofs	limita l'elenco a filesystem del tipo specificato
-x tipofs	limita l'elenco a filesystem *non* del tipo specificato

Un esempio di uso del comando **df**:

- `df -Ht ext3`

  Mostra lo spazio occupato solo dai dischi con filesystem **ext3**, utilizzando il suffisso specifico per l'unità di misura.

free mostra informazioni sulla memoria di sistema. Molto utile se si vuole rendersi conto della memoria disponibile sul sistema, della memoria attualmente in uso e di quella libera.
La sintassi del comando è la seguente:

```
free [opzioni]
```

Alcune opzioni da utilizzare con il comando **free**:

Opzione	Risultato
-b	mostra la quantità di memoria in byte
-k	mostra la quantità di memoria in KB (impostato di default)
-t	mostra una riga contente i totali

top visualizza informazioni riguardanti il proprio sistema, processi in esecuzione e risorse di sistema, utilizzo di CPU, RAM e spazio swap utilizzato e il numero di task in esecuzione.

La sintassi del comando è la seguente:

```
top
```

Per uscire dal programma, premere il tasto «**q**».

uname mostra informazioni sul sistema.
La sintassi è la seguente:

```
uname [opzione]
```

Alcune opzioni da utilizzare con il comando **uname**:

Opzione	Risultato
-a	visualizzerà tutte le informazioni del sistema
-m	mostra il tipo di macchina
-n	mostra il nome host del nodo di rete della macchina
-s	mostra il nome del kernel
-r	mostra la release del kernel
-o	mostra il nome del sistema operativo

lsb_release mostra informazioni sulla distribuzione installata.

La sintassi è la seguente:

```
lsb_release [opzione]
```

Alcune opzioni da utilizzare con il comando **lsb_release**:

Opzione	Risultato
-d	mostra la descrizione della distribuzione
-c	mostra il nome in codice della distribuzione
-r	mostra il numero di rilascio della distribuzione
-a	mostra tutte le informazioni sulla distribuzione

Amministrazione degli utenti

In GNU/Linux c'è un utente particolare, detto **super utente** (utente **root**, UserID 0) che ha totale accesso al sistema senza nessuna restrizione, cioè ne è l'**amministratore**.

sudo (super**u**ser **do**) consente di eseguire un comando come se si fosse un altro utente. Per eseguire dei comandi con privilegi d'amministrazione è sufficiente digitare **sudo** e

successivamente il comando che si desidera eseguire come utente **root**. Ad esempio:

```
sudo passwd nomeutente
```

Una volta digitato il comando, il sistema chiederà la password dell'utente attuale (e non la password dell'utente root) La password viene chiesta la prima volta e memorizzata per un certo lasso di tempo, quindi è possibile usare il comando sudo più volte consecutive senza dover inserire ogni volta la password.

passwd consente di cambiare o impostare la propria password o la password di un utente. Esempio:

```
sudo passwd nomeutente
```

Gestione dei pacchetti

Quando si installa del software, molti altri file possono essere richiesti solo per poterlo avviare ed è necessario che questi file siano posizionati nei corretti percorsi. GNU/Linux fa uso dei pacchetti per archiviare tutto ciò di cui un particolare software necessita per essere eseguito. Un pacchetto, quindi, è un insieme di file raccolti in un unico file che comprende, oltre ai file necessari all'esecuzione del software, dei file speciali chiamati script di installazione che copiano i file nelle posizioni richieste. **APT** (Advanced Package Tool) è il sistema di gestione dei pacchetti predefinito in Ubuntu, Debian e in altre distribuzioni.

apt-get installa e rimuove i pacchetti e permette di aggiornare il sistema. Esempio:

```
sudo apt-get install nomepacchetto
```

Installa il pacchetto **nomepacchetto**. Oltre ad install è possibile specificare altri comandi:

Comandi per installare o rimuovere pacchetti	
install *packagename*	Installa un nuovo pacchetto.
remove *packagename*	Rimuove un pacchetto.
--purge remove *packagename*	Rimuove un pacchetto, compresi tutti i file di configurazione.
autoremove *packagename*	Rimuove un pacchetto e tutte le dipendendenze inutilizzate.
-f install	Tenta di di riparare i pacchetti con delle dipendenze non soddisfatte.

Consente di impostare la password dell'utente **nomeutente**. Il seguente comando invece consente di cambiare la propria password:

```
passwd
```

useradd consente di aggiungere nuovi utenti al sistema. Esempio:

```
sudo useradd nuovoutente
```

Crea un nuovo utente chiamato **nuovoutente**.

chown consente di cambiare l'utente proprietario di un file . Esempio:

```
sudo chown nomeutente nomefile
```

Imposta l'utente **nomeutente** come proprietario del file **nomefile**.

Comandi per aggiornare il sistema	
update	Aggiorna la lista dei pacchetti disponibili dai repository. Va lanciato dopo aver apportato delle modifiche a `/etc/apt/sources.list` o a `/etc/apt/preferences`.
upgrade	Scarica e installa gli aggiornamenti per tutti i pacchetti installati
dist-upgrade	Aggiorna l'intero sistema ad una nuova versione. Delega APT a svolgere tutti i compiti necessari all'aggiornamento dell'intera distribuzione, anche l'eventuale cancellazione di pacchetti.

apt-cache ricerca dei pacchetti all'interno del database e le informazioni riguardanti il contenuto di essi. Esempio:

```
sudo apt-cache search stringa
```

Cerca **stringa** nella lista dei pacchetti. Oltre a search è possibile specificare altri comandi:

Comandi	
search *string*	Cerca una stringa nella lista dei pacchetti conosciuti.
showpkg *pkgs*	Mostra alcune informazioni riguardo ai pacchetti.
dumpavail	Stampa una lista di tutti i pacchetti disponibili.
show *pkgs*	Visualizza tutte le informazioni riguardo un pacchetto, similmente a **dpkg --print-avail**
depends *pkgs*	Visualizza le dipendenze e i conflitti dei pacchetti selezionati.

policy *pkgs*	Nel caso siano disponibili numerose versioni dei pacchetti selezionati visualizza la versione attualmente installata e il repository di origine.
pkgnames	Lista veloce di ogni pacchetto del sistema.

apt-file cerca un pacchetto, anche non installato.

