Corso Di Linux

Liceo Fermi, Bologna - Marcello Galli, novembre 2009

La shell

La shell e' l'interfaccia di base per gli utenti dei sistemi Unix (e Linux). La shell e' un'interfaccia a comandi testuali: un programma che legge i comandi scritti su una finestra del video, e li esegue. E' un po' come il vecchio DOS (per chi lo ricorda), ma e' molto piu' potente, infatti e' un vero proprio linguaggio di programmazione. Un file che contiene un programma, ed e' contrassegnato come eseguibile, viene eseguito dalla shell, se si scrive il suo nome sul terminale.

La shell era l'interfaccia che si usava quando la grafica non esisteva ancora; adesso molte cose si fanno con interfacce piu' comode, usando il mouse; ma ci sono ancora situazioni in cui la shell e' indispensabile: per lavorare su computer via rete, (per usare un'interfaccia grafica via rete ci vogliono collegamenti attorno ai 5-10 Mbit/sec) o su dei server, che non hanno la grafica. Oppure serve per venire a capo di problemi del computer, quando l'interfaccia grafica non funziona. Inoltre il funzionamento di un sistema Unix dipende in modo critico dalla shell, dato che il sistema al suo interno usa procedure di shell per fare tante cose e a volte occorre darci un'occhiata.

Ma usare la shell serve sopratutto per capire come funziona il sistema; le interfacce grafiche, per loro natura, hanno un numero limitato di opzioni e nascondono il sistema all'utente, che non sa mai cosa sta succedendo dietro le quinte ed e' perso appena qualcosa non funziona o se vuole fare qualcosa di particolare, non previsto dalle procedure grafiche.

Quindi, se si vuole capire un sistema Linux, bisogna imparare la shell; poi si usa l'interfaccia preferita, ma si sa cosa si sta facendo e non si e' piu' schiavi dell'interfaccia, ma padroni del proprio computer.

Ci sono diversi tipi di shell, circa una decina, come al solito su una macchina linux si possono usare tutte, anche contemporaneamente, ma noi vedremo solo la shell di nome "bash", quella piu' usata.

I comandi che si danno alla shell hano tutti sintassi simile: il comando vero e proprio, in genere una abbreviazione impronunciabile, seguito, sulla stessa linea, da parametri che gli dicono in dettaglio cosa fare; i parametri sono preceduti da un segno meno:"-", poi ci sono, sempre di seguito, degli argomenti, che in genere sono files su cui si opera od altre specifiche.

Ad esempio per fare una lista di files si scrive:
ls
per fare una lista piu' dettagliata:
ls -l
per fare la lista dettagliata del file di nome "pollicino":
ls -l pollicino

Se uno si dimentica i paramtri o gli argomenti, cosa che succede sempre, li chiede al computer. C'e' un comando che da informazini sui comandi e si chiama: "man", che sta per manual (ovvero manuale).

Ad esempio per avere informazioni sul comando ls si scrive:
man ls

Usare questi comandi puo' sembrare un po' ostico, ma in realta' se ne usano pochi, sempre gli stessi, e la cosa migliore e' scriverli su un fogliettino da tenere vicino al computer. Quando si ha bisogno di sapere dei dettagli si usa il comando di aiuto: "man".

Ci sono alcune cose comode della shell, che conviene sapere:

- i tasti freccia servono a richiamare i comandi gia' dati, per cui non occorre riscriverli se li usate spesso;
- il tasto "tab" sinistro serve a completare un conamdo o nome di file che non avete scritto interamente; ne scrivete l'inizio, premete tab e vedete se il computer indovina il resto. Completa anche i nomi dei files.

Infine un suggerimento sull'uso del mouse: il copia ed incolla (cut and paste) in Linux si puo' fare senza tastiera, solo col mouse. Si usa il tasto sinistro per selezionare, poi ci si sposta e si clicca sul tasto centrale per copiare.

Puo' anche essere utile sapere che per interromere un comando si usa il tasto Control assieme al tasto C.

Nomi di files

Spesso si usa la shell per guardare o modificare files, bisogna sapere alcune cose su come sono fatti i nomi dei files.

- abbiamo gia' detto che il filesystem ha una struttura gerarchica, con cartelle e sottocartelle, e che tutti i file system del computer, hard disk, CD, chiavette usb etc. vengono "montati" in un unico filesystem. L'inizio di questo file system (la root, o radice) e' indicato con una barra: "/" .

