

Shell Bash

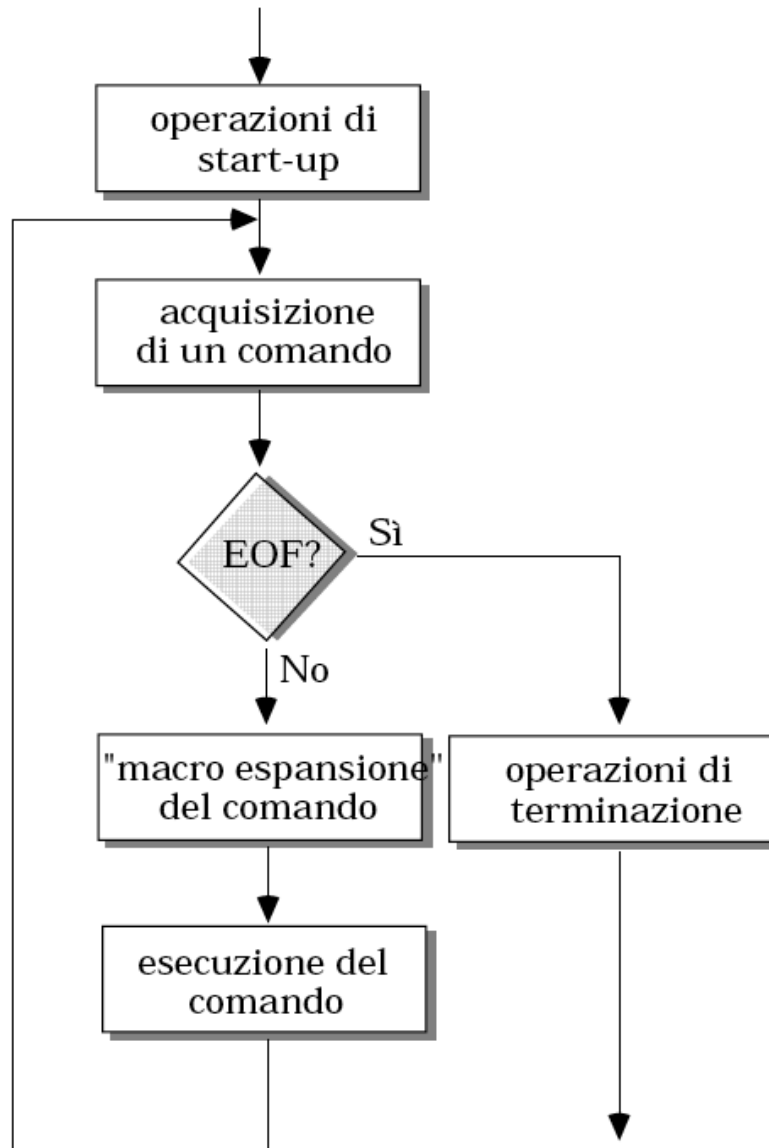
Shell

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 built-in oppure
- Nome di un file eseguibile oppure
- Nome di Script, cioè file ASCII presente nel sistema dotato del premezzo 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 tty1 00:00:00 sh
2488 tty1 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 normalmente un programma produce i suoi output
standard error	il file su cui normalmente un programma invia i messaggi di errore

