Shell Bash

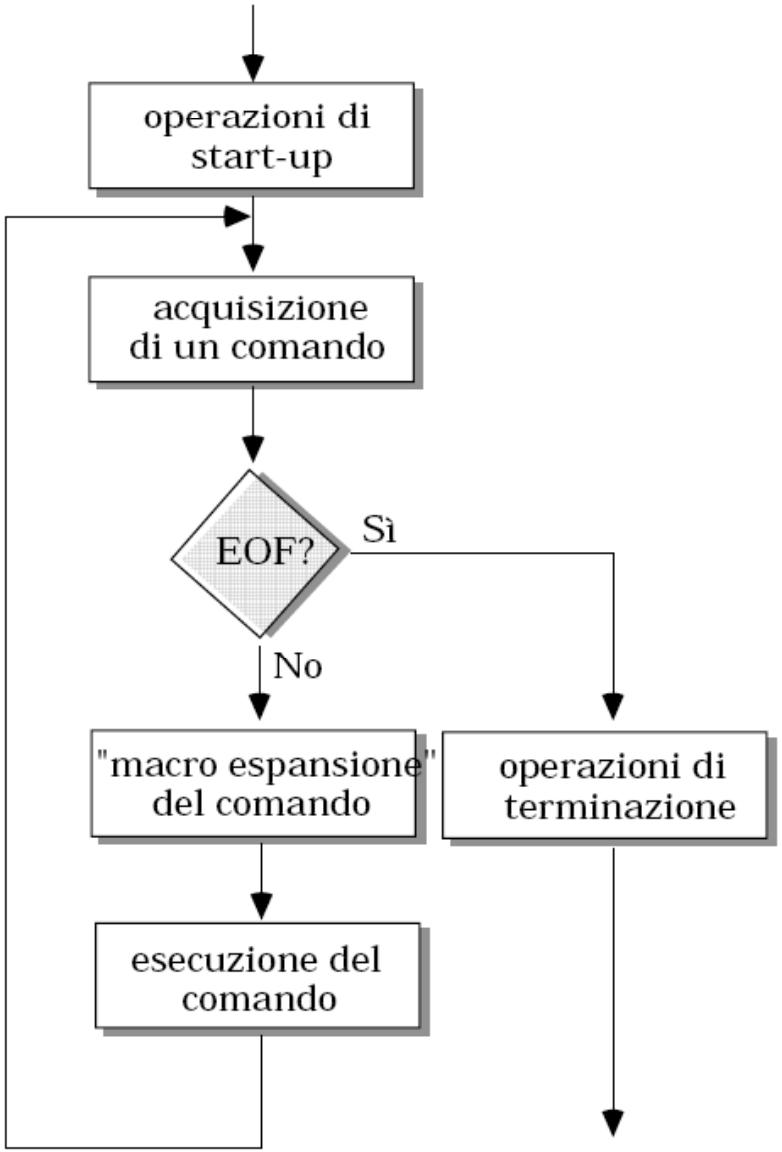
Shell

Programma che interpreta il linguaggio a linea di comando attraverso il quale l'utente utilizza le risorse del sistema. Permette la gestione di variabili e dispone di costrutti per il controllo del flusso delle operazioni.

Viene generalmente eseguito in modalità interattiva, all'atto del login, restando attivo per tutta la durata della sessione di lavoro ed effettuando le seguenti operazioni:

- Gestione del "main command loop";
- Analisi sintattica;
- Esecuzione di comandi ("built-in", file eseguibili) e programmi in linguaggio di shell (script);
- Gestione dello standard I/O e dello standard error ;
- Gestione dei processi da terminale.

Ciclo Esecuzione Shell



Variabili di shell predefinite

Esistono delle variabili di shell predefinite (variabili di ambiente), che permettono di caratterizzare il comportamento della shell.

Per convenzione, il nome di tali variabili è in caratteri tutti maiuscoli:

- **HOME** argomento di default per il comando **cd**, inizializzato da login con il path della home directory, letto dal file **/etc/passwd**;
- **PATH** Il path di ricerca degli eseguibili;
- **PS1** stringa del prompt, di default " $ " per l'utente normale e "#" per il super-user;
- ....

Variabili predefinite

- PATH percorso di ricerca eseguibili
- USER nome utente
- HOME directory home dell'utente
- PS1 il prompt
- HOSTNAME nome computer
- SHELL la shell corrente
- ...

