Guida ai comandi da terminale - Gestione di file e directory

Da Guide@Debianizzati.Org.

Indice

Muoversi tra le directory

Accedere da terminale ad una directory è semplicissimo, perché il comando è uguale a quello del DOS.
Per entrare nella directory “cartella” digitiamo:

```
cd cartella
```

se ci troviamo nella directory "genitore" della directory "cartella". Oppure digitiamo:

```
cd /percorso/assoluto/per/arrivare/a/cartella
```

Si noti che per utilizzare il percorso assoluto dobbiamo per prima cosa inserire il / che sta ad indicare la radice del filesystem. Per entrare nella directory “sottocartella” presente in “cartella” con un solo comando, digitiamo:

```
cd cartella/sottocartella
```

Se siamo in “sottocartella” e vogliamo tornare in “cartella” digitiamo:

```
cd ..
```

Se siamo in “sottocartella” e vogliamo uscire sia da lì che da “cartella”, digitiamo:

```
cd ../..
```

Visualizzare il contenuto di una directory

Se avete bisogno di vedere cosa è presente in una cartella, tutto quello che dovrete fare sarà entrare nella cartella che volete esaminare e lanciare:

```
ls
```

Esistono una serie di opzioni da aggiungere per perfezionare la ricerca. Avrete una guida completa di quello che potete aggiungere digitando:

```
ls --help
```

Le opzioni più utili sono secondo me:

```
ls -a
```

con cui si potranno visualizzare anche gli eventuali file nascosti, compresi `.` e `..` (directory corrente e directory superiore)

```
ls -A
```

con cui si potranno visualizzare anche gli eventuali file nascosti

```
ls -la
```

con cui si potranno visualizzare i file nascosti (`a`) e produrre un listato (`l`) con diversi dettagli utili (permessi di lettura/scrittura/esecuzione, utente e gruppo proprietari, dimensione, data di ultima modifica, ecc...)

Creare una directory

Per creare una directory da terminale, dobbiamo spostarci dentro la directory nella quale desideriamo creare la nuova directory, e digitare:

```
mkdir nuovacartella
```

Se volessimo creare in un colpo solo la directory “cartella” e subito al suo interno la directory “sottocartella” scriviamo:

```
mkdir cartella/sottocartella
```

Rinominare una directory

Rinominare una cartella da terminale è semplice. Se ad esempio dobbiamo assegnare alla directory “cartella” il nuovo nome “directory”, dobbiamo portarci nella directory genitore e scrivere:

```
mv cartella/ directory/
```

Spostare (tagliare ed incollare) una directory

Spostare (tagliare ed incollare) una directory da terminale richiede un solo semplice comando. Supponiamo di avere la seguente struttura:

```
/home
|
|__ ferdy
 |
 |__ cartella
 | |__ sottocartella
 |
 |__ directory
```

e di voler spostare la directory `sottocartella` dalla directory `cartella` alla directory `directory`:

```
cd /home/ferdy/cartella
mv sottocartella /home/ferdy/directory
```

oppure, con un solo comando:

```
mv /home/ferdy/cartella/sottocartella /home/ferdy/directory
```

Copiare una directory

Copiare ed incollare una directory da terminale richiede un solo semplice comando. Supponiamo di avere la seguente struttura:

```
/home
|
|__ ferdy
 |
 |__ cartella
 | |__ sottocartella
 |
 |__ directory
```

e di voler copiare la directory `sottocartella` dalla directory `cartella` alla directory `directory`:

```
cd /home/ferdy/cartella
cp -r sottocartella /home/ferdy/directory
```

oppure, con un solo comando:

```
cp -r /home/ferdy/cartella/sottocartella /home/ferdy/directory
```

L'opzione `-r` permette di estendere ricorsivamente l'opzione di copia alle eventuali sottodirectory contenute della directory "sottocartella".

Cancellare una directory vuota

Volendo cancellare la directory “cartella”, dobbiamo scrivere:

```
rmdir cartella
```

Questo comando presuppone che la directory sia vuota. In caso contrario verrà restituito l’errore:

```
rmdir: failed to remove `cartella’: La directory non è vuota
```

Cancellare una directory non vuota

Volendo cancellare la directory non vuota “cartella” dovremo digitare:

```
rm -r cartella
```

L'opzione `-r` è in grado di cancellare ricorsivamente ogni file o sottodirectory di "cartella". Prestate attenzione a come utilizzate il comando perché **non viene chiesta alcuna conferma** e rischiate di eliminare file o directory di sistema.
Il comando precedente funziona correttamente anche se si vogliono cancellare directory vuote.