Redirezione std I/O

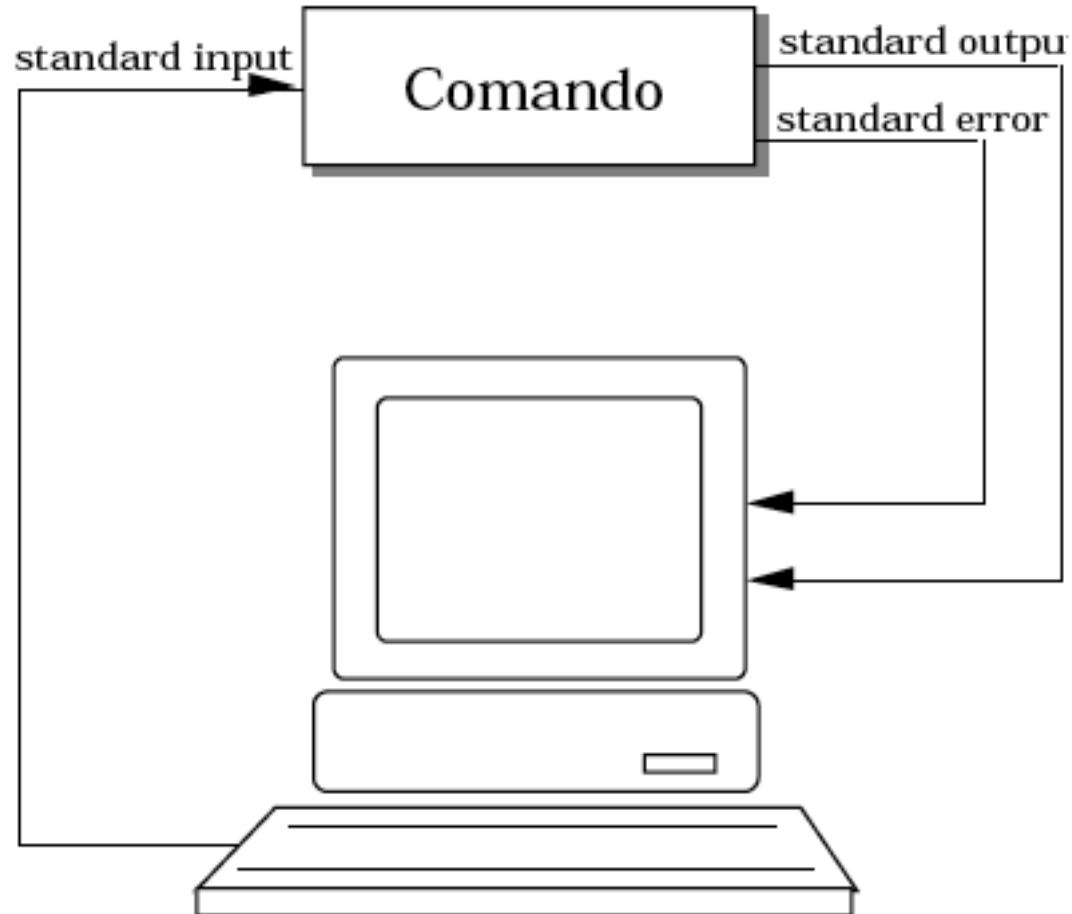
- I programmi dispongono di 3 canali di comunicazione:
 - Standard input (codice 0), per input
 - Standard output (1), per output
 - Standard error (2), per errore

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 Std Output

comando argomenti $\begin{matrix} > \\ >> \end{matrix}$ *file*



Redirige lo standard output del comando sul *file*:

- se *file* non esiste, viene creato
- se *file* non esiste, viene riscritto (>) oppure il nuovo output viene accodato (>>)

~>ls -a > listaFile.txt

~>echo \$PATH >> listaFile.txt

Ridirezione Std Input

command arg1 ... argn < file



Il file file viene rediretto sullo
standard input del comando

Comando cat

```
cat file...
```

"concatenate"

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

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

comando argomenti << stringa



stringa

Lo standard input del
comando viene preso da qui
(fino a *stringa* esclusa)
(lo si copia prima su un file
temporaneo, da cui si
prende l'input)

Ridirezione Std Error

```
comando argomenti 2> file  
2>>
```

(Analogo a > e >>)

Esempio:

```
> echho "ciao!"  
bash: echho: command not found  
> echho "ciao!" 2> /dev/null
```

Ridirezione

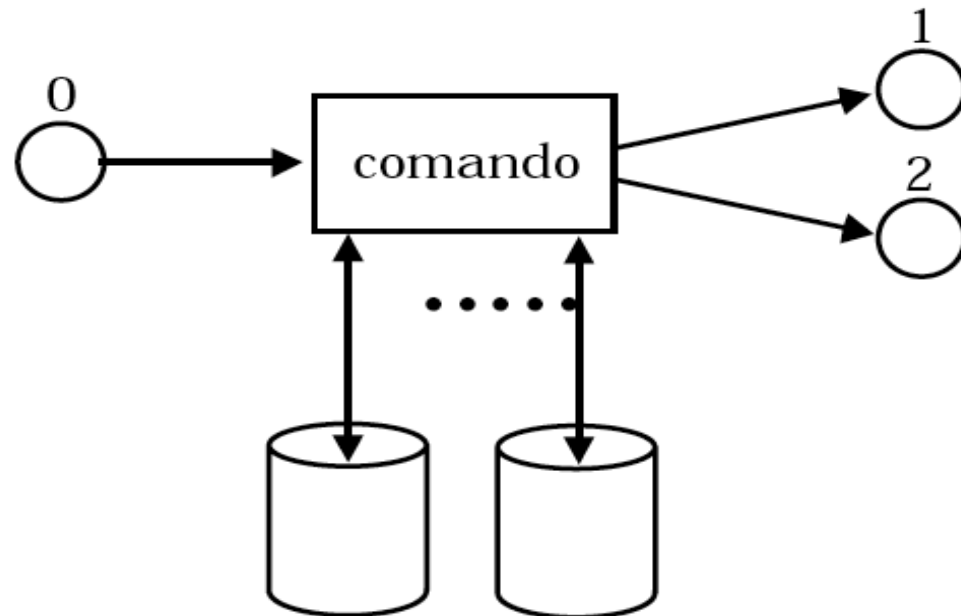
comando (codiceA)>&(codiceB) reindirige il canale A sul canale B

