

CAPITOLO 5

LA SHELL BASH

Pietro Buffa

Con questo capitolo comincia il nostro viaggio nel mondo delle *Command Line Interface (CLI)*, ovvero le interfacce testuali a linea di comando tipiche dei sistemi UNIX - based. Quando lavoriamo in modalità testuale, utilizzando magari un terminale a finestra all'interno di X, immettendo comandi, visualizzando liste di files o controllando i processi in esecuzione, non comunichiamo direttamente con il kernel, tra noi ed il cuore del sistema c'è un mediatore, che protegge l'utente dalle trappole del kernel ed il kernel dalla distrazione dell'utente. Questa sorta di "guscio" che si interpone tra l'utente ed il kernel prende infatti il nome inglese di shell. La shell svolge diversi compiti fondamentali: si occupa di accettare input dell'utente, interpretare i comandi e mettere a disposizione vari strumenti per poter interagire ed utilizzare efficacemente il sistema, restituendo output ed errori in modo appropriato dopo aver fatto eseguire al kernel i processi desiderati.

Come solitamente accade, in un sistema GNU/Linux possiamo trovare installate differenti shell: **cs**h (C shell), **ks**h (Korn shell), **tc**sh (TC shell). A queste va aggiunta la **bash** (Bourne Again shell), la versione GNU della shell Bourne, la shell UNIX originale, sviluppata dalla Free Software Foundation e che oggi rappresenta la scelta di default della stragrande maggioranza delle distribuzioni.

5.1 IL COMANDO bash

Per invocare una qualsiasi shell è sufficiente digitarne il nome al prompt dei comandi e premere invio. Se la shell è installata sul vostro sistema entrerà in esecuzione. Per determinare quale shell è in uso, digitate il comando:

```
> echo $SHELL
```

Se state utilizzando la shell bash, ovvero la shell di riferimento per questo manuale, il terminale visualizzerà il seguente output:

```
/bin/bash
```

Data la sua complessità intrinseca, discuteremo più dettagliatamente del sopracitato comando più avanti, per adesso limitatevi ad utilizzarlo "alla cieca".

Se non state utilizzando la shell bash, digitare semplicemente il seguente comando per mandarla in esecuzione:

```
> bash
```

5.2 PERCHE' UTILIZZARE LA SHELL BASH

Ogni shell possiede caratteristiche proprie. Con il passare del tempo, alcune di queste caratteristiche, si sono fatte apprezzare più di altre e questo ha avuto come conseguenza diretta l'affermazione di alcune shell, ritenute più flessibili rispetto ad altre. Bash è oggi sicuramente la shell più utilizzata ed apprezzata e basta elencare alcune sue caratteristiche fondamentali, che approfondiremo nel corso di questo manuale, per rendersi subito conto delle sue straordinarie capacità.

1) STORICO DEI COMANDI

Lo storico dei comandi (cronologia), è un registro degli ultimi comandi inseriti dall'utente. Questo registro è generalmente salvato in un file nascosto chiamato *.bash_history* all'interno della vostra personale directory home. L'utente è così in grado di ripescare facilmente un comando utilizzato poco prima, senza doverlo riscrivere completamente, con la possibilità di mandarlo nuovamente in esecuzione, modificarlo o completarlo.

2) COMPLETAMENTO AUTOMATICO

Il completamento automatico, aiuta a scrivere automaticamente il nome di un file o di un percorso, dopo aver digitato solo le prime lettere, semplicemente premendo il tasto TAB. Se esiste ambiguità, cioè se esistono due file con le stesse lettere iniziali, otterrete un elenco premendo due volte TAB.

3) INTERPRETAZIONE DEI COMANDI

Quando la shell viene utilizzata, attende che l'utente digiti dei comandi, li elabora, interpretando per esempio eventuali caratteri speciali (metacaratteri) e se tutto è andato a buon fine, li esegue. Questo è il motivo per cui si utilizza spesso il nome di "interprete dei comandi".