Shell Interattiva

- Comunicazione tra utente e shell avviene tramite comandi o script:
- Nome comando bult-in oppure
- Nome di un file eseguibile oppure
- Nome di Script, cioè file ASCII presente nel sistema dotato del premesso di esecuzione.

Sintassi dei comandi

comando [*argomento* ...]

Gli argomenti possono essere:

- opzioni o flag (-)
- parametri

separati da almeno un separatore

Nota: Il separatore di default è il carattere spazio; per alcune shell può essere modificato grazie alla ridefinizione di una variabile d'ambiente opportuna (cfr. seg.).

Una volta interpretata la prima parola sulla linea di comando, la shell ricerca nel file system un file con il nome uguale a tale prima parola.

La ricerca avviene ordinatamente all'interno delle directory elencate nella variabile d'ambiente PATH

Variabili

- Scrittura/definizione: a=3 (senza spazi)
- Lettura: ${a} o semplicemente $a

Esempi:

```
> a=3
> echo $a
3
> echo $aa
> echo ${a}a
3a
```

```
> a=ciao pippo
bash: pippo: command not found
> echo “ecco: $a”
ecco: 3
> echo 'ecco: $a'
ecco: $a
```

Comando echo

echo [*argomenti*]

Visualizza gli argomenti in ordine, separati da singoli blank

Esempio:

```
% echo uno due tre
uno due tre
%
```

echo $SHELL
echo $PATH

(Ri)definizione di variabili di shell

