# GUIDA BASH

**Realizzata da Rusila Marcus**

# Indice

**Cos’è la Bash Shell?** 1
**Cos’è lo shell Scripting?** 2
**Come funzionano gli script Bash?** 2
**Perché usare lo Shell Scripting?** 2
**La prima riga sulla bash shell** 3
**Variabili nella shell** 3
**Le variabili di sistema** 4
**Operatori aritmetico-logici** 5

## Cos’è la Bash Shell?

Il suo termine, bash deriva innanzitutto dal suo autore, Stephen Bourne, evoluzione della standard shell di Unix.
La Bash shell è un’interfaccia a linea di comando che permette l’interazione con SO derivanti da Unix come Mac OS X o molte distribuzioni GNU/Linux. Permette l’utente di comunicare in maniera testuale con il sistema operativo tramite una serie di comandi.

## Cos’è lo shell Scripting?

Lo shell scripting è l’arte di creare e modificare script eseguibili nella shell. La shell, come Bash, consente di lavorare in due modalità:

- Interattiva, dove si esegue un comando alla volta e si ottiene immediatamente un risultato.
- Scripting, dove si scrivono file contenenti una sequenza di comandi da eseguire automaticamente.

Uno script Bash (o semplicemente "script") è un programma scritto nel linguaggio di Bash. Può essere molto semplice, come una lista di comandi, oppure più complesso, includendo:

- Funzioni
- Cicli (for, while)
- Condizioni (if, case)
- Altri costrutti.

# Come funzionano gli script Bash?

Gli script possono essere eseguiti in diversi modi:

- Manuale: richiamando da una sessione interattiva.
- Automatizzato: configurandosi per avviarsi in base a eventi specifici, come:
    - Al momento dell'avvio del sistema.
    - A un orario programmato (es. tramite cron).
    - Ogni volta che un utente si autentica.

# Perché usare lo Shell Scripting?

Lo shell scripting è essenziale per l’amministrazione dei sistemi, perché permette di:

- Automatizzare attività ripetitive: analizzare log, eseguire backup, gestire file e utenti.

- Semplificare processi complessi: combinare più comandi e programmi esterni in un unico script.

- Personalizzare l’ambiente: creare strumenti specifici per esigenze particolari.

# La prima riga sulla bash shell

La prima riga **#!/bin/bash** ha un ruolo cruciale ovvero indica al sistema quale interprete utilizzare per eseguire lo script, /bin/bash è il percorso standard della shell bash sui sistemi Unix.
La scelta di iniziare con #! non è casuale infatti è una funzione speciale di Unix. Anche con l’utilizzo di un’altra shell la presenza dei simboli #! forza l’uso dell’interprete specificato, ovvero Bash.

# Variabili nella shell

La variabile è un nome dato a una porzione di memoria per contenere informazioni o valori, leggerne e manipolare il contenuto.
Le variabili possono essere definite sia dall'utente che dal sistema e tramite il comando **set** possiamo visualizzare la variabile inizializzata e il suo contenuto.
Ecco come inizializzare una variabile:

**nome_variabile=valore**

**Esempio di inizializzazione di una variabile:**

```
#!/bin/bash

location=World
```

Questo script utilizza la variabile location per memorizzare il valore *world*.

Per **visualizzare il contenuto** della variabile utilizziamo il comando **echo**, tramite il simbolo $ che sta a indicare alla shell di sostituire il come della variabile con il suo contenuto

```
#!/bin/bash

location=World

echo "Hello ${location}"
```

Il risultato finale sarà *Hello World.*

# Le variabili di sistema

In Bash, uno script può ricevere argomenti al momento dell'esecuzione, proprio come i comandi standard. Questi argomenti vengono passati allo script e resi disponibili come parametri posizionali, rappresentati da variabili numeriche.

## Parametri posizionali

Gli argomenti forniti a uno script vengono associati a variabili numerate, come:

- $1: Contiene il primo argomento.
- $2: Contiene il secondo argomento.
- ...e così via.

Ad esempio, $1 o ${1} rappresentano il primo argomento passato allo script, e così per gli altri.

Possiamo creare uno script che accetti due argomenti e stampi un messaggio

```
#!/bin/bash
```

```
# Questo script saluta due persone
echo "Hello, $1 and $2!"
```

Eseguendo lo script fornendo due nomi come argomenti il risultato sarà:

*Hello, name1 and name2!*

L'uso degli argomenti permette di adattare lo script a diversi scenari senza dover modificarne il contenuto, inoltre, possono essere utilizzati per passare informazioni dinamiche da altri script o comandi.

# Operatori aritmetico-logici

Le operazioni aritmetico logiche in Bash si ispirano verosimilmente allo stile dei linguaggi di programmazioni derivanti dal C, la loro sintassi infatti è molto simile.
Bash tuttavia utilizza l'**aritmetica intera** non permettendo quindi numeri decimali o frazioni.
Infatti se è possibile fare 9/2 il risultato dato dalla divisione sarà 4 e NON 4.5

Per calcolare il risultato di un'operazione si utilizza la cosiddetta espansione aritmetica, denotata da $(()). Ad esempio, il comando echo $((4 + 4 * (5 - 1) )) stampa 20.

Bash supporta tutti e quattro gli operatori classici +, -, *, / aggiungendo la presenza dell'operatore di modulo % ovvero il resto della divisione.

Esempio:

(10%5 risulta 2)

Possiamo riferirci all'interno di un'espressione aritmetica a delle variabili dichiarate precedentemente nello script senza il bisogno di utilizzare l'espansione di variabile $.

Esempio:

```
i = 4+3

echo $(( 7 * i ))
```

Questo stamperà il valore 49.

Utilizzando la scrittura $i avremmo ottenuto 7*4+3 che risulta 31.

In bash possiamo utilizzare anche operatori detti composti cioè che eseguono operazioni seguite da un assegnamento, ad esempio, (( i *= 3 + 3 )) è equivalente a (( i = i * (3 + 3) )).

Tutti gli operatori complessi:

+=, -=, *=, /= , %=

In bash possiamo usare anche operatori di incremento come ++ e operatori di decremento come - - che rispettivamente aggiungono o tolgono 1 dal valore della variabile

Esempio:

```
i=4
```

```
(( i = i++ ))

echo "$i"

i=5
```

Ecco un esercizio che comprende gli argomenti detti in precedenza: