COMANDI UNIX DI BASE [1]

Questo breve riassunto sui principali comandi UNIX vuole semplicemente essere di aiuto a quanti, armati di buona volontà, vogliono cercare di gestire un po' meglio il loro lavoro e le potenzialità che la macchina offre.

I comandi principali di UNIX sono simili ai comandi DOS, vediamo di passare in rassegna quelli più usati.

Comandi indispensabili in UNIX

Al momento della connessione con UNIX il sistema ci pone due domande: la login e la password. Queste due richieste servono al sistema stesso per identificare l'utente e permettergli di posizionarsi nello spazio a sua disposizione, ovvero dove sono presenti i suoi file e le sue directory.

login	Si deve digitare la propia login. Spesso non è uguale al cognome dell'utente a causa del limitato numero di caratteri che possono essere usati.
password	E' la conferma che dobbiamo dare al sistema per dimostrare che siamo veramente proprietari della login. Ogni utente ha una sua password, deve ricordarla e non deve renderla nota ad altri.
exit	Al termine di ogni sessione di lavoro su UNIX è indispensabile chiudere la connessione con il sistema: il modo corretto è appunto usare il comando **exit** (solo a questo punto si puo' eventualmente usare il comando ATL-X per scaricare il software di rete).

Comandi per la gestione dei file

UNIX	DOS	SCOPO
ls	**dir/w**	Il comando **ls** (list) permette di vedere l'elenco compattato dei file contenuti nella directory corrente.
ls -l	**dir**	Visualizza la lista dei file, le loro dimensioni, il proprietario e la data di creazione.
cp	**copy**	Il comando **cp** (copy) permette di fare una copia del file; la forma più usata è **cp [***filename***] [***newname***]**

[1] Questo compendio relativo ai principali comandi UNIX è stato realizzato nell'aprile del 1993; in questo periodo potrebbero essere state apportate modifiche ad alcuni comandi, che potrebbero avere una diversa sintassi.

mv	**ren**	Questo comando (move) permette di rinominare il file. Es. **mv [***filename***] [***newname***]** . La differenza con il **cp** è notevole: dopo aver copiato (cp) il file ci troviamo ad avere due file uguali con nomi diversi, mentre con mv avremo solo un file con il nuovo nome.
rm	**del**	Il comando **rm** (remove) ci permette la cancellazione di un (o più) file. Es. **rm [***filename***]** permette di cancellare un determinato file; **rm *.doc** permette di cancellare tutti i file che hanno l'estensione .DOC; **rm *** permette di cancellare tutti i file presenti nella directory di lavoro (va usato con attenzione).
cat	**type**	Il comando **cat** (concatenate and print) permette di visualizzare il contenuto di un file; equivale esattamente al comando DOS type.
more	**more**	Visualizza il file una pagina alla volta sullo schermo. Es. **more [***filename***]**

Tutti i comandi destinati a più file possono essere attivati con le wild cards (***** e **?**), analogamente al DOS.

Comandi per la gestione delle directory

UNIX	DOS	SCOPO
mkdir	**md**	Crea un subdirectory dal directory di lavoro. Il comando e' **mkdir [**dirname**]**
rmdir	**rd**	Cancella un subdirectory della directory di lavoro. Analogamente al DOS è necessario che la directory da cancellare sia assolutamente vuota, altrimenti il comando non funziona. La sintassi è **rmdir [***dirname***]**
cd	**cd**	Analogamente al DOS permette di cambiare directory di lavoro. Es. **cd [***dirname***]** permette di spostarsi in una directory immediatamente inferiore a quello di lavoro; **cd ..** permette di spostarsi indietro di una directory; **cd ~** permette di spostarsi nella directory di entrata in UNIX (nella directory loginname), indipendentemente dalla nostra posizione attuale.