- esempio: comando > file 2>&1

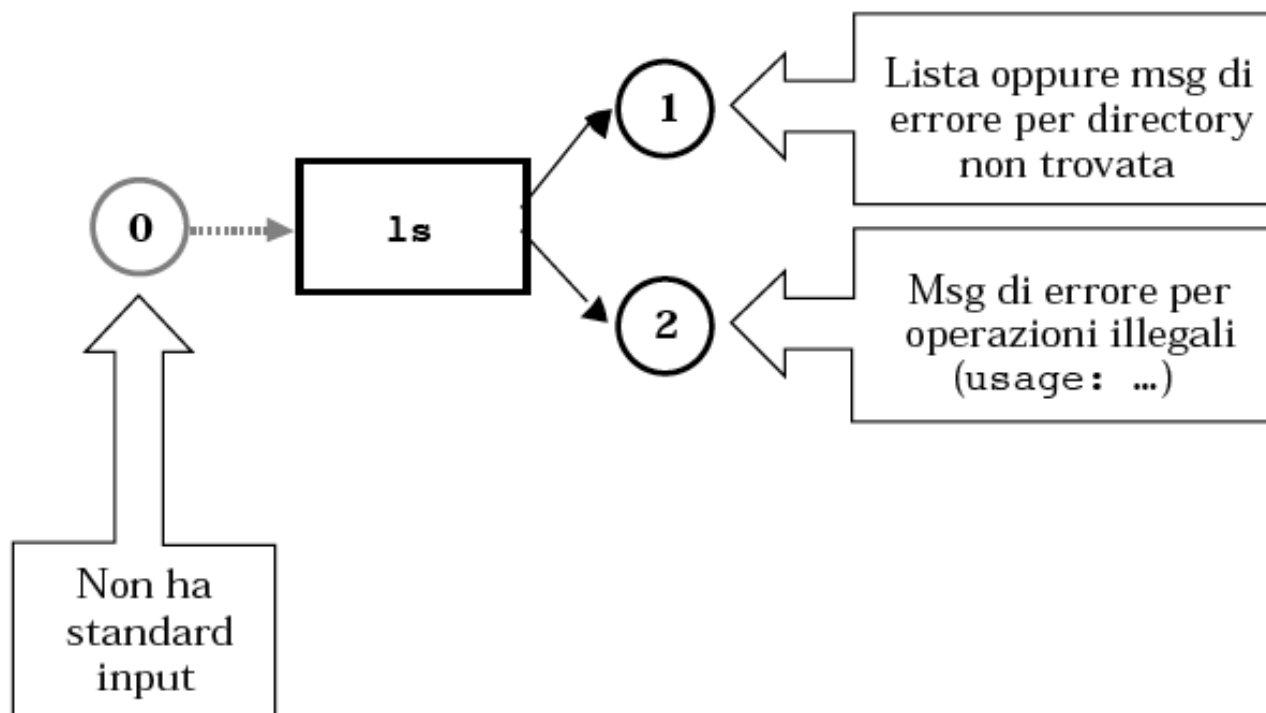
Ridirezione

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

Metacaratteri

- La shell riconosce alcuni caratteri speciali, chiamati **metacaratteri**, che possono comparire nei comandi.
- Quando l'utente invia un comando, la shell lo scandisce alla ricerca di metacaratteri che processa in modo speciale
- Esempio:

```
user> ls *.java
```

```
Albero.java      div.java      ProvaAlbero.java  
AreaTriangolo.java  EasyIn.java  ProvaAlbero1.java  
AreaTriangolo1.java IntQueue.java
```

