Bash shell Linux🐧

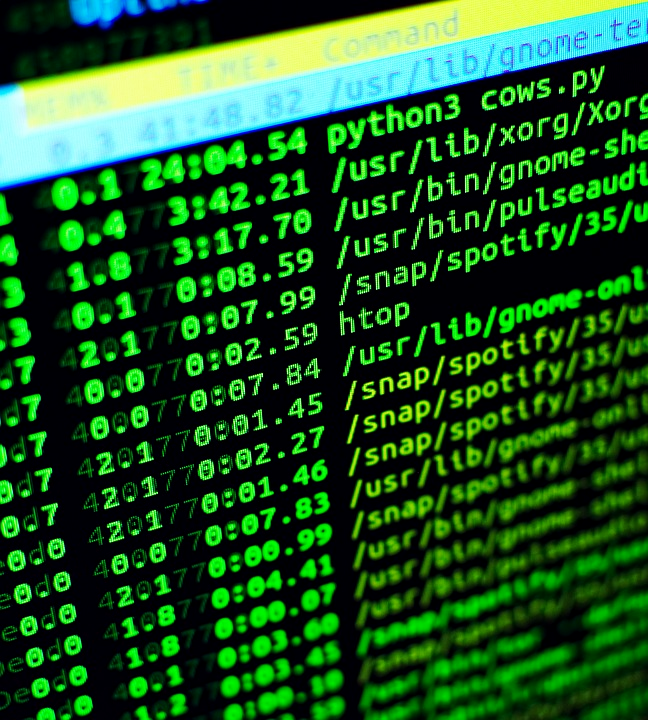
## **Introduzione**

La Bash shell di Linux, è **l’interfaccia** che permette all’utente di interagire tramite linea di comando con il SO Linux. Venne ideata e progettata dal britannico **Brian Fox** nel 1989, in sostituzione alla tradizionale shell Unix e alla Bourne shell. L’intenzione dell’informatico erano non solo di creare una nuova e migliore shell, ma anche completamente **Open Source,** a differenza della Bourne shell che era limitata dalla sua licenza proprietaria.

La novità della Bash shell fu un **progresso nel linguaggio di scripting,** aggiungendo nuove funzionalità moderne tra cui:

* Gestione array.
* Scripting più avanzato: con l’aggiunta di costrutti logici ( &&, ||, regex) e operatori di assegnazione ( +=, ++).
* Completamento automatico, facilitando la stesura del codice.
* Cronologia dei comandi.
* Maggiore compatibilità, in particolare con lo standard POSIX.

Queste ottimizzazioni portarono la Bash shell ad essere una svolta nel mondo informatico, e ad essere usata fino ai giorni nostri.



Una volta che la shell viene lanciata, esegue uno script di partenza, chiamato .bashrc o .bash\_profile situato nella directory home. Il prompt della shell, ovvero quando la shell è pronta a ricevere le istruzioni, è rappresentato dal simbolo del dollaro: $. Se invece eseguiamo la shell come root, o superuser, verrà visualizzato il carattere: #.

In questo modo: [username@host ~]$ [root@host ~]#



←—–Tux rappresenta Linux.

## **Script Bash**

Uno script bash è una serie di comandi scritti in un file, che vengono letti ed eseguiti dal programma bash, riga per riga. La prima riga dello script deve essere obbligatoriamente **#!/bin/bash,** chiamata linea shebang, che indica alla shell il nome dell'interprete di comandi da utilizzare. Per convenzione la loro estensione è .sh

Inoltre possiamo assegnare dei permessi di esecuzione ai script, ad ogni singolo utente, il permesso di esecuzione è rappresentato dal carattere “x”.

### **Eseguire uno script da terminale**

Per eseguire uno script da terminale Ubuntu è necessario creare il file *“nome.sh”*, usare il comando *which bash* e copiare il percorso del file, successivamente aprire il file con un editor di testo, scrivere il commento con il percorso copiato precedentemente e l’operazione che vogliamo eseguire, assegnare i permessi di esecuzione con il comando *chmod u+x nome\_file.sh,* e infine eseguirlo con il comando *bash nome\_file.sh*

### **Debug di uno Script**

Gli errori in cui l’utente può incorrere sono gli stessi di qualsiasi altro linguaggio di programmazione, ovvero: errori di sintassi, errori di runtime, errori logici. Per il debug sono utili le seguenti opzioni durante l’esecuzione:

1. **-n** : Non esegue direttamente lo script, ma verifica gli errori di sintassi.
2. **-u** : Durante l’esecuzione visualizza un messaggio di errore se si tenta di utilizzare una variabile non dichiarata.
3. **-v** : Esegue e visualizza ogni riga di codice.
4. **-x** : Visualizza il valore di ogni variabile durante l’esecuzione.

