

The Bash Language

Esecuzione di uno script

- Diretta:
 - o `./scriptname args`
 - Il file `scriptname` deve essere e includere `"#!/bin/bash"` come prima riga
- Indiretta
 - o `source ./scriptname args`
 - È la shell corrente a eseguire lo script

Assegnazione di variabili

`nome_var=valore`

Comando echo (per stampa su stdout)

`echo [OPZIONI] [STRINGA]`

- Opzioni:
 - o `-n`: per non andare a capo.
 - o `-e`: per interpretare i caratteri di escape.

Comando read (per la lettura da stdin)

`read`

- o con una o più variabili passate come argomento
- o uso della variabile `$REPLY`

Quoting

- Single quoting `' '`
 - o le variabili non vengono espanse
- Double quoting `" "`
 - o le variabili vengono espanse
- Es:
 - o `a=pippo`
 - o `echo "$a pippo '$a' pluto"`
 - o `pippo pippo $a pluto`
 - o `echo $a pippo '$a' pluto`
 - o `pippo pippo $a pluto`

Utilizzo delle parentesi { } per delimitare il nome di una variabile

- Es.
 - o `nome=Gian`
 - o `echo ${nome}marco`
 - o `Gianmarco`

Cattura del stdout di un comando

- `$(<comando>)`

Comando exit

- `exit [numero]`
 - o termina l'esecuzione di un processo restituendo un valore al processo chiamante
- Es:
 - o `exit 0`

- o restituisce un valore vero

Esecuzione di calcoli aritmetici

- un metodo a scelta dello studente
- Es.
 - o `let s=$n1+$n2`
 - o Assegna alla variabile \$s la somma di \$n1 e di \$n2

Variabili di shell speciali

- \$0, \$1, \$2, ...
 - o passaggio di parametri sulla riga di comando
- \$*
 - o lista completa dei parametri, escluso il nome dello script
- \$#
 - o Numero di parametri
- \$\$
 - o PID del processo
- \$?
 - o valore di ritorno dell'ultimo processo eseguito

Costrutto if-then-else (e elsif)

```
if condizione ; then
    statements
elif condizione
then
    statements
else
    statements
fi
```

Costrutto while (compresa ridirezione di stdin e stdout)

```
while condizione
do
    statements
done << $fileIn >> $fileOut
```

Formati richiesti nella condizione dei costrutti if e while

Solo le condizioni espresse tra simboli [...] sono richieste (non sono invece richieste le condizioni basate sulla parola chiave test)

- Confronti numerici:
 - o `-eq` uguaglianza (==)
 - o `-ne` non uguaglianza (!=)
 - o `-gt` maggiore (>)
 - o `-ge` maggiore o uguale (>=)
 - o `-lt` minore (<)
 - o `-le` minore o uguale (>=)
- Confronti tra stringhe:
 - o `=` uguaglianza
 - o `!=` non uguaglianza

- Condizioni su file:
 - o `-d <arg>` vero se `<arg>` è un direttorio
 - o `-f <arg>` vero se `<arg>` è un file
 - o `-r <arg>` vero se `<arg>` ha il permesso di lettura
 - o `-w <arg>` vero se `<arg>` ha il permesso di scrittura
 - o `-x <arg>` vero se `<arg>` ha il permesso di esecuzione
- Operatori logici utilizzabili all'interno di una condizione:
 - o `!` not
 - o `-a` and
 - o `-o` or
- Operatori logici utilizzabili in un elenco di condizioni:
 - o `&&` and
 - o `||` or

Costrutto for

```
for var in [ list ]
do
    statements
done
```

Istruzioni

- `break`
- `continue`

Vettori

```
# Dichiarazioni
array[3]="valore"
array=( 4 8 7 )
array=( [0]=4 [1]=8 [2]=7 [5]=10 )
# Accesso
echo ${array[1]} # Accesso all'elemento 1 dell'array (valore 8)
echo ${array[*]} # Stampa tutti gli elementi dell'array
echo ${!array[*]} # Stampa tutte le chiavi dell'array
echo ${#array[*]} # Numero di elementi contenuti nell'array
```

Vettori associativi

```
# Dichiarazioni
declare -A array
array["key"]="value"
array=( [pippo]=hello [2]="pluto" ["pluto"]=2 )
# Accesso
echo ${array[pippo]} # Accesso all'elemento con chiave "pippo"
dell'array (valore "hello")
echo ${array[*]} # Stampa tutti gli elementi dell'array
echo ${!array[*]} # Stampa tutte le chiavi dell'array
echo ${#array[*]} # Numero di elementi contenuti nell'array
```