Ad esempio:

```
sudo apt-file search nomefile
```

Cerca un pacchetto che contenga il file **nomefile**.

Apt-file è un pacchetto indipendente, e deve essere installato tramite il comando apt-get. Una volta installato, è utile aggiornare il database dei pacchetti con il comando **apt-file update**.

Altri comandi utili

cat e **less** servono per mostrare il contenuto di un file:

- **cat** mostra semplicemente il contenuto del file specificato;
- **less** visualizza il contenuto di file, permette di spostarsi avanti e indietro nel testo utilizzando i tasti freccia quando i file occupano più di una pagina di schermo. È inoltre possibile eseguire delle ricerche nel testo digitando «/» seguito dalla parola da cercare e premendo «**Invio**».

Per terminare il programma premere il tasto «**q**». La sintassi del comando **cat** è la seguente:

```
cat nomefile
```

La sintassi del comando **less** è la seguente:

```
less nomefile
```

more viene solitamente utilizzato in abbinamento ad altri comandi. È un filtro che permette di visualizzare l'output di un comando una schermata alla volta.
Alcuni esempi d'uso del comando **more** abbinato ad altri comandi:

- `ls | more`
- `cat miofile | more`

Il simbolo «|», solitamente chiamato *pipe*, serve per redirigere l'output del comando a sinistra al comando alla sua destra.

vi o **vim** (**vi**sual editor i**m**proved) permette di modificare un file di testo. Ad esempio:

```
vi nomefile
```

Se **nomefile** esiste viene aperto in modifica, se non esiste viene creato un nuovo file dai nome **nomefile**. Esistono due modalità di funzionamento: modo comando (**command**) e modo inserimento (**input**). In modo inserimento ogni parola verrà inserita direttamente nel file, per entrarci è possibile usare il comando **i** (insert). Per entrare in modo comando è possibile in qualsiasi momento premere il tasto **ESC**, ogni lettera verrà interpretata come un comando:

Opzione	Risultato
:w	Salva il file
:wq	Salva il file ed esce
:q	Se non sono state effettuate modifiche esce senza salvare
:q!	Esce incondizionatamente.
yy	Copia la riga corrente
dd	Cancella la riga corrente

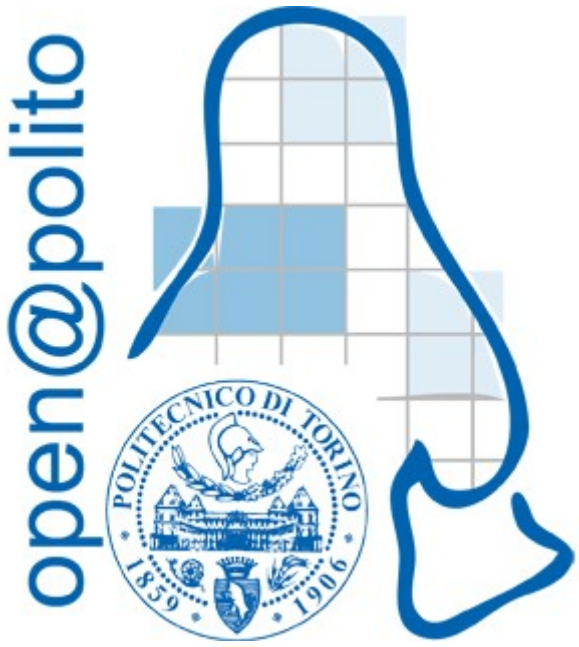
Ulteriori risorse

- Documentazione della comunità italiana di Ubuntu (it): http://wiki.ubuntu-it.org
- Italian Linux Documentation Project (it): http://ww.pluto.it/ildp
- Tutorial online con ulteriori esempi (en): http://linuxcommand.org
- Versione HTML delle pagine man (en): http://man.cx
- Corso Linux Amministrazione Base (it): http://www.coresis.com/extra/linuxcorsobase/programma.htm

---

Copyleft

Quest'opera, è stata realizzata grazie al contributo di molte persone. La prima versione è stata estratta da una guida realizzata dalla comunità italiana di Ubuntu distribuita con licenza Creative Commons all'url: http://wiki.ubuntu-it.org/AmministrazioneSistema/ComandiBase. Successivamente sono state apportate delle modifiche da chi ha collaborato a vario titolo alla realizzazione delle lezioni dei corsi GNU/Linux gratuiti tenuti al Politecnico di Torino. In ordine sparso (e sperando di non dimenticare nessuno): Giovanni Berton Giachetti, Daniele Lussana, Alessandro Ugo, Emmanuel Richiardone, Andrea Garzena, Stefano Cotta Ramusino, Roberto Preziusi, Marco Papa Manzillo, Puria Nafisi Azizi, Luca Necchi, Luca Barbato, David Putzer, Alberto Grimaldi, Nicola Tuveri, Stefano Colazzo, ecc.



Questo prontuario è stato realizzato e stampato grazie alla collaborazione del:

Centro di Competenza per l'open source e il software libero

http://open.polito.it

linux@studenti

Esiste grazie all'attività per lo più volontaria degli studenti del Politecnico di Torino.

http://linux.studenti.polito.it