Il metacarattere `*` nel pathname è un'abbreviazione per un nome di file. Il pathname `*.java` viene espando dalla shell con tutti i nome di file che terminano con `.java`. Il comando `ls` fronsce la lista di tutti i file con tale estensione

Abbreviazione pathname

- I seguenti metacaratteri, detti wildcard sono usati per abbreviare il nome di un file in un path name:

Metacarattere	Significato
*	stringa di 0 o più caratteri
?	singolo carattere
[]	singolo carattere tra quelli elencati
{ }	stringa tra quelle elencate

Esempi:

```
user> cp /JAVA/Area*.java /JAVA_backup
```

copia tutti i files il cui nome inizia con la stringa Area e termina con l'estensione .java nella directory JAVA_backup.

```
user> ls /dev/tty?  
/dev/ttya /dev/ttyb
```

Abbreviazione pathname

- I seguenti metacaratteri, detti wildcard sono usati per abbreviare il nome di un file in un path name:

... esempi

```
user> ls /dev/tty?[234]
/dev/ttyp2 /dev/ttyp4 /dev/ttyq3 /dev/ttyr2 /dev/ttyr4
/dev/ttyp3 /dev/ttyq2 /dev/ttyq4 /dev/ttyr3
```

```
user> ls /dev/tty?[2-4]
/dev/ttyp2 /dev/ttyp4 /dev/ttyq3 /dev/ttyr2 /dev/ttyr4
/dev/ttyp3 /dev/ttyq2 /dev/ttyq4 /dev/ttyr3
```

```
user> mkdir /user/studenti/rossi/{bin,doc,lib}
crea le directory bin, doc, lib .
```

Quoting

Il meccanismo del **quoting** è utilizzato per inibire l'effetto dei metacaratteri. I metacaratteri a cui è applicato il quoting perdono il loro significato speciale e la shell li tratta come caratteri ordinari.

Ci sono tre meccanismi di quoting:

- il metacarattere di **escape** \ inibisce l'effetto speciale del metacarattere che lo segue:

```
user> cp file file\?  
user> ls file*  
file      file?
```

- tutti i metacaratteri presenti in una stringa racchiusa tra **singoli apici** perdono l'effetto speciale:

```
user> cat 'file*?'  
...
```

- i metacaratteri per l'abbreviazione del pathname presenti in una stringa racchiusa tra **doppi apici** perdono l'effetto speciale (ma non tutti i metacaratteri della shell):

```
user> cat "file*?"
```

Metacaratteri di Shell

Simbolo	Significato	Esempio d'uso
>	Ridirezione dell'output	<code>ls >temp</code>
>>	Ridirezione dell'output (append)	<code>ls >>temp</code>
<	Ridirezione dell'input	<code>wc -l <text</code>
<<delim	ridirezione dell'input da linea di comando (here document)	<code>wc -l <<delim</code>
*	Wildcard: stringa di 0 o più caratteri, ad eccezione del punto (.)	<code>ls *.c</code>
?	Wildcard: un singolo carattere, ad eccezione del punto (.)	<code>ls ?.c</code>
[...]	Wildcard: un singolo carattere tra quelli elencati	<code>ls [a-zA-Z].bak</code>
{...}	Wildcard: le stringhe specificate all'interno delle parentesi	<code>ls {prog,doc}*.txt</code>

Metacaratteri di Shell

Simbolo	Significato	Esempio d'uso
	Pipe	<code>ls more</code>
;	Sequenza di comandi	<code>pwd;ls;cd</code>
	Esecuzione condizionale. Esegue un comando se il precedente fallisce.	<code>cc prog.c echo errore</code>
&&	Esecuzione condizionale. Esegue un comando se il precedente termina con successo.	<code>cc prog.c && a.out</code>
(...)	Raggruppamento di comandi	<code>(date;ls;pwd)>out.txt</code>
#	Introduce un commento	<code>ls # lista di file</code>
\	Fa in modo che la shell non interpreti in modo speciale il carattere che segue.	<code>ls file.*</code>
!	Ripetizione di comandi memorizzati nell'history list	<code>!ls</code>