Visualizzare la struttura di una directory

`tree` è un comando shell che ha il compito di listare ad albero una data directory. La sintassi del comando è:

```
tree [ -adfgilnopqrstuxACDFNS ][ -L level [-R] ][ -H baseHREF ][ -T title ][ -o filename ][ --nolinks ][ -P pattern ]
 [ -I pattern ][ --inodes ][--device ][ --nore-port ][ --dirsfirst ][ --version ][ --help ][ directory ... ]
```

Alcune delle numerose opzioni che possiamo passare al comando tree sono:

- `-a`: Stampa a video tutti i file, compresi quelli nascosti
- `-d`: Lista soltanto le directory
- `-f`: Stampa tutto il percorso di ogni file
- `-i`: Non stampa un output ad 'albero'
- `-s`: Stampa affianco al nome del file la sua dimensione
- `-r`: Riordina alfabeticamente l'output

Se vogliamo ad esempio listare la cartella `/home/user/tmp`, basta dare il comando:

```
tree /home/user/tmp
```

Rinominare un file

Per rinominare un file da terminale deve essere usato il comando `mv`. Volendo rinominare il file “testo.txt” in “text.txt” dovremo dirigerci prima nella directory in cui il file è presente, e poi digitare:

```
mv testo.txt text.txt
```

Spostare (tagliare ed incollare) un file

Anche per questa operazione deve essere usato il comando `mv`. Supponiamo di avere il file "testo.txt" in una specifica directory e di volerlo spostare nella nostra directory Home. Dirigiamoci da terminale nella directory in cui il file è presente, e lanciamo:

```
mv testo.txt /home/vostronomeutente/
```

Copiare un file

Se intendiamo duplicare un file, dobbiamo usare il comando `cp`. Con questo comando si possono fare più cose:

- copiare un file da una directory ad un'altra:

```
cp testo.txt /home/vostronomeutente/testo.txt
```

- creare un duplicato del file nella stessa direcctory, cambiandogli solo il nome:

```
cp testo.txt text.txt
```

- creare un duplicato e cambiare anche l'estensione:

```
cp testo.txt testo.bak
```

Cancellare un file

La cancellazione di un file da terminale è semplice, basterà digitare il comando `rm` seguito dal nome del file che si vuole cancellare. Ad esempio:

```
rm testo.txt
```

Cancellare definitivamente un file

`shred` è un comando shell che sovrascrive ripetutamente i FILE specificati in modo da rendere più difficile recuperare i dati, e opzionalmente li cancella. La sintassi del comando è:

```
shred [OPZIONI] FILE [...]
```

Le opzioni che possiamo passare al comando sono:

- `-f, --force`: change permissions to allow writing if necessary
- `-n, --iterations=N`: Overwrite N times instead of the default (25)
- `--random-source=FILE`: get random bytes from FILE (default /dev/urandom)
- `-s, --size=N`: shred this many bytes (suffixes like K, M, G accepted)
- `-u, --remove`: truncate and remove file after overwriting
- `-v, --verbose`: show progress
- `-x, --exact`: do not round file sizes up to the next full block; this is the default for non-regular files
- `-z, --zero`: add a final overwrite with zeros to hide shredding

Visualizzare il contenuto di un file

Il comando `cat` (che analizzeremo nel dettaglio tra pochi paragrafi) visualizza i contenuti di un intero file sullo schermo. Ad esempio, digitate:

```
cat filename.txt
```

per vedere il contenuto del file. Se il file è piuttosto lungo, il suo contenuto scorrerà velocemente sullo schermo. Per evitare ciò, usate un pager come less o most (more scorre solo in avanti):

```
less filename.txt
```

Confrontare due file

`diff` è un comando shell che trova la differenza tra due file. La sintassi del comando è:

```
diff [opzioni] da-file verso-file
```

Alcune delle opzioni che possiamo passare al comando sono:

- `-righe`: Mostra "righe" (un intero) righe di contesto. Questa opzione non specifica da sola un formato di output; non ha nessun effetto a meno che venga combinata con `-c` o `-u`. Questa opzione è obsoleta. Per operare correttamente, diff ha, di solito, bisogno di almeno due righe di contesto.
- `-a`: Tratta tutti i file come testo confrontandoli riga per riga, anche se non sembrano essere testo.
- `-b`: Ignora differenza nella quantità di spazi bianchi.
- `-B`: Ignora differenze che consistono solo in righe vuote (eccedenti o mancanti).
- `--brief`: Riferisce solo se i file sono diversi, e non i particolari della differenza.
- `-c`: Usa il formato «a contesto» (context output, in inglese).
- `-d`: Usa un algoritmo diverso per trovare un, probabilmente, minore insieme di modifiche da fare. Questo rende diff più lento (a volte molto più lento).

Comprimere file e directory

La creazione di un nuovo file con il comando `tar` viene fatta mediante l'opzione `-c` indicando il nome del file da creare e il file o la directory originali: `$ tar cf nuovo.tar dir/` Il file creato contiene l'intera directory `dir/`. Per estrarre la directory dall'archivio utilizziamo l'opzione `-x`: `$ tar xf nuovo.tar` l'archivio verrà estratto interamente nella directory corrente.
L'opzione `-f` utilizzata fino ad ora serve ad indicare che il campo direttamente successivo sarà il nome dell'archivio.
Tuttavia in questo esempio non abbiamo utilizzato alcun software di compressione per ridurre la dimensione dei dati nell'archivio. Per farlo possiamo scegliere se utilizzare il software di compressione *gzip* (opzione -z), *bzip2* (opzione -j) o *xz* (opzione -J):

```
$ tar czf nuovo.tar.gz dir/ # utilizza gzip
$ tar cjf nuovo.tar.bz2 dir/ # utilizza bzip2
$ tar cJf nuovo.tar.xz dir/ # utilizza xz
```

Analogamente per decomprimere:

```
$ tar xzf archive.tar.gz
$ tar xjf archive.tar.bz2
$ tar xJf archive.tar.xz
```

L'estensione dei due file è stata cambiata da *.tar* a *.tar.gz*, *.tar.bz2* e *.tar.xz* per indicare chiaramente con quale tipo di algoritmo sono stati compressi i dati.
Esistono versioni compatte delle stesse estensioni: *.tgz* equivale a *.tar.gz*, *.tbz* equivale a *.tar.bz2* e *.txz* equivale a *.tar.xz*. Si può anche saltare l'algoritmo da usare per la decompressione, usando semplicemente:

```
$ tar xf archive.tar.*
```

Nota: l'estensione corretta in realtà non è assolutamente necessaria. Possiamo chiamare tutti gli archivi .tar indipendentemente dalla compressione e questi funzioneranno perfettamente. Tuttavia in questo modo l'utente non ha modo di conoscere a colpo d'occhio l'algoritmo utilizzato.
Una opzione basilare interessante è `-t` con la quale possiamo stampare il contenuto di un file compresso. Ad esempio:

```
$ tar tzf archive.tar.gz
```

Nel caso in cui l'archivio non sia stato compresso utilizzando *gzip*, *bzip2* o *xz* è possibile aggiungervi singoli file o intere directory utilizzando la modalità append

attivata dall'opzione `-r`:

```
$ tar rf archive.tar file dir/
```

Nota: è importante che l'archivio non sia compresso. Nel caso lo sia è possibile prima eliminare la compressione dall'archivio e poi aggiungere dei file. Alla fine verrà ricompresso il tutto.

Esempi

- Comprimere tutte le immagini jpg dentro una directory:

```
$ tar cvzf photo.tar.gz photo/*.jpg
```

- Come sopra ma il file viene inserito nella directory dove le foto risiedono:

```
$ tar cvzf photo/photo.tar.gz photo/*.jpg
```

- Comprimere la directory corrente (1):

```
$ tar cvjf current.tar.bz2 .
```

- Comprimere la directory corrente (2):

```
$ tar cvjf current.tar.bz2 *
```

Nota: La differenza tra i due comandi precedenti è abbastanza semplice. Nel primo caso ad essere compressa è la reale direcotry corrente. Scompattando l'archivio ci troveremo con una directory che contiene tutti i file. Nel secondo caso invece ad essere compressa non è la directory bensì tutti i file che essa contiene. Scompattando non troveremo una cartella contenente ma tutti i file contenuti nell'archivio saranno copiati nella directory di scompattazione senza che sia creata la subdirectory.