Altri comandi UNIX che non hanno riferimenti nel DOS

man	Attiva l'aiuto in linea che l'utente può sfruttare per avere chiarimenti sui comandi da utilizzare, la loro potenzialità e la loro sintassi. Es. **man ls**
pwd	Mostra la directory di lavoro (**p**rint **w**orking **d**irectory).
w	Permette di visualizzare chi è connesso al sistema e che cosa sta facendo (non sempre il nome della login è uguale al nome reale dell'utente).
finger	Mostra le login attive, il nome reale dell'utente e la workstation con la quale è connesso al sistema.
write	Permette di comunicare con un altro utente connesso al sistema. Es. **write [***loginname***]** . A questo punto possiamo digitare una stringa (riga) e, all'atto dell'invio (enter) questa apparirà sul video del destinatario.
talk	Questo comando è infinitamente più evoluto del write: permette di attivare una sessione di colloquio tra due utenti, suddividendo lo schermo in due parti. Il comando è **talk [***loginname***]** ed il destinatario del talk deve rispondere con **talk [***loginname***]** per attivare la connessione. Al termine si interrompe il "dialogo" con il comando CTRL-C.
mesg	Durante la sessione di lavoro su UNIX esiste sempre la possibilità che qualcuno cerchi di comunicare con noi (ad es. con talk): questo comando permette di non essere interrotti da altri utenti o da messaggi e-mail in arrivo. Il comando si divide in due parti: **mesg n** per disabilitare la possibilità di colloquio **mesg y** per riattivare la possibilità di colloquio.
passwd	Questo comando permette di modificare la password di accesso. Digitando semplicemente il comando **passwd** il sistema ci chiederà la conferma della vecchia password, l'inserimento della nuova e la conferma della nuova. E' bene ricordare che la password deve avere una lunghezza minima (sei caratteri) ed è opportuno che contenga almeno un carattere maiuscolo o un numero.

I comandi visti in questo breve compendio sono una piccolissima parte di quelli messi a disposizione da UNIX, ma sono sufficienti per una gestione minima dei file personali e dei propri subdirectory.

Comandi per la gestione dei processi

Il sistema operativo UNIX è “multitasking” e “multi-user”. Questo permette a più utenti di connettersi al sistema e di lavorarci sopra contemporaneamente. Ad ogni “task” il sistema operativo associa un “processo”. Ad esempio, quando viene aperta una sessione di lavoro (login) viene aperto un processo di *shell* che ha il compito di interfacciare il sistema operativo UNIX al terminale collegato.

Nello stesso modo, quando diamo un comando, il sistema operativo apre un processo che ha il compito eseguire il comando dato.

Vediamo brevemente alcuni comandi che permettono la gestione dei processi.

ps	Visualizza tutti i propri processi attivi. Se si usa il comando **ps -a** si ottiene la lista di tutti i processi attivi degli utenti; con **ps -x** si ottiene la lista di tutti i processi attivi dell’utente indipendentemente dal terminale a cui è collegato.
^C	Questo comando (CTRL-C) serve a terminare (interrompere) il processo che abbiamo attivato.
^Z	Con il comando CTRL-Z possiamo “sospendere” un processo (ad es. di text editing) per tornare temporaneamente allo shell ed usare comandi UNIX. La differenza con ^C è notevole: il processo è sospeso, quindi può essere ripreso con un apposito comando; con ^C il processo viene “ucciso” definitivamente.
fg	Con questo comando (foreground) riattiviamo il processo che avevamo sospeso con il comando ^Z. Il processo interrotto (es. text editing) riprenderà esattamente dove era stato interrotto.
bg	Questo comando (background) permette la riattivazione di un processo, ma in background, dissociando cioè il processo dal terminale. E’ necessario puntualizzare che un processo sospeso non è attivo, ad esempio un ricerca di database viene interrotta quando il processo e’ sospeso. Con il comando bg il processo viene riattivato in background, permettendo così di mantenere il prompt di UNIX ed il controllo del terminale.
at	Il comando at permette di eseguire un comando ad una certa ora, in un certo giorno.