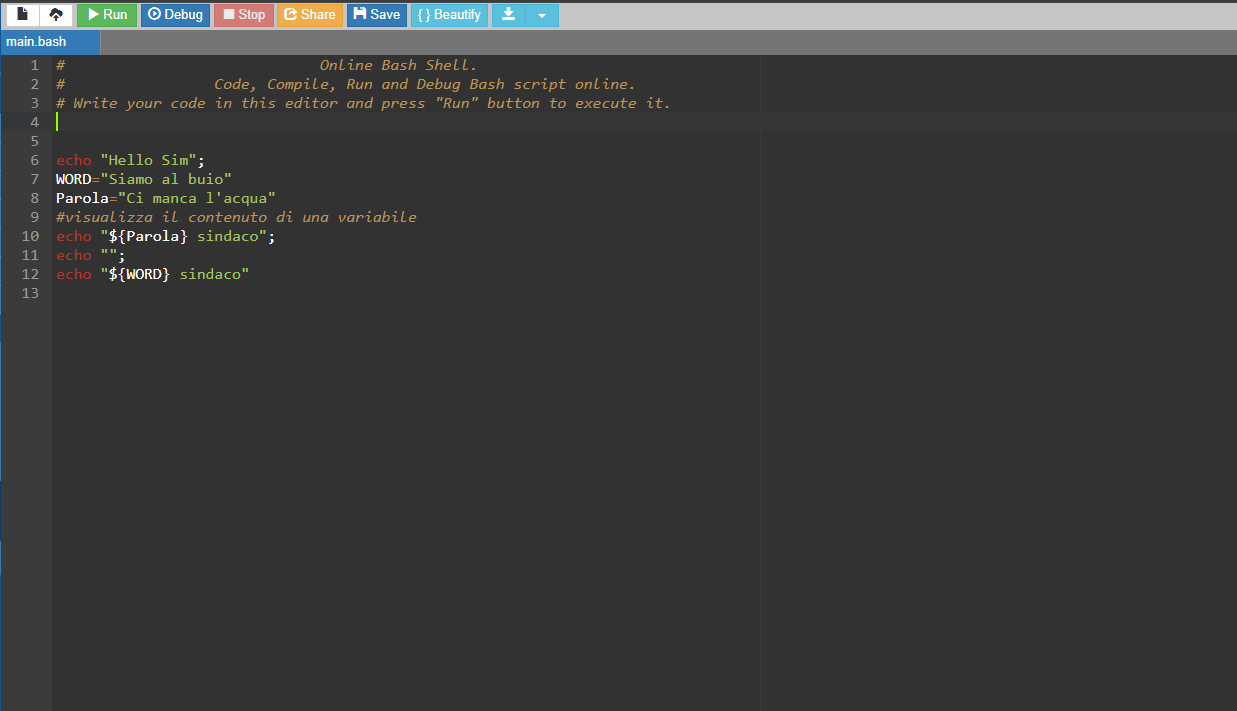
Per implementare queste opzioni è sufficiente aggiungerle nel comando di esecuzione dello script, in questo modo: *bash -n script.sh*

## **Caratteri Speciali**

* **\*** :Indica una qualsiasi stringa.
* **?** :Indica un singolo carattere.
* **[ ]** :
* **~** : Indica la directory home dell’utente.
* **|** : Carattere di Pipeline, l’output di un comando è l’input del comando successivo.
* **&** :
* **;** : Separatore di comandi, permette di inserire più comandi nella stessa riga.
* **&&** :
* **\** :
* **$** : Indica una variabile.
* **#** : La riga viene considerata come un commento

## 

## **Variabili**



Le variabili vengono usate per memorizzare stringhe di caratteri e numeri, e possono essere inizializzate dall’utente o di sistema. Sono locali alla shell in uso, e non possono essere viste da altri comandi o applicazioni. Le variabili utente vengono inizializzate in questo modo: nome\_variabile=valore

Da notare l’assenza di spazi prima e dopo l’uguale, in caso contrario verrebbe visualizzato un errore.