- Copiare una directory:

```
$ tar cf - /some/directory | (cd /another/directory && tar xf -)
```

- Estrarre tutti i file .h da un archivio:

```
$ tar xvzf source.tar.gz *.h
```

- Cerca tutti i file .jpeg nella home e crea un archivio:

```
$ find ~ -type f -name "*.jpg" | xargs tar cvzf photo.tar.gz
```

- Se stiamo cercando di creare un archivio molto grande è più conveniente il seguente metodo:

```
$ find ~ -type f -name "*.jpg" | xargs tar rvzf photo.tar.gz
```

nel quale abbiamo usato l'opzione append (-r) invece della create (-c). Questo perchè nel caso di archivi molto grandi xargs passa tutti gli argomenti a blocchi e tar ricrea ogni volta l'archivio con solamente i file elencati in quel blocco all'interno. La modalità append risolve il problema.

- Estrarre singoli file da un archivio:

```
$ tar xvjf archive.tar.bz2 file1.cpp file2.cpp file3.cpp
```

- Estrarre file dalle subdirectory di un archivio: o subdirectory intere:

```
$ tar xvjf archive.tar.bz2 subdir1/file1.cpp subdir2/file2.cpp
```

- Estrarre una intera subdirectory da un archivio:

```
$ tar xvjf archive.tar.bz2 subdir1/
```

Nota: Per poter estrarre singoli file o subdirectory occorre conoscerne con esattezza i nomi. Tuttavia è possibile utilizzare altri comandi come find, grep, awk per ottenere un elenco dei file che vogliamo estrarre.

- Estrarre da un archivio solo i file elencati in un file di testo:

```
$ tar xvf archive.tar -T list.txt
```

- Importare in un file di testo tutti gli elementi di un archivio:

```
$ tar tzf archive.tar.gz > list.txt
```

- Eliminare da una directory contenente molti file solo quelli contenuti anche nell'archivio:

```
$ tar tzf archive.tar.gz | xargs rm -r
```

- Stimare la dimensione di un archivio prima di crearlo:

```
$ tar czf - directory/ | wc -c
```

(Grazie a Nerotux (http://nerotux.tuxfamily.org) per gli esempi sull'utilizzo di tar)

Modificare l'ora di accesso a un file

`touch` è un comando shell che modifica l'orario di accesso o/e modifica di un file (il cosiddetto **timestamp**). I file vengono modificati con l'orario corrente. La sintassi del comando è:

```
touch [-acfm] [-r file][-t orario_decimale] [-d orario]
 [--time={atime,access,use,mtime,modify}][--date=orario][--reference=file]
 [--no-create] [--help] [--version] [--] file...
```

Le opzioni che possiamo passare al comando sono:

- `-a`: Cambia l'ora di accesso di file.
- `-c`: Non crea file.
- `-m`: Cambia l'ora di modifica di file.
- `-r ref_file`: Usa i corrispondenti orari di ref_file come i nuovi valori per gli orari da cambiare.
- `-t orario`: Usa l'orario specificato come nuovo valore per gli orari cambiati. L'argomento è un numero decimale nella forma [[SS]AA]MMGGoomm[.ss]
- `-d, --date=orario`: Usa orario al posto dell'ora corrente. Può contenere il nomi di mese, fusi orari, «am» e «pm», ecc.

Manipolazione di file

Un modo veloce (soprattutto negli script) per modificare il contenuto di un file o per accodare altro testo alla sua fine è utilizzare il comando `cat`.
`cat` è un comando shell che legge file in sequenza e può scrivere i loro contenuti sullo standard output nella stessa sequenza. Lo scopo di cat è infatti concatenare file (generalmente di testo o che abbiano particolari intestazioni, ad esempio il formato MPEG2 usato dai VOB nei DVD video). Dimenticatevi follie quali "cat file.txt | less" o "cat file.txt | grep foo" perché è definito UUOC: Useless Use Of Cat (http://en.wikipedia.org /wiki/Cat_%28Unix%29#Useless_use_of_cat) :-)

La sintassi del comando è:

```
cat [opzioni][file ...]
```

Note:

- `> file.estensione`: Se nomefile.estensione non esiste, viene creato e viene scritto il valore al suo interno. Nel caso esistesse già, il suo valore **verrebbe sovrascritto**.