La necessità di ridurre la spiegazione di un comando a poche righe rende impossibile una trattazione esauriente, si rammenta quindi che una spiegazione ampia e dettagliata di qualsiasi

comando può essere ottenuta tramite l'aiuto in linea offerto da UNIX. Per consultare l'help in linea offerto da UNIX è sufficiente digitare

man [*commandname***]**

Il comando chmod

Il comando **chmod** non esiste in DOS e necessita di una spiegazione più consistente: ci permette di gestire i file (e le directory) stabilendo le "security", cioè la possibilità che gli altri possano vederli, ci possano scrivere sopra, li possano eseguire o meno. Con questo comando potremmo ad esempio rendere visibili solo alcuni file di comune interesse e proteggere i nostri file personali rendendoli "invisibili".

Digitando il comando **ls -l** si ottiene una lista di questo tipo:

```
total 55
-rwxrwxrwx  1 owner        5412 Mar 26 10:34 filename1
-rwxrwx---  1 owner        3423 Mar 30 09:39 filename2
-rwx------  1 owner       21236 Mar 12 09:12 filename3
-rw-r--r--  1 owner        9856 Feb 21 08:44 filename4
-rwxr-x---  1 owner        3520 Feb 20 09:35 filename5
drwxr-xr-x  1 owner        2048 Mar 12 10:25 directoryname1
drwxr-x---  2 owner        1024 Mar 21 16:12 directoryname2
drwx------  2 owner        1024 Mar 11 12:51 directoryname3
```

In questa lista abbiamo una serie di colonne: la prima a sinistra (-rwxrwxrwx) è l'argomento della nostra spiegazione; la seconda è formata da un numero e dal nome del proprietario dei file (owner); la terza indica le dimensioni dei file, quindi abbiamo la data della loro creazione, l'ora ed il loro nome.

Il primo gruppo di informazioni è formato di dieci caratteri: il primo carattere di sinistra può essere un trattino (-) o una **d**. Questo ci indica che si tratta di un file (-) o di una directory (**d**). I nove caratteri seguenti sono una serie di tre lettere (**rwx**) ripetute per tre volte. Cerchiamo di capire qual'è il loro significato: **r** sta per read, cioè la possibilità di leggere il file; **w** sta per write, cioè la possibilità di scrivere sul file e **x** sta per execute, cioè la possibilità di eseguirlo (qualora lo sia).

Il proprietario (owner) di file e directory può stabilire se alcuni file o directory possono essere lasciati a disposizione degli altri o meno. In pratica la serie di informazioni della prima colonna si puo' riassumere cosi':

- (d)	**r w x**	**r w x**	**r w x**
file (o dir)	possibilità per l'owner	possibilità per il gruppo	possibilità per gli altri

Il file "filename1" (-rwxrwxrwx) è leggibile, scrivibile ed eseguibile da tutti, ovvero chiunque può modificarlo o cancellarlo.

Il file "filename2" (-rwxrwx---) è leggibile, scrivibile ed eseguibile sia dal proprietario (owner) che dal gruppo di appartenenza; gli esterni al gruppo non possono accedervi.

Il file "filename3" (-rwx------) è leggibile, scrivibile ed eseguibile solo dal proprietario (owner), nessun altro può accedervi.

Il file "filename4" (-rw-r--r--) è leggibile e scrivibile dal proprietario, ed è leggibile sia dal gruppo che dagli altri.

Il file "filename5" (-rwxr-x---) è leggibile, scrivibile ed eseguibile dal proprietario, ed è leggibile ed eseguibile dal gruppo.

Lo stesso discorso può essere fatto per le directory: si può permettere l'accesso al gruppo o a tutti, oppure si può negare questo accesso rendendo non visibili i contenuti delle directory che contengono dati importanti o riservati.