La shell offre all'utente sia la possibilità di ridefinire alcune variabili d'ambiente, sia di definire delle nuove variabili a proprio piacimento.

Esempio 1

```
$ frutto=mela
$ verbo=mangia
$ nome=Stefania
$ echo $nome $verbo una $frutto
Stefania mangia una mela
$
```

(Ri)definizione di variabili di shell

Esempio 2

```
$ echo $PATH
$ /usr/bin:/home/gio:.
$ ps
sh: ps: No such file or directory
$ PATH=$PATH:/bin
$ ps
PID TTY TIME CMD
2487 ttyp1 00:00:00 sh
2488 ttyp1 00:00:00 ps
$
```

(Ri)definizione di variabili di shell

Esempio 3

```
$ frutto=mela
$ frutto=${frutto}banana
$ echo $frutto
melabanana
$ tipo="mela banana"
$ echo $tipo
mela banana
$
```

File Standard

Normalmente, un programma (comando) opera su più file

In Unix esiste il concetto di **file standard**:

File standard	Che cos'è
standard input	il file da cui normalmente il programma acquisisce i suoi input
standard output	il file su cui normalmete un programma produce i suoi output
standard error	il file su cui normalmente un programma invia i messaggi di errore

Redirezione std I/O

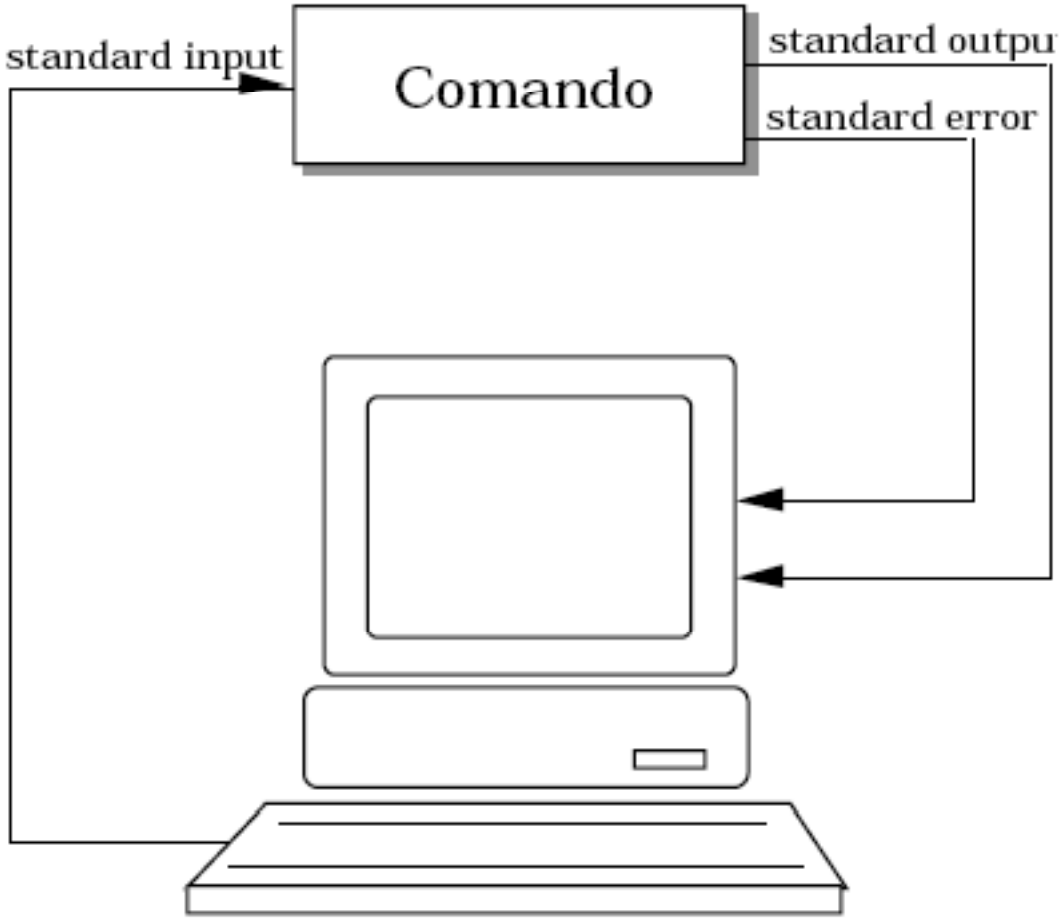
- I programmi dispongono di 3 canali di comunicazione:
  - Standard input (codice 0), per input
  - Standard output (1), per output
  - Standard error (2), per errore