- **>>** `nomefile.estensione`: Aggiunge alla fine del file il valore che gli date a riga di comando.

Altri comandi utili per la manipolazione veloce di file sono:

- `head`, che mostra le prime dieci linee di un file. Utile quando si vuole avere un'idea del contenuto di un file senza doverlo aprire con un editor
- `tail`, che mostra al contrario le ultime dieci linee di un file

Esempi

Se vogliamo mostrare il contenuto del file di testo `/home/ferdy/prova.txt`, dobbiamo dare il comando:

```
cat /home/ferdy/prova.txt
```

Se vogliamo aggiungere il contenuto del file `/home/ferdy/aggiunte.txt` alla fine del file, diamo il seguente comando:

```
cat /home/ferdy/aggiunte.txt >> /home/ferdy/prova.txt
```

Se vogliamo unire i due file e crearne un terzo, diamo il comando:

```
cat /home/ferdy/prova.txt /home/ferdy/aggiunte.txt > /home/ferdy/file_finale.txt
```

Per visualizzare le prime dieci o le ultime dieci linee di un file:

```
head /home/ferdy/prova.txt
tail /home/ferdy/prova.txt
```

Un comando preziosissimo per gli amministratori di sistema è il seguente, che permette di vedere in modo interattivo la variazione di un file di log:

```
tail -f /var/log/syslog | ccze
```

Nell'ultimo comando sono stati usati:

- **`-f`** che permette a `tail` di seguire (follow) il file e i suoi cambiamenti
- **`| ccze`** che dirige l'output di tail verso il programma `ccze`, che colorerà l'output del log rendendolo facilmente leggibile

In quest'ultimo esempio, per rilasciare il terminale e riottenerne il pieno utilizzo bisogna stoppare `tail` con la combinazione di tasti `CTRL`+`C`.

Un altro uso interessante di cat è creare file di testo sfruttando lo stdin della shell. Lanciamo questo comando:

```
cat - > file.txt
```

e possiamo scrivere il nostro testo sfruttando lo standard input come editor, si può correggere solo se non si è andati a capo e si termina con *Ctrl-d*.

Creare un collegamento

Un collegamento è un particolare tipo di file che non è altro che un rimando ad un altro file o directory. Per creare un collegamento tra file o directory si deve usare il comando `ln`, di solito con l'opzione `-s` per creare link simbolici.

```
ln -s <file da collegare> <nome del link>
```

Supponiamo di avere il file testo.txt nella nostra home e di voler creare un collegamento in `usr/bin` chiamato "mylink.txt". Scriverò:

```
ln -s testo.txt mylink.txt
```

Su Linux esistono due tipi di collegamenti: Hard link e Symbolic link.

Hard Link

Sono di fatto una copia di una voce di directory, hanno nomi diversi ma puntano allo stesso inode e quindi condividono esattamente lo stesso dato (oltre agli stessi permessi, data di modifica, owner ecc.). Non c'è modo di distinguere l'originale dal link perché sono la stessa cosa: e per cancellare il file dal file system è necessario eliminare tutti gli hard-link associati allo stesso inode.

Sono permessi soltanto per file regolari, non per directory, e soltanto se i file risiedono sullo stesso file system, se li supporta. Sono supportati da ext2/3/4.

SymLink

Sono dei piccoli file che contengono un puntamento ad altri file regolari o directory. Questi file hanno i-node autonomo e possono puntare a file di altri file system (sia locali, che di rete). Si possono facilmente visualizzare con un normale ls -l e se viene cancellato o spostato il file a cui puntano rimangono "stale": continuano ad esistere ma puntano a qualcosa che non esiste.

Un symlink appare come avente tutti i permessi aperti a tutti gli utenti, di fatto è trasparente rispetto a permessi e ownership e riflette quelli del file o directory a cui punta.