Finora abbiamo visualizzato le varie protezioni che possono essere realizzate con le security, cerchiamo ora di capire come si gestiscono, tenendo presente che il file deve essere <u>almeno</u> visibile al proprietario.

Ognuna delle tre lettere che gestiscono le security ha un valore preciso: **r** ha valore **4**; **w** ha valore **2**; **x** ha valore **1**.

Vediamo ora le possibili combinazioni dei valori delle security:

1) Se si imposta un valore=**4** (r--), questo permette solo di leggere il file, non permette di scriverci sopra ne' di eseguirlo.
2) Se si imposta un valore=**6** (rw-), questo permette di leggere e di scrivere il file, non permette di eseguirlo.
3) Se si imposta un valore=**7** (rwx), questo permette di leggere, di scrivere e di eseguire il file.
4) Se si imposta un valore=**5** (r-x), questo permette di leggere e di eseguire il file, non permette di scriverci sopra.

Vediamo ora praticamente i comandi che possono essere utilizzati per modificare le security di file e directory:

- chmod **744 [***filename***]** imposta le protezioni `-rwxr--r--`
- chmod **700 [***filename***]** imposta le protezioni `-rwx------`
- chmod **444 [***filename***]** imposta le protezioni `-r--r--r--`
- chmod **500 [***filename***]** imposta le protezioni `-r-x------`
- chmod **777 [***filename***]** imposta le protezioni `-rwxrwxrwx`
- chmod **640 [***filename***]** imposta le protezioni `-rw-r-----`
- chmod **555 [***filename***]** imposta le protezioni `-r-xr-xr-x`
- chmod **750 [***filename***]** imposta le protezioni `-rwxr-x---`
- chmod **400 [***filename***]** imposta le protezioni `-r--------`

La sintassi esatta del comando è quella sopra descritta: una volta che abbiamo deciso come vogliamo proteggere il file o la directory, digitiamo il comando:

chmod [*nnn***] [***filename***]**

dove **[***nnn***]** è la sequenza di tre numeri vista nell'esempio sopra, e **[***filename***]** (o **[***directoryname***]**) è la destinazione del comando.

E' ovvio che, per quanto si possa scrivere sull'argomento, la cosa più importante è fare dei tentativi per vedere le security in funzione: ad esempio se abbiamo un file importante possiamo proteggerlo con 400, in modo di non poterci scrivere sopra accidentalmente, se volessimo modificarlo dovremmo prima cambiare le security con un altro chmod (600).

CREAZIONE DI FILE - IL TEXT EDITOR "vi"[1]

Il **vi** è un editor di testi in UNIX che permette una gestione nettamente superiore ad altri editor di base ("**e**" ed "**ed**") ed ha due modalità di funzionamento: la modalità comando e la modalità inserimento.

Per lanciare il **vi** come text editor è sufficiente usare il comando:

vi [*filename***]**

A questo punto parte il **vi** in modalitè comando. Per iniziare a digitare dobbiamo premere il tasto "**i**", in questo modo passiamo alla modalità inserimento e possiamo digitare il testo.

Ogni qualvolta siano necessarie modifiche, cancellazioni od altro, è necessario uscire dalla modalità inserimento con la pressione del tasto **ESC**ape.

A questo punto possiamo spostarci sul carattere che necessita della correzione: il problema dello spostamento nel testo potrebbe verificarsi nel caso in cui il terminale non sia correttamente configurato. Per spostare il cursore possiamo usare le quattro frecce che troviamo a destra sulla tastiera; qualora queste non funzionino, dobbiamo servirci dei comandi del "vi". Quindi, una volta premuto il tasto **ESC**ape, digitiamo:

j per spostarci verso il basso

k per spostarci verso l'alto

l per spostarci verso destra

h per spostarci verso sinistra

Ci posizioniamo nel punto in cui i caratteri (o la frase) sono scorretti ed operiamo la cancellazione. I modi principali per cancellare del testo sono due:

x cancella il carattere sopra il quale è posizionato il cursore,

dd cancella la linea corrente.