  I nomi completi dei files, quelli che partono dalla root, hanno una sintassi tipo: /cartella/sottocartella/nomefile con le barre che separano i nomi delle sotto-cartelle.

  Se un nome non inzia con "/" si intende che si parte, nell'albero dei files, dalla cartella in cui siete. Per cui, ad esempio, il file /home/jack/pollicino se siete in /home/jack lo listate con: *ls pollicino*, se siete da un'altra parte lo potete listare con il nome completo: *ls /home/jack/pollicino*

  Se ad esempio avete una sottocartella in /home/jack/ , di nome jobs con dentro il file "orco", se siete in /home/jack/ lo listate con: *ls jobs/orco*, altrimenti potete listarlo con: *ls /home/jack/jobs/orco*

- Lettere maiuscole e minuscole sono considerate cose diverse, per cui un file che si chiama "pollicino" e' diverso da un file che si chiama "Pollicino". Si dice che il nome dei files e' "case sensitive".
- Alcuni files iniziano con un punto, sono files di configurazione, e non vengono mostrati dal comando ls. Sono un po' come file nascosti, ma non e' che siano poi tanto ben nascosti, si vedono se si usa ls con l'opzione "-a": *ls -a*
- La directory corrente e' indicata da un singolo punto, per cui, ad esempio, *ls .* vi fa vedere i files nella direttory in cui siete esattamente come ls, senza opzioni.
- La directory sopra quella corrente e' indicata con 2 punti: ".." per cui, sempre nel contesto dell'esempio sopra, se siete in /home/jack/jobs potete listare il file "pollicino" con: *ls ../pollicino*, che significa: vai su di un livello e mostra il file pollicino.
- Il carattere "~" indica la cartella principale di proprieta' di un utente. Ad esempio se jack ha la sua cartella principale in: /home/jack , ~jack indica proprio questa cartella.
- L'asterisco indica qualunque insieme di caratteri, ad esempio p* sono tutti i files che iniziano con la lettera p. P*p tutti i files che iniziano con "P" e finiscono con "p".
- Il punto interrogativo indica un carattere qualunque, ma uno solo, per cui P?p indica files come: Pop Pap , ma non il file Poop.

- [ab] indica la lettera a oppure la lettera b, [ab]* sono tutti i files che iniziano con a oppure con b
- [a-z] indica le lettere dalla a alla z, per cui [a-z]* indica tutti i files che iniziano con lettere minuscole.
- Non usate spazi nei nomi di files, la shell li interpreta come separatori e si fa confusione, analogamente non usate caratteri strani. Se proprio dovete usare carateri speciali mettete il nome fra apici, ad esempio: “nome di file”
- La barra rovesciata “\” impedisce alla shell di interpretare il carattere seguente. Al esempio *ls \ ** mostra il file che come nome ha un asterisco, non tutti i files della cartella.

Comandi per files e cartelle

Un comando che crea un file (vuoto) e’: “touch” questo comando “tocca” il file e gli aggiorna la data di ultimo accesso, ma se il file non esiste lo crea.

touch pincopallo crea il file di nome *pincopallo*

per fare un directory di usa il comando *mkdir* (che sta per: “make directory”), per distruggere una directory si usa il comando “rmdir” (che sta per: “remove directory”). Una directory si puo’ eliminare solo se e’ vuota.

Per spostarsi in una directory si usa il comando: “cd” (che sta per “change direcotory”). Il comando “pwd” vi mostra in che directory siete.

Per vedere il contenuto di un file di nome pincopallo si puo’ dare il comando: *cat pincopallo* o meglio il programma “more”. Scrivendo: *more pincopallo* vedete il contenuto del file una pagina alla volta. (premete la lettera q per uscire dal programma more, q sta per “quit” che significa “smetti”).

Per cambiare il nome ad un file si usa il comando “mv”, che sta per “move”. *mv pollicino strega* cambia nome al file “pollicino” che adesso si chiama “strega”. mv sposta anche files da una cartella ad un’altra.

Ci sono anche tanti altri comandi per i files, eccone alcuni:

- tail pincopallo : vi fa vedere la fine del file pincopallo (tail significa coda)
- head pincopallo : vi fa vedere l’inizio del file pincopallo (head significa testa)
- wc pincopallo : conta caratteri, le parole, le linee
- grep stringa pincopallo : cerca una stringa in un file
- sort pincopallo : mette le linee in ordine alfabetico
- uniq pincopallo : elimina linee doppie
- diff pincopallo pincopallo1 : mostra le differenze fra 2 files

Bisognerebbe fare qualche esercizio per abituarsi a spostarsi nel file system con questi comandi, senza perdersi. La gente abituata a certe interfacce grafiche fa fatica ad utilizzare una semplice struttura gerarchica come il file system.