Se il file originale è cancellato, o anche solo spostato, l'esistenza di un link simbolico diviene inutile, non è intercambiabile come per gli hard-link. Infatti mentre l'hard-link punta allo stesso inode, un link simbolico punta al percorso del file originale, e nient'altro.

In genere, per prevenire problemi e salvo si necessiti di conservare una copia dell'inode, è quasi sempre consigliabile ricorrere a link simbolici.

Cambiare i permessi di lettura, scrittura ed esecuzione

Può succedere di aver bisogno di dover cambiare i permessi ad un file, per consentire la lettura, la scrittura o l'esecuzione ad ogni utente, oppure al contrario per proteggerlo da utilizzi indesiderati e restringerlo ad un particolare utente o gruppo.
Il comando da utilizzare per queste operazioni è `chmod`.
I permessi `chmod` sono tipicamente utilizzati dai sistemi operativi UNIX e Linux. Non sono altro che le linee guida che vengono impartite al sistema sugli accessi o non accessi ad una data directory e/o file. Questi sistemi operativi offrono la possibilita di operare con un file facendo la distinzione fra 3 tipi di operatori: Proprietario (User), Gruppo (Group), Pubblici (Others). Per ciascuno di questi utenti é necessario specificare i diversi permessi riguardanti la directory e/o il file in questione. I permessi sono generalmente indicati da alcuni numeri, ovvero:

- Lettura : permetti l'acceso al file (4)
- Scrittura : permetti le modifiche del file da parte di un utente (2)
- Esecuzione : permetti di eseguire lo script (1)

Questi 3 permessi devono essere indicati per ognuno dei 3 tipi di utenti. Ad esempio `chmod 755`, non é altro che il tipo di permesso che si imposta ad un file, rendendolo:

- leggibile-scrivibile-eseguibile al proprietario
- leggibile-eseguibile al gruppo
- leggibile-eseguibile agli altri utenti

La tabella seguente indica il significato dei singoli valori

Valore binario (`rwx`)	**Valore decimale**	**Permessi**
111	7	lettura, scrittura ed esecuzione
110	6	lettura e scrittura
101	5	lettura ed esecuzione

100	4	solo lettura
011	3	scrittura ed esecuzione
010	2	solo scrittura
001	1	solo esecuzione
000	0	nessuno

Il permesso `chmod 755` si calcola in questo modo:

```
Permessi proprietario:
Lettura  sì = 4
Scrittura sì = 2
Esecuzione sì = 1
Totale    = 7
Permessi gruppo:
Lettura  sì = 4
Scrittura no = 0
Esecuzione sì = 1
Totale    = 5
Permessi utenti:
Lettura  sì = 4
Scrittura no = 0
Esecuzione sì = 1
Totale    = 5
```

Affiancando i tre totali si ottiene il permesso 755.
Per cambiare i permessi al file "testo.txt" dovremo quindi agire da autori del file e digitare:

```
chmod xxx testo.txt
```

Al posto delle lettere `xxx` devono essere indicati i numeri ottali visti in precedenza. Ad esempio:

- 755 – Lettura, scrittura, esecuzione per il proprietario, lettura, esecuzione per il gruppo ed altri utenti.
- 644 – Lettura, scrittura per il proprietario, lettura per il gruppo ed altri utenti.
- 666 – Lettura e scrittura per tutti.
- 700 – Lettura, scrittura, esecuzione per il proprietario, nessun permesso per il gruppo ed altri.

In generale valori che vanno da "0" a "7" rappresentano in forma numerica i permessi come di seguito riportato:

```
4 è uguale a lettura(r),
```

```
2 è uguale a scrittura(w),
1 è uguale a esecuzione(x),
0 rimuove i permessi.
```

Il valore numerico risultante è pari alla somma di tali valori.
Partendo sempre da sinistra, la prima cifra si riferisce ai permessi per l'utente proprietario.
La seconda cifra fa riferimento ai permessi per il gruppo proprietario.
La terza e ultima cifra si riferisce ai permessi per gli altri utenti non appartenenti al gruppo proprietario.
Un'altra sintassi utilizzabile è:

```
chmod u|g|o|a|+rwx file1/directory1
```

Significato delle opzioni disponibili del precedente comando: u = user - applica i permessi all'utente proprietario del file, g = group - applica i permessi al gruppo proprietario del file, o = other - applica i permessi ad altri utenti, a = all - applica i permessi a tutti gli utenti, + = questo operatore logico aggiunge i permessi specificati, - = questo operatore logico rimuove i permessi specificati, r = indica il permesso di lettura, w = indica il permesso di scrittura, x = indica il permesso di esecuzione.