Una volta terminata la sessione di lavoro è necessario salvare il file ed uscire dal text editor: il primo passo è uscire dalla modalità inserimento con il comando ESC, quindi si digitano i due punti "**:**", seguiti dai caratteri "**w**" (write), "**q**" (quit) e dal punto esclamativo "**!**", originando sul video la stringa

:wq!

In questo modo informiamo il **vi** che vogliamo scrivere tutte le modifiche apportate a video al file e che vogliamo abbandonare il text editor.

In realtà questi comandi sono una piccolissima parte di quelli messi a disposizione da **vi**, ora elencheremo altri comandi (non tutti) che possono essere usati e spiegheremo la loro funzione. Sarà

[1] Anche questo compendio relativo ai principali comandi del "vi" e' stato realizzato nell'aprile del 1993. A differenza dei comandi UNIX, questi comandi sono perfettamente funzionanti.

l'utente a verificarne la praticità e valutare la loro facilità di utilizzo. Tutti i comandi che seguono devono essere usati nella modalità comando, cioè dopo aver premuto il tasto ESCape.

COMANDI

i	Attiva la modalità inserimento (insert), quindi dopo aver digitato **i**, si inizia il testo vero e proprio. Si comincia a scrivere nel punto in cui ci si posiziona.
I	Questo comando porta il cursore all'inizio della riga e attiva la modalità inserimento.
a	Attiva la modalità inserimento (append) ma non inserisce il testo dove si trova il cursore, bensì nella posizione seguente.
A	Attiva la modalità inserimento al termine della riga corrente (append).
o	Permette l'apertura di una nuova linea al di sotto della nostra posizione ed entra in modalità inserimento.
cc	Questo comando cancella la riga corrente e attiva la modalità inserimento.
s	Cancella il carattere sotto il quale si trova il cursore ed entra in modalità inserimento.
x	Cancella il carattere sotto il quale ci siamo posizionati con il cursore. Qualora non si voglia cancellare un solo carattere ma più caratteri si deve digitare il numero seguito dalla "x". Es. vogliamo cancellare la parola "relazione" ci posizioniamo sotto la lettera r, quindi digitiamo 9x : tutti i caratteri vengono cancellati. Questo comando cancella solo dalla linea corrente.
X	Cancella il carattere precedente il cursore.
dd	Cancella la riga sulla quale ci siamo posizionati. Qualora volessimo cancellare più righe, potremmo precedere il comando "dd" dal numero di righe da cancellare. Es. se vogliamo cancellare nove righe (compresa quella sulla quale ci troviamo) digitiamo "9dd". Quando si esegue una cancellazione il testo cancellato viene inserito in un buffer di memoria dove rimane fino alla prossima cancellazione; da questo buffer il testo può essere recuperato con il comando "p" (put). Questo vale soprattutto se abbiamo cancellato accidentalmente del testo.
d$	Questo comando cancella tutti i caratteri seguenti il cursore fino al termine della riga.

$	Sposta il cursore al termine della riga corrente.
0	Sposta il cursore all'inizio della riga corrente.
y$	Copia la linea di testo corrente dalla posizione del cursore alla fine.
yy	Copia la linea di testo corrente in un tampone di memoria. Qualora si vogliano copiare più righe, si deve indicare il numero di righe, a partire da quella corrente, che devono essere copiate. Ad es. se vogliamo copiare 6 linee digitiamo "6yy". A questo punto sull'ultima riga dello schermo apparirà la scritta "6 lines yanked", che indica appunto la messa in memoria delle sei linee scelte.
p	Questo comando (put) permette, una volta posizionati nel punto in cui devono essere inserite, di "incollare" le righe messe in memoria con il comando **yy**.
r	Permette di sostituire il carattere sotto il quale ci siamo posizionati. Il comando esatto è replace ed è valido per un solo carattere alla volta.
u	Questo comando permette di annullare l'ultima modifica realizzata (undo).
U	Questo comando permette di annullare l'ultima serie di modifiche effettuate (undo).