Input ed output

Ogni comando in unix ha un input, un output, ed anche un output speciale, dove vengono scritti i messaggi di errore.

Questi 3 canali di comunicazione si chiamano stdin, stdout ed stderr, che sta per: standard input, standard output e standard essor. A chi conosce il linguaggio C questo ricordera’ qualcosa, non per niente Unix (e Linux) sono scritti in C.

Quando usate la shell per dare comandi: l'input e' la tastiera, l'output la finestra terminale, lo standard error pure.

Ogni comando lavora un po' come una "pipe" cioe' un tubo, da una parte si butta dentro l'input, dall'altra esce l'output.

Finche' si lavora dando comandi da tastiera la cosa non significa granche', ma il bello e' che potete decidere di collegare l'output ad un file, e scrivere su un file, oppure di unire 2 comandi, in modo che l'uscita di uno finisce nell'ingresso dell'altro. In questo modo si creano comandi composti: mettendo insieme comandi semplici si montano comandi complicati; e' come collegare dei tubi.

Il comando piu' semplice per fare queste cose e' il comando "cat"; il nome significa "concatenate" perche' puo' servire per mettere insieme piu' files, come vedremo piu' avanti. *cat* fa una cosa semplice; copia il suo input sul suo output. Siccome il suo output e' il terminale se facciamo: *cat pollicino* vediamo sul terminale il contenuto del file "pollicino".

C'e' un altro comando che scrive su terminale, ed e': echo. *echo "stringa"* vi scrive la stringa su terminale (echo serve nelle procedure, cioe' nei files che contengono comandi di shell, da eseguire come un unico programma).

Per collegare i comandi, si usano i 3 simboli : "<" , ">" ed "|"

- "<" il comando prende l'input da un file, ovvero collega un file all'input;
- ">" collega l'output ad un file, ovvero il comando scrive su un file;
- "|" collega l'output di un comando all'input dell'altro ed unisce 2 comandi.

Vediamo qualche esempio:

```
ls > listafiles
```

crea un file di nome listafiles con la lista dei files della cartella (output del comando ls)

```
cat > nuovofile
```

questo prende tutto quello che scrivete sul terminale e lo mette nel file di nome: "nuovofile". Per riprendere possesso del terminale dovete premere il tasto control assieme alla lettera "d". Cntrol/d significa: fine del file. Quando il comando cat lo incontra pensa che il suo input sia finito e smette di lavorare.

```
cat nuovofile | more
```

cat mette il file: "nuovofile" sul suo output, ma il suo output finisce dentro il comando "more", che vi fa vedere il file una pagina per volta. Analogamente *ls | more* vi fa vedere la lista dei files una pagina per volta.

Il simbolo ">>" si usa al posto di ">" se si vuole aggiungere qualcosa al file su cui va l'output, che con ">" viene riscritto completamente da capo.

I links

I links sembrano files e cartelle vere, ma non lo sono, sono solo un'indicazione di dove il file o la cartella vera si trova. Pensate ad un cartello stradale, il link e' come un cartello stradale che dice dove si trova il file vero. Anche Windows ha i link e li chiama "collegamenti". I link sono comodi, perche' fanno da scorciatoie nell'albero delle directory e non vi obbligano sempre a fare lunghi percorsi per arrivare ad un file.

Il comando ls -l vi mostra quali sono i links e quali i files.

Per fare un link si usa il comando ln, che ha la sintassi: *ln nomefilevero nomelink*

Permessi di accesso ai files

Abbiamo gia' descritto i permessi di accesso ai files. Per cambiare permessi ad un vostro file si usa il comando chmod (abbreviazione per "change mode").

La sintassi puo' sembrare un po' involuta. Il primo parametro e' una combinazione delle lettere a,o,u,g , che stanno per all (tutti), others (gli altri) , u (user), g(group) ed indicano a chi si danno i permessi, poi c'e' un + se si aggiunge il permesso, un - se si toglie, poi ci sono le lettere: w,r,x per scrittura(write), lettura (read) ed esecuzione (execute) Vediamo qualche esempio:

- *chmod o-x /home/jack* : impedisce agli altri di entrare nella cartella; solo il padrone o il gruppo della cartella entrano
- *chmod a+rw pincopallo* : da a tutti la possibilita' di leggere e scrivere sul file pincopallo
- *chmod a+x programma* : da possibilita' a tutti di eseguire il programma
- *chmod ug-x programma* : impedisce al proprietario ed al gruppo di eseguire il programma.