Normalmente:

Standard input = tastiera

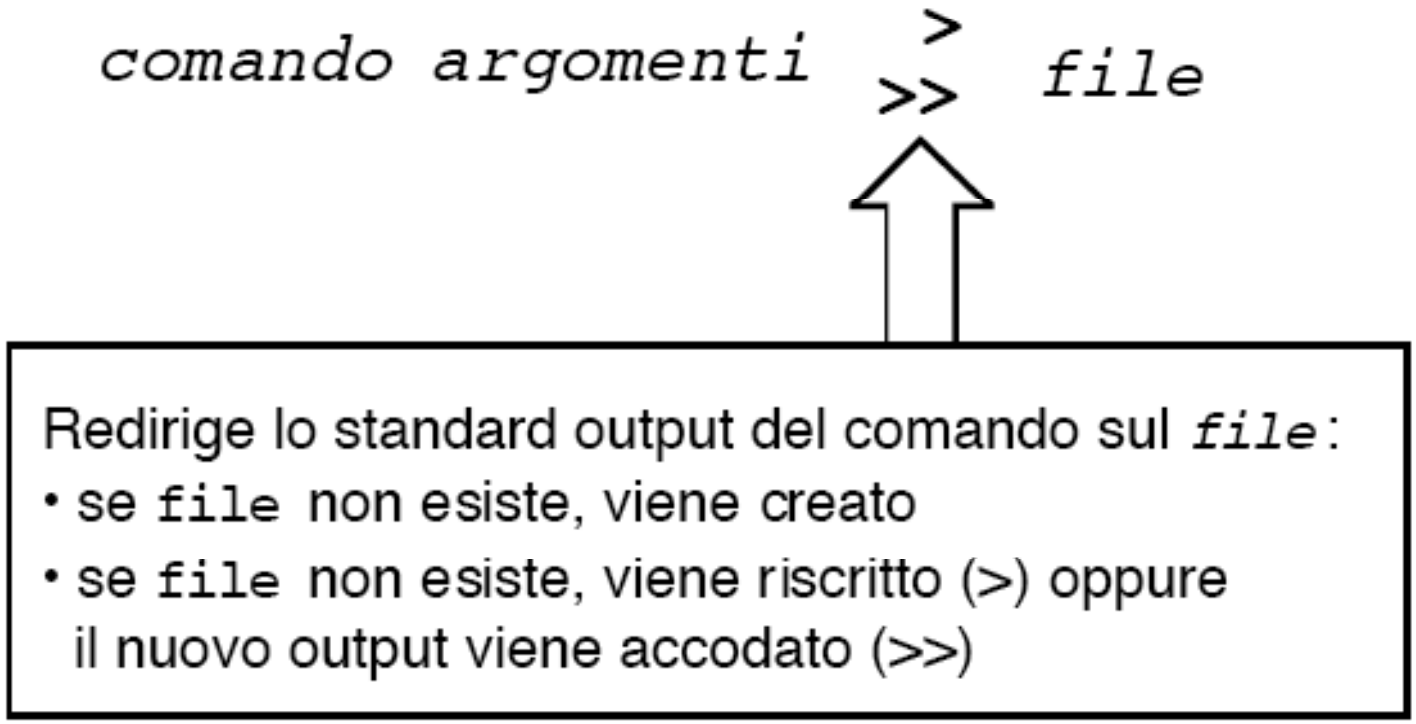
Standard output= schermo

Redirezione File Standard



La shell può variare queste associazioni di default **redirigendo** i files standard su qualsiasi file nel sistema

Ridirezione Stn Output

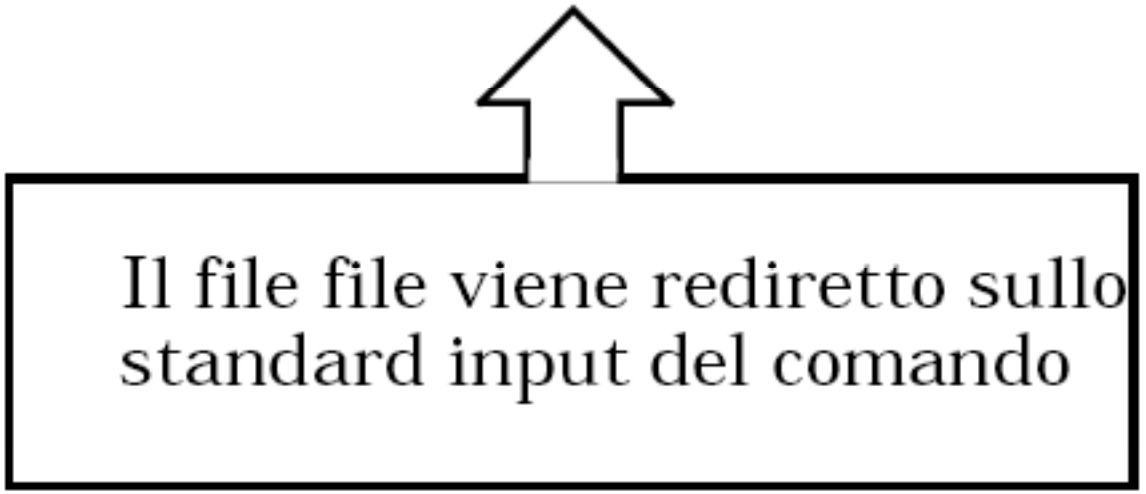


~>ls -a > listaFile.txt

~>echo $PATH >> listaFile.txt

Ridirezione Stn Input

command arg1 ... argn < file



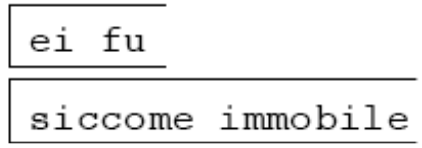
Comando cat

cat *file...*

"concatenate"

Concatena i *file* e li scrive sullo standard output...

```
% cat file1 file2
ei fu
siccome immobile
% cat file1 file2 > file3
%
```