Nei comandi che seguono l'accento circonflesso indica la pressione contemporanea del tasto **CTRL** seguito dalla lettera specificata accanto.

^b	Permette di spostarsi lungo il testo mostrando le 24 righe precedenti (la finestra precedente).
^f	Permette di spostarsi lungo il testo mostrando le 24 righe seguenti (la finestra seguente).
^M	Sposta il cursore al primo carattere della riga seguente.
{	Questo comando sposta il cursore all'inizio del paragrafo precedente. Per "vi" un paragrafo è separato dal seguente da una linea vuota.
}	Questo comando sposta il cursore all'inizio del paragrafo seguente.
H	Permette lo spostamento del cursore al primo carattere dello schermo.
M	Sposta il cursore al centro dello schermo, al primo carattere che risulta in bianco (riga vuota).
L	Sposta il cursore al primo carattere dell'ultima riga.

c$	Questo comando permette la sovrascrittura del testo dal punto in cui è posizionato il cursore fino al termine della riga. Una volta terminata la riga già presente siamo in modalità inserimento, possiamo quindi continuare a digitare il testo.
G	Questo comando permette lo spostamento rapido lungo il testo. Se digitiamo semplicemente "G", andiamo a posizionarci alla fine del documento. Qualora volessimo spostarci in una certa riga dovremmo precedere la G con il numero di riga. Ad es. per portarci alla riga 35 digiteremo 35G . Se invece volessimo spostarci all'inizio del documento sarebbe sufficiente digitare 1G
^g	Visualizza il nome del file, il numero di riga sulla quale siamo posizionati, il numero di righe righe totali del testo e la posizione del cursore rispetto alla globalità del testo in percentuale.
:susp	Questo comando permette di sospendere la sessione di scrittura in "vi". Per attivarlo si preme ESC (per uscire dalla modalità inserimento), si digitano i due punti ":" e "susp", originando la stringa **:susp** . A questo punto possiamo lavorare in unix; quando vogliamo ritornare al nostro testo dobbiamo digitare (al prompt unix (cribi % o altro)) **fg** .
:w	Nel caso di un documento molto lungo è preferibile salvare anche durante la sessione, per non ritrovarsi alla fine e, commettendo un'operazione sbagliata, perdere tutto il testo digitato. In qualsiasi momento possiamo premere il tasto ESCape e digitare la sequenza due punti ":" e la lettera "w" originando la stringa **:w** In questo modo registriamo sul disco il file elaborato fino a quel momento. Dopo aver dato questo comando il cursore si riposiziona dove era prima del comando.
:q!	Questo comando comunica al "vi" che intendiamo interrompere la sessione di lavoro senza salvare il documento editato. E' ovvio che con questo tipo di comando perdiamo tutto il testo editato dopo l'ultimo salvataggio, ovvero TUTTO IL TESTO qualora sia un nuovo file.
:r	Questo comando permette di "caricare" in vi un file già editato (magari in dos e trasferito con ftp). Il comando da dare è **:r [***filename***]** oppure **:r [***path***][***filename***]**

Quasi tutti i comandi riportati possono essere realizzati anche usando altre lettere, si è ritenuto opportuno sorvolare sulla loro spiegazione per non creare ulteriore confusione nell'utenza.

N.B. Ci sono utenti che, abitualmente, non vanno a capo a fine riga, ma lasciano che sia il “vi” a gestire gli “a capo”. Questo potrebbe creare problemi alla ricezione del nostro messaggio, soprattutto perchè UNIX gestisce stringhe lunghissime di caratteri che, all’arrivo, potrebbero essere visibili solo in parte. Si raccomanda perciò di andare regolarmente a capo al termine di ogni riga.