4) ALIAS

Alcune shell permettono la definizione di nuovi comandi creando alias di comandi già esistenti. Per comprendere il senso di tutto questo consideriamo un semplice esempio: immaginiamo di voler creare l'alias "**dir**", comando molto utilizzato da chi viene dal mondo DOS che mostrava il contenuto di una determinata directory, riferito al comando "**ls -l**" che ha, come avremo modo di vedere, lo stesso effetto sui sistemi GNU/Linux. Una volta generato l'alias, si potrà tranquillamente utilizzare il comando **dir** per visualizzare il contenuto di una determinata directory anche su di un sistema GNU/Linux, la shell si farà carico della corretta interpretazione del comando.

5) VARIABILI

Una variabile è, in generale, un'area di memoria alla quale viene assegnato uno specifico valore. Alcune variabili vengono impostate dal sistema, mentre altre possono essere definite all'interno di file di configurazione, che la shell legge durante la fase di avvio.

La shell bash supporta ben tre tipi differenti di variabili:

- Variabili di configurazione
- Variabili di shell definite dal programmatore
- Variabili di ambiente

In particolare le variabili d'ambiente sono un mezzo elementare e pratico di configurazione del sistema in quanto definiscono l'ambiente nel quale ogni programma opera. Molti programmi infatti, a seconda dei loro compiti e del loro contesto, cercano di leggere alcune variabili di loro interesse ed in base al contenuto di queste, adeguano il loro comportamento.

6) PIPELINE E REINDIRIZZAMENTO DEI COMANDI

La shell bash è in grado di ridirigere a nostro piacimento il flusso di dati standard, ovvero: standard Input, standard Output e standard Error. Questa caratteristica, che avremo modo di vedere in dettaglio in un apposito capitolo, è fondamentale per la realizzazione dei così detti **comandi complessi** ed è forse la caratteristica più apprezzata nel mondo UNIX - Linux.

7) CARATTERISTICHE DI PROGRAMMAZIONE

Le shell sono generalmente ambienti programmabili, bash, in tal senso, è considerata la migliore. Una serie di istruzioni precise, combinati in un file eseguibile dalla shell, formano ciò che viene chiamato **shell script**.

8) UTILIZZO DEI CARATTERI JOLLY

I caratteri jolly (metacaratteri), sono dei simboli utilizzati per fare facilmente riferimento a gruppi di file o di directory. Nei sistemi UNIX/Linux è la shell ad occuparsi della traduzione e quindi dell'espansione dei caratteri jolly eventualmente utilizzati all'interno dei comandi.

9) CONTROLLO DEI JOBS

Caratteristica tecnica di grande rilievo che consente all'utente di lavorare utilizzando più programmi contemporaneamente mediante il preemptive multitasking, anche da terminale.

5.3 FILE DI CONFIGURAZIONE FONDAMENTALI

Quando cominciamo una sessione di lavoro utilizzando un terminale a finestra, viene automaticamente creata una shell bash di login interattiva. La shell bash inizia la sua attività leggendo ed eseguendo le istruzioni che si trovano in particolari **files di configurazione generale**. Essi indicano le impostazioni di alcuni parametri di sistema, valide per tutti gli utenti. I più comuni sono:

```
/etc/profile  
/etc/bashrc
```

Successivamente vengono lette ed eseguite le istruzioni all'interno dei **files di configurazione personale**, generalmente posizionati all'interno della home directory dell'utente e non visibili in quanto file di sistema nascosti. Essi indicano le impostazioni per alcuni parametri di sistema, valide soltanto per il singolo utente. I più comuni sono:

```
~/.bash_profile  
~/.bashrc
```

E' diffusa l'usanza di creare, nel caso non sia presente, o modificare, nel caso sia presente, il proprio file *~/.bashrc* in modo che ogni singolo utente possa personalizzare alcuni parametri personali che riguardano il proprio ambiente.