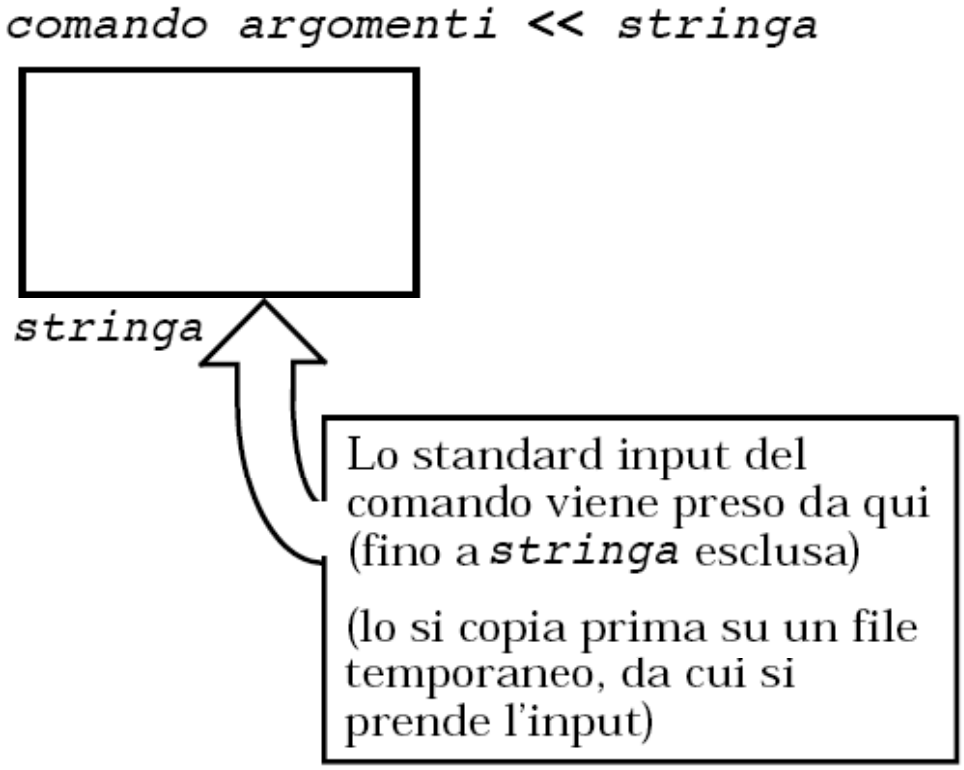


... a meno che manchino gli argomenti, nel qual caso scrive lo standard input sullo standard output

Comando cat

Esempio:

```
%cat << :
caro amico,
leggi questa
lettera
:
caro amico,
leggi questa
lettera
%
```



Ridirezione Stn Error

comando argomenti **2>** *file*

2>>

(Analogo a > e >>)

Esempio:

> echho “ciao!”
bash: echho: command not found
> echho “ciao!” 2> /dev/null

Ridirezione

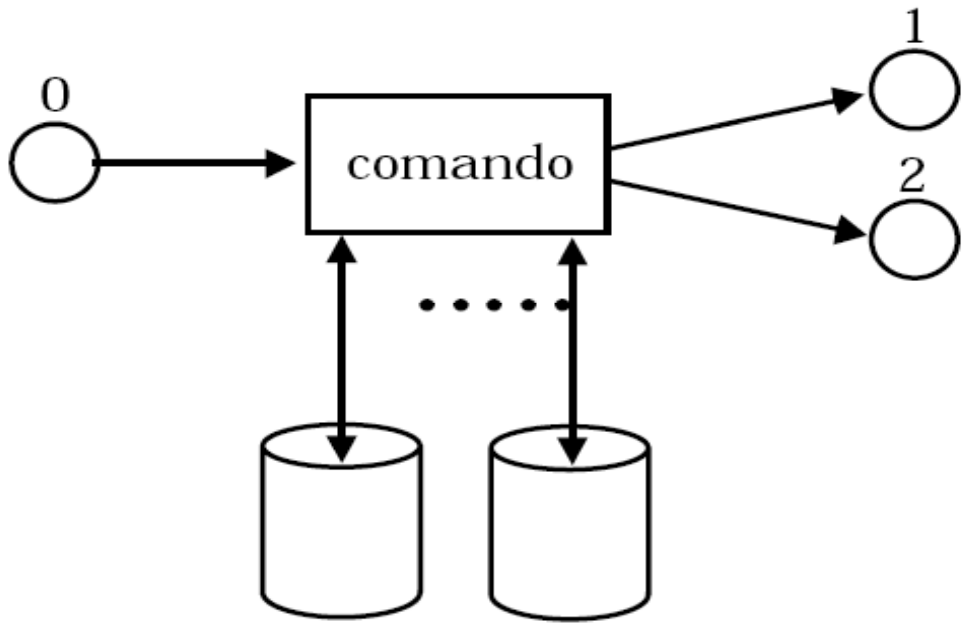
comando (codiceA)>&(codiceB) redirige il canale A sul canale B