Variabili di ambiente e shell come linguaggio di programmazione

La shell e' un vero e proprio linguaggio di programmazione e si usa per fare procedure, cioe' files contenenti comandi di shell che possonoe essere eseguiti, come un qualunque programma. La shell quindi ha tutti i soliti costrutti dei linguaggi di programmazione: if, then, else, le istruzioni ripetute, (i loop), con while, for ; strutture simili a quelle che si possono trovare nel Pascal, nel C od in altri linguaggi.

Siccome e' un linguaggio ha anche delle variabili, non si deve definire il loro tipo prima di usarle perche' in genere sono delle stringhe; la shell non e' fatta per scrivere programmi di calcolo.

Alcune di queste variabili definiscono il comportamente della shell, fra queste ricordiamo:

- La variabile PATH, che contiene l'elenco delle cartelle dove vengono cercati comandi da eseguire;
- DISPLAY : indica il video su cui vengono disegnate le finestre;
- USERNAME : il nome dell'utente.
- HOME : la vostra directory principale ;
- PS1 : il "prompt", la stringa che viene scritta a sinistra, a inizio linea, prima dei vostri comandi

Per vedere il valore di una variabile si puo' usare il comando echo: *echo $HOME* vi mostra la vostra cartella principale. Il carattere "$" serve a dire alla shell che HOME e' una variabile e non una stringa, e va sostituita con il suo valore, se scrivete *echo HOME* , senza il "$" vedete la scritta "HOME", infatti echo scrive una stringa sul terminale.

Per vedere quali sono le variabili gia' definite nel vostra ambiente di shell potete scrivere: "export"; sono tante, ma non serve conoscerle tutte. Quello che e' interessante e' che le potete cambiare.

Vediamo un esempio: PS1 e' il prompt, e spesso e' fatto in modo da farvi vedere il vostro nome o la macchina su cui state lavorando. Se fate: PS1="xxx " avrete ad inizio riga tre x, seguite da uno spazio. Le variabili si assegnano o si cambiano con un segno "=", ma valgono solo nella finestra in cui siete; se volete che restino valide anche all'interno dei comandi che date dopo, dovete usare il comando export. Ad esempio il comando: export PATH=".:$PATH" , aggiunge un punto (la directory corrente) alla directory in cui cercare comandi, e, con export, vale anche nelle finestre che fate a partire dalla vostra.

Interazione col sistema utilizzando la shell

In unix ci sono molti comandi per controllare e gestire il vostro computer, per utilizzarli si una una finestra terminale e si scrivono al terminale i loro nomi, la shell li fara' partire e vi mostrera' i risultati.

Vediamo alcuni di questi programmi:

- *who* : mostra chi sta lavorando nel vostro PC, e da quando si e' collegato. C'e' una versione abbreviata: "w" , che vi mostra da dove si e' collegato.
- *df* : mostra quanto spazio e' rimasto sui dischi, disco per disco.
- *du*: vi mostra quanto spazio usano le vostre cartelle, se sono molte e' conveniente guardare il risultato una pagina per volta: *du | more*
- *ps* mostra i processi, ps ha un sacco di parametri diversi, se fate: *ps aux* vedete tutti i programmi che stanno correndo nel computer. Ad inizio riga vedete il proprietario dei processi, poi c'e' un numero che e' il numero che identifica il processo (detto PID: process identifier); se volete fermare un processo potete usare il comando kill con questo numero; kill serve a mandare messaggi ad un processo, ma se gli mandate il segnale 9 il processo muore, la sintassi e': *kill -9 numeroprocesso* Potete uccidere solo i vostri processi (ad esempio il terminale su cui scrivete), solo root puo' uccidere tutti i processi. A volte e' utile uccidere un processo impazzito.
- *top* e' un comando che vi mostra chi sta utilizzando piu' cpu e quanto e' impegnato il computer (si esce con Cntrl/C)
- *date*, mostra che ora e'.
- *last*: mostra gli ultimi che si sono collegati alla macchina e quanto sono stati collegati.
- *uname -a* : mostra la versione del sistema operativo;
- *groups* : mostra a che gruppi di utenti si appartiene;
- *whoami* : mostra chi siete. Sembrerebbe un comando inutile, ma usando molte finestre contemporaneamente, con utenti diversi, su macchine diverse, capita di avere dubbi sulla propria identita'.