Per visualizzare il contenuto di una variabile utilizziamo il comando:

echo “${nome\_variabile}”;

Il carattere **$** serve a identificare che si tratta di una variabile.

L'inizializzazione degli array avviene attraverso l’uso di parentesi quadrate, che indicano la posizione dell'elemento. **numeri[0]=22**

nome\_array[n]=22

Con il seguente comando possiamo visualizzare ogni elemento dell’array: echo ${numeri[\*]};

Inoltre possiamo modificare la dimensione dell’array, aggiungendo nuovi elementi, e può essere eliminato l’intero array, o un singolo elemento (specificando la posizione), tramite il comando **unset**.

### **Variabili di sistema:**

Alcune variabili predefinite sono le variabili di sistema che rappresentano i

valori di inizializzazione dell'ambiente di lavoro per l'utente.

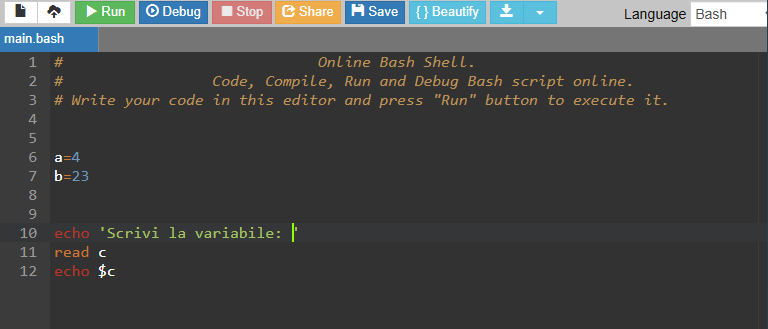
*pwd*

*pat*

*home*

**Lettura input:**

Per leggere valori di variabili in input viene usato il comando read, in questo modo:



## **Operatori aritmetici**

Per assegnare un valore ad una variabile viene utilizzato il comando let,in questo modo:

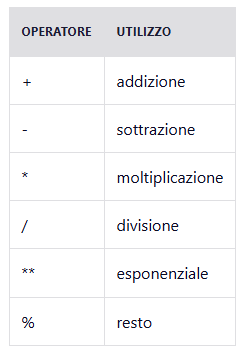
#!/bin/bash

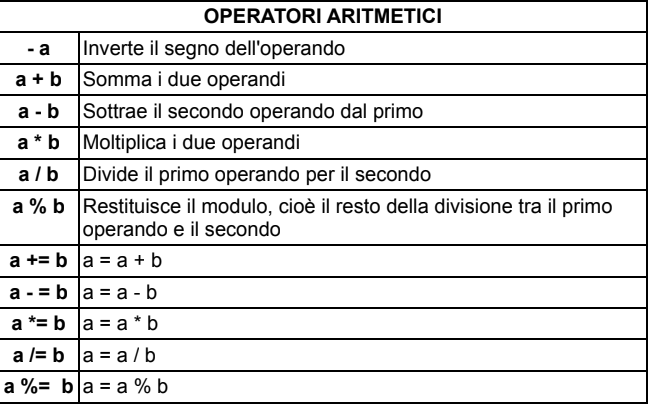
lato\_quadrato=5

let perimetro=lato\_quadrato\*4 #Calcolo perimetro quadrato

echo $perimetro

Sono stati introdotti questi operatori per svolgere l’insieme delle operazioni.

****

****

**Struttura di selezione:**

Viene effettuato attraverso i comandi  **if, then, else, fi.** Dove if indica la prima condizione, then il blocco di comandi da eseguire, else la condizione opposta, e fi la fine della struttura.

if [ condizione ]

then

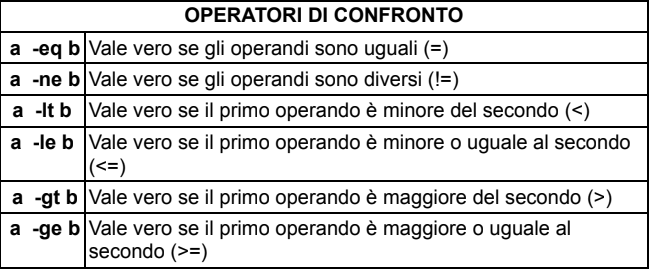
*blocco\_comandi\_di\_then*

else

*blocco\_comandi\_di\_else*

fi

**Operatori di confronto:**

****

**Ciclo for**

Il ciclo for cicla un condizione finché quest’ultima è vera:

for i in 1 2 3 4 5

do

echo "Numero $i"

done

## **Esempi di esercizi**

### **Struttura if**

# Write your code in this editor and press "Run" button to execute it.

echo "Scrivi il tuo voto di TPSIT: "

read voto

echo "Il tuo voto è: "$voto

if [[ "${voto}" -le 10 ]]

then

case $voto in

10)

echo "Hai preso 10, molto bene!";;