Assicuratevi che all'interno del file *.bash_profile* siano presenti le seguenti righe:

```
if [ -f ~/.bashrc ]; then  
    . ~/.bashrc  
fi
```

In questo modo, ogni volta che la shell andrà in esecuzione, verrà letto anche *~/.bashrc*. Potremo quindi fare ogni modifica in *~/.bashrc* con la sicurezza che questo file verrà sempre letto.

Riportiamo di seguito, a titolo esemplificativo, un piccolo file *~/.bashrc* che mostra la personalizzazione di alcuni parametri. Tutte le stringhe che cominciano con il simbolo "#", verranno considerate dalla shell come commenti e quindi non interpretate, il loro scopo è esclusivamente quello di documentare determinati punti del file in questione:

```
-----  
# imposta l'history in modo che contenga 20 comandi, agendo sulla variabile d'ambiente HISTSIZE.  
HISTSIZE=20  
  
# imposta il prompt dei comandi nel modo completo, agendo sulla variabile d'ambiente PS1.  
PS1='\h: \w >>>'  
export PS1  
  
# imposta il percorso in cui bash deve ricercare comandi e programmi.  
PATH=$PATH:~/myscript  
export PATH  
  
# Mostra i diversi tipi di file utilizzando colori diversi.  
alias ls='ls --color=auto'  
-----
```

Al termine della sessione di lavoro, bash legge ed esegue il contenuto di: *~/.bash_logout*, in questo file sono presenti tutte le operazioni di chiusura che l'utente vuole eseguire.

5.4 PERSONALIZZARE IL PROMPT DEI COMANDI

Abbiamo più volte ripetuto che quando la shell funziona in modo interattivo, mostra all'utente il prompt dei comandi. Il prompt è generalmente definito dalla variabile di ambiente "PS1".

Aprirete un terminale e provate a digitare il comando:

> **echo \$PS1** e premete invio

Apparirà il contenuto della variabile PS1, che potrebbe presentarsi leggermente criptico. Il contenuto di questa variabile è infatti una stringa composta da alcuni simboli speciali preceduti dal carattere di escape “\” a cui bash assegna il significato indicato nella seguente tabella:

Codice	Descrizione
\t	Visualizza l’orario attuale nel formato <i>hh:mm:ss</i> (ore, minuti, secondi).
\d	Visualizza la data attuale.
\n	Interruzione di riga.
\s	Visualizza il nome della shell in esecuzione.
\w	Visualizza la directory corrente.
\W	Percorso precedente alla directory corrente (<i>basename</i>).
\u	Visualizza il nome dell’utente corrente.
\h	Visualizza il nome host del computer sul quale la shell è in esecuzione.
\#	Numero del comando attuale.
\!	Numero del comando nello storico.
\\$	# se UID = 0; \$ se UID > 0.
\nmn	Carattere corrispondente al numero ottale indicato.
\	Una barra obliqua inversa singola (\).
\[Inizio di una sequenza di controllo.
\]	Fine di una sequenza di controllo.

In particolare merita attenzione il simbolo speciale “\\$”, il cui significato potrebbe non essere chiaro dalla breve descrizione fatta in tabella. Esso rappresenta un simbolo che cambia in funzione del livello di importanza dell’utente: se bash rivela uno UserID pari a zero (utente “root”), assegna il simbolo “#” per indicare la fine del prompt, negli altri casi assegna il simbolo “\$”.

La shell quindi legge il contenuto della variabile d’ambiente PS1 all’avvio, interpreta ed espande i simboli appena visti e fornisce all’utente un prompt dall’aspetto più gradevole.