Cambiare l'utente proprietario di un file

Per cambiare l'utente e il gruppo proprietari di ciascun file dato (specificato dal primo argomento che non sia un'opzione) si utilizza il comando `chown` nel seguente modo: se viene dato solo un nome utente (o una user ID numerica), quell'utente diventa proprietario di ciascun file dato, il cui gruppo non viene modificato. Se il nome utente è seguito da `:` e un nome di gruppo (o una group ID numerica), senza spazi frapposti, allora anche il gruppo proprietario del file viene cambiato.
La sintassi del comando è:

```
chown [opzioni] utente[:gruppo] file...
```

Per motivi di sicurezza soltanto con privilegi di root è possibile cambiare il proprietario di un file, a prescindere che sia o meno uno dei propri.

Esempi

- Cambiare il proprietario del file `prova.txt`

```
# chown ferdy prova.txt
```

- Cambiare proprietario e gruppo proprietario del file `prova.txt`

```
# chown ferdy:gruppo_desiderato prova.txt
```

- Cambiare ricorsivamente il proprietario di tutti i file della directory `prova`

```
# chown -R ferdy prova/
```

Cambiare il gruppo proprietario di un file

Per cambiare il gruppo proprietario di un file si utilizza il comando `chgrp`. Il gruppo può essere identificato col nome o con l'ID. La sintassi del comando è:

```
chgrp [opzioni] gruppo file...
```

Ad esempio, per cambiare il gruppo proprietario di tutti i file contenuti nella directory `prova` si digiti:

```
chgrp -R gruppo_voluto prova/
```

Per motivi di sicurezza, possono cambiare il gruppo di un file soltanto root oppure un utente che è sia proprietario del file sia membro del gruppo scelto.

Ricerca di file nel sistema

Se siamo alla ricerca di un determinato file all'interno del nostro computer, il mio consiglio è andare su Risorse e selezionare Cerca file. In questo modo il processo sarà più rapido e meno macchinoso, ma in questa guida siamo qui per vedere come è possibile farlo anche da terminale.
Il motore di ricerca da terminale si chiama `locate` e possiamo installarlo normalmente digitando:

```
# apt-get install locate
```

Esso fa uso di un database ad aggiornamenti costanti, ma non frequenti, pertanto se volessimo ricercare un file di recente creazione, spostamento o immissione nel sistema dovremo prima lanciare il comando di update per questo database. Da root o con permessi sudoer aggiungendo "sudo" prima della sintassi digitiamo:

```
# updatedb
```

Attendiamo che l'aggiornamento sia terminato, e poi lanciamo:

```
$ locate testo.txt
```

Scaricare un file da Internet

Da terminale è possibile anche scaricare file da Internet, utilizzando il comando `wget` con i protocolli HTTP o FTP. Ad esempio se volessimo scaricare il file http://www.sito.it/file.rar, ci basterà digitare:

```
wget http://www.sito.it/file.rar
```

Il file sarà messo in download e, una volta scaricato, piazzato nella directory in cui ci trovavamo al momento del lancio del comando.

Sincronizzare il contenuto di due directory

`Rsync` e' un comando (ed un protocollo) che permette la copia di file via rete, ottimizzando i tempi di backup e ripristino dei dati. Non è presente in un'installazione minimale di Debian; occorre pertanto installarlo col comando:

```
# apt-get install rsync
```

La sintassi del comando è la seguente:

```
# rsync -av --delete /Directory/Sorgente /Directory/Destinazione
```

Le opzioni usate hanno il seguente significato:

- `-avr`: specifica di copiare i file mantenendo ownership, permessi e in modalità *directory recursive*
- `--delete`: elimina dalla directory di destinazione i file non più presenti nella directory sorgente

Maggiori informazioni qui.