- esempio: comando > file 2>&1

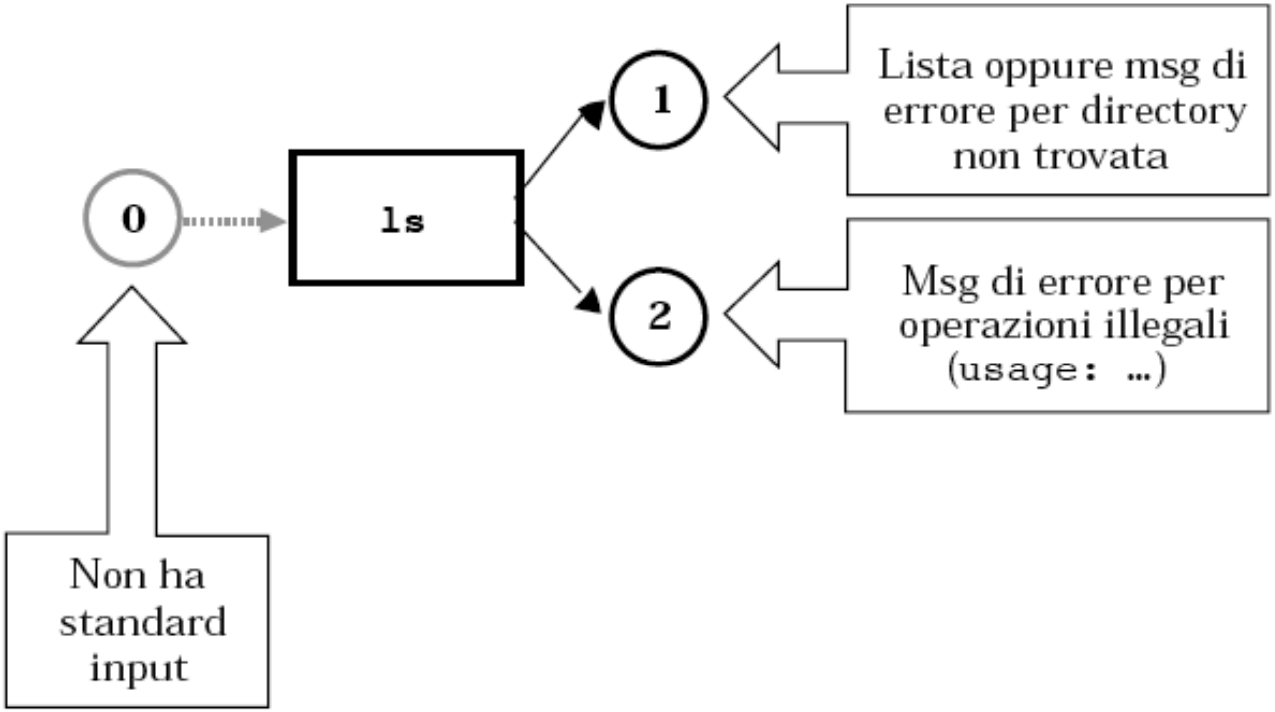
Ridirezione

Per redigere correttamente, è necessario conoscere, di ogni comando:

- come usa lo standard input
- come usa lo standard output
- come usa l'error output
- come usa eventuali altri files



Esempio



Inoltre accede ai files di sistema:

`/etc/passwd` per trovare lo user name

Pipe (tubo)

```
comando1 | comando2
```

Pipeline di due o più comandi:
Lo standard output di com1 funge da input a com2...

- ***com1 [arg ..] | com2 [arg ..]..|..***

Esempi di comandi concatenabili: cat, sort, wc

~> cat file | sort ~> ls | less

Esercizi

- Creare un file che si chiami come l'utente corrente
- Creare un file che si chiami come l'host corrente, e che contenga il nome dell'host corrente

Command substitution

- Il pattern $(comando) viene sostituito con l'output del comando
- Esempi:
  - $(ls) equivale a *
  - $(echo ciao) equivale a ciao
  - $(cat nomefile) equivale all'intero contenuto del file
  - a=$(ls) assegna ad a l'elenco dei file nella dir corrente
  - touch “$(date)” crea un file chiamato come la data attuale

Gestione dei processi da terminale

Quando si richiede alla shell di eseguire un comando, questo viene lanciato come un processo a se stante. I processi lanciati dalla shell si distinguono in due tipi:

- I processi in foreground sono quelli che sottraggono alla shell il controllo del terminale durante la loro esecuzione;
- Nei processi in background, il controllo del terminale rimane alla shell: il prompt appare immediatamente dopo che il processo è stato avviato.

Per default, l'esecuzione di un comando o di uno script genera un processo in foreground.
Per segnalare alla shell che si vuole lanciare il comando (o lo script) in background, è necessario farne seguire il nome dal carattere & :

- ***comando [argomento..] &***