Provate adesso a modificare il prompt dei comandi del vostro terminale a piacimento, assegnando alla variabile PS1 alcuni dei valori presenti nella tabella, come mostrato nell’esempio seguente:

```
> PS1='\d\t > '
```

Riavviamo la shell bash in maniera tale che possa rileggere la variabile PS1, digitando il seguente comando:

```
> bash -l
```

La shell visualizzerà adesso un prompt con data ed ora.

Proviamo adesso ad assegnare alla variabile PS1 la seguente combinazione di valori:

```
> PS1='\u@\h:\w >>> '
```

Otterremo un prompt che visualizza il nome dell’utente, il nome dell’elaboratore ed il percorso della directory corrente. Il simbolo finale “>>>”, ha semplicemente lo scopo di evidenziare maggiormente la fine del prompt. Questo viene generalmente detto “prompt completo” poiché ricco di informazioni.

In questa maniera è quindi possibile personalizzare il prompt dei comandi a proprio piacimento, ma una tale modifica, ha però la durata della sessione di lavoro. Al riavvio del computer infatti, il sistema mostrerà nuovamente il prompt originale poiché la variabile d’ambiente che abbiamo settato, tornerà al suo valore originale. L’assegnamento fatto alla variabile PS1, è quindi solo temporaneo. Per ovviare a tale problema, bisogna agire sul file di configurazione specifico che la shell bash legge al suo avvio e che definisce stabilmente il valore che dovrà assumere la variabile PS1. Linux è famoso per la sua completa configurabilità, modellabile a nostro piacimento come un “pezzo di argilla”, con una sola limitazione: che siate amministratori e che sappiate bene dove mettere le mani. Vediamo come procedere.

Andiamo ad intervenire su di uno specifico file di sistema per personalizzare, questa volta in maniera permanente, il prompt dei comandi a nostro piacimento. Anche se questo esempio non ha una vera utilità pratica, sappiate che la modifica di qualunque file di configurazione in GNU/Linux, avviene, modificando in maniera opportuna, specifici file di testo come quello che adesso andremo a configurare, quindi prestate ugualmente molta attenzione.

Per svolgere questo esempio, potete adoperare un semplice editor grafico, i più conosciuti sono: kate, se lavorate in ambiente KDE o gedit, se lavorate in ambiente GNOME.

La shell bash, come abbiamo già detto all'inizio del capitolo, prima di presentarsi all'utente, legge alcuni importanti file di configurazione. In particolare ci interessa il file `~/.bashrc`, posto all'interno della vostra personale directory home come file nascosto. La modifica di alcune righe di questo file ci consentirà di personalizzare, in maniera definitiva, il prompt dei comandi, inoltre, le modifiche che apporteremo riguarderanno soltanto l'utente in questione. Eventuali altri utenti visualizzeranno il prompt di default. Viceversa, la modifica di `/etc/bashrc` o di `/etc/profile` porta a variazioni che riguardano tutti gli utenti del sistema, poiché questi non sono file personali.

Editiamo quindi il file `~/.bashrc` utilizzando un editor di testo ed aggiungiamo la stringa di assegnamento per la variabile PS1, come mostra l'esempio in basso:

```
PS1="\h:\w >>> '  
export PS1
```

Salviamo il file così modificato.

Riavviamo la shell bash utilizzando il comando:

```
> bash -l
```

In modo da consentire alla shell di riavviarsi e rileggere questo file di configurazione impostando la variabile PS1 al valore da noi inserito. Il prompt dei comandi dovrebbe adesso avere un aspetto simile al seguente:

```
Nome_HOST:~ >>>
```

ATTENZIONE: Tra degli strumenti che un buon utilizzatore GNU/Linux deve imparare a padroneggiare vi è senza dubbio l'uso di alcuni programmi, che consentano di editare, modificare e creare file di testo, anche quando, per diversi motivi, non si dispone di un ambiente grafico. Il più potente editor testuale, disponibile su tutti i sistemi GNU/Linux è *Vi*, ci occuperemo di questo formidabile programma in uno specifico capitolo di appendice.