9)

echo "Hai preso 9, bene";;

8)

echo"Hai preso 8, bene ";;

7)

echo "Hai preso 7, non male ";;

6)

echo "Hai preso 6, ok ";;

5)

echo "Hai preso 5, poteva andare meglio ";;

4)

echo "Hai preso 4, male! ";;

3)

echo "Hai preso 3, molto male!";;

\*)

echo "Voto non valido!";;

esac

else echo "voto non valido"

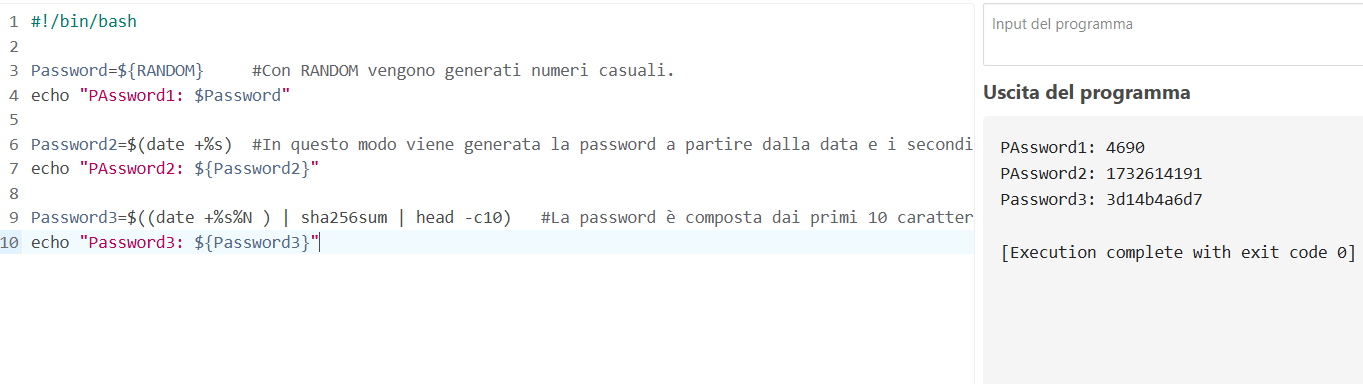
fi

In questo esercizio ho utilizzato una prima struttura if, e successivamente un switch per associare un output diverso per ogni voto.

### **Generazione Password**

Per generare delle Password relativamente complesse e randomiche utilizzeremo varie tecniche:

1. *RANDOM* genera numeri randomici interi da 0 a 32767*.*
2. Con il comando: *date +%s* la Password2 sarà formata dai secondi trascorsi dall’**epoch.**
3. (*date +%s+%N | sha256sum | head -c10*): utilizzando questo comando, i passaggi per ottenere l’output saranno questi:
   * date +%s+%N: otteniamo i secondi (s) e nanosecondi (N) trascorsi dal **epoch**, per esempio 1709180882+123456789.
   * attraverso la pipeline passiamo l’output precedente ( *date*) al comando successivo.
   * Con il comando *sha256sum* otteniamo il checksum, ovvero una stringa da 64 caratteri esadecimali, del comando date.
   * attraverso la pipeline passiamo l’output precedente ( *sha256sum*) al comando successivo.
   * infine tramite *head -c10* prendiamo solo i primi 10 caratteri della stringa, creata dal *sha256sum*



# **Glossario**

Epoch: è il punto di riferimento per moltissimi sistemi Unix, e indica il tempo trascorso da una data arbitraria, il **1 gennaio 1970.** Viene misurata in secondi interi, ed è stata scelta

Shebang: Indica i primi due caratteri dello script: **#!**

Storia Bash shell:

<https://managedserver.it/guida-alle-shell-linux-da-bash-a-bourne-passando-per-zsh-ed-altre/>

Schema riassuntivo:

<http://www.scienze.unitn.it/~fiorella/AppuntiLinux/al-22.html>

Articolo di CodeCamp con esempi di script:

<https://www.freecodecamp.org/italian/news/shell-scripting-per-principianti/>