GUIDA SHELL

-Introduzione alla Shell Bash

La **Bash** (acronimo di *Bourne Again Shell*) è una delle shell più utilizzate nei sistemi operativi Unix, come Linux e macOS. Sviluppata come sostituta della *Bourne Shell* (sh), è ricca di funzionalità che permettono di lavorare in modo efficiente, eseguendo comandi e script .
La BASH SHELL è un interprete di linguaggio di comandi rappresentato con il nome BASH e si trova nella directory **/bin**. Il suo pathname è quindi **/bin/bash**. La shell assegna all'utente una directory personale di lavoro detta home directory.

-Cos'è Bash?

Bash è un interprete di comandi che consente all'utente di interagire con il sistema operativo tramite una **linea di comando**. La shell Bash lavora su un'interfaccia testuale in cui ogni operazione viene eseguita digitando comandi e ricevendo output testuali.

-Cos'è uno 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. Per esempio, puoi portarti in un certo percorso, creare una cartella e generare un processo al suo interno usando la riga di comando. Gli script devono avere come estensione file "**.sh**". Sono anche identificati con uno shebang al loro interno.

-Cos' è la Shebang?

Shebang è la combinazione dei termini inglesi bash # e bang ! seguito dal percorso della shell bash. Questa è la prima riga dello script. Shebang dice alla shell di eseguire lo script tramite la shell bash. Shebang è semplicemente il percorso assoluto dell'interprete bash.

-Definire variabili

Possiamo definire una variabile usando la sintassi "**nomevariabile=valore**".
Per ottenere il valore di una variabile aggiungi **$** davanti al nome della stessa.
esempio: NOMEVARIABILE="Cicetto_Coccoloso"
Per vederla: **echo "${NOMEVARIABILE}"**

-Espressioni algebriche

Qui sono riportate le varie espressioni algebriche:

OPERATORE	UTILIZZO
+	addizione
-	sottrazione
*	moltiplicazione
/	divisione
**	esponenziale
%	resto

-Leggere input da tastiera

Talvolta occorre acquisire un input utente per eseguire certe operazioni.
In bash, possiamo ottenere l'input dall'utente con il comando **read** e l'opzione -p.
read -p "inserisci un nome per la variabile: " NOME_VARIABILE

-Condizioni if e operatori logici

Con le condizioni if andiamo a verificare se una condizione sia vera oppure falsa, andando ad utilizzare gli operatori logici riportati nella tabella sotto.
La sintassi sarebbe la seguente:

```
if [[ condizione ]]; then
        comando
else (se ci fosse il bisogno di un'altra condizione si usa elif)
        comando
fi
```

OPERAZIONE	SINTASSI	DESCRIZIONE
Uguaglianza	num1 -eq num2	num1 è uguale a num2
Maggiore di o uguale a	num1 -ge num2	num1 è maggiore di o uguale a num2
Maggiore di	num1 -gt num2	num1 è maggiore di num2
Minore di o uguale a	num1 -le num2	num1 è minore di o uguale a num2
Minore di	num1 -lt num2	num1 è minore di num2
Non Uguale a	num1 -ne num2	num1 non è uguale a num2

-Ciclo for

Il ciclo for su bash è molto simile a quello usato su java e la sintassi è la seguente:

```
for ((i=0; i<3; i++));
do
        istruzione
done
```

-Switch

Il costrutto switch è perfetto per quando dobbiamo eseguire delle istruzioni in base al valore di una variabile. La sintassi è la seguente:

```
case $VALORE in
        1)
                istruzione
                ;;
        2)
                istruzione
                ;;
        *)                              (in caso il VALORE non sia tra le opzioni)
                istruzione
```

```
        ;;
    esac
```

-Variabili speciali

In bash abbiamo alcune variabili speciali come:

- $0 : da il nome dello script
- $? : il codice di uscita, come quello dato dagli exit (0 è successo, fallimento tutti quelli diversi da 0)
- $1, $2 : danno gli argomenti passati nello script
- $# : dice quanti argomenti sono stati passati
- $@ : mette in fila tutti gli argomenti

-Comandi Exit

Si tratta di comandi che interrompono lo script ma vengono usate in modo diverso:

- Exit 0 indica che lo script si è concluso con successo senza errori
- Exit 1 indica che c'è stato un errore generico e lo script ha incontrato un problema
- Exit 2 indica un errore di input
- Exit 3 indica un errore di connessione

-Esempi esercizi:

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

//In questo esercizio andiamo a creare un nuovo utente per il sistema. Inseriamo nome utente (login), nome reale e una password. Il nome utente, la password e l'host per l'account verranno mostrati.//

```
if [[ "${UID}" -ne 0 ]]
then
   echo 'please run with sudo or as root'
   exit 1
fi
```

//Inserisci il nome utente (login)//

```
read -p 'Enter the username to create: ' USER_NAME
```

//Inserisci il nome reale//

```
read -p 'Enter the name of the person or application that will be
using this account: ' COMMENT
```

//Inserisci la password//

```
read -p 'Enter the password to use for the account: ' PASSWORD
```

//Crea l'account//

```
useradd -c "${COMMENT}" -m ${USER_NAME}
```

//Controlla se il comando "useradd" ha funzionato con successo//

//Non vogliamo dire all'utente che un account è stato creato quando non è vero//

```
if [[ "${?}" -ne 0 ]]
then
   echo 'The account could not be created'
   exit 1
fi
```

//Imposta la password e controlla se la password può essere utilizzata//

```
echo ${PASSWORD} | passwd --stdin ${USER_NAME}
if [[ "${?}" -ne 0 ]]
then
   echo 'The password for the account could not be sent'
   exit 1
fi
```

//Forza la password a cambiare al primo login//

```
password -e ${USER_NAME}
```

//Mostra l'username, la password, l'host e dove è stato creato l'utente//

```
echo
echo 'username:'
echo "${USER_NAME}"
echo
echo 'password:'
echo "${PASSWORD}"
echo
echo 'host:'
echo "${HOSTNAME}"
exit 0
```

---

questo fa il backup di una directory

```
#!/bin/bash
# Script per fare il backup di una directory

origine="/home/utente/documenti"
destinazione="/home/utente/backup"
```

```
if [ ! -d "$destinazione" ]; then
    mkdir -p "$destinazione"
fi

  cp -r "$origine"/* "$destinazione"
echo "Backup completato!"
```

-Creare numeri casuali

```
#!/bin/bash
#PASSWORD=${RANDOM}
#echo "${PASSWORD}${PASSWORD}${PASSWORD}"

#PASSWORD=$(date +%s%N)
#echo "${PASSWORD}"

PASSWORD=$(date +%s%N | sha256sum | head -c10)
#echo "${PASSWORD}"

S_C1=$(echo '!@$%&^*()_-+=' | fold -w1 | shuf |head -c1)
S_C2=$(echo '!@$%&^*()_-+=' | fold -w1 | shuf |head -c1)
echo "${S_C1}${S_C2}${PASSWORD}${S_C2}${S_C1}"
```

sha256sum= genera una stringa alfanumerica
head -c8=massimo 8 cifre
fold -w1=prende un carattere e lo mette su una riga diversa
shuf=mischia le righe
head -c1 = prende la prima riga

PASSWORD=$(date +%s%N | sha256sum | head -c8)
S_C=$(echo '!@$%&^*()_-+=' | fold -w1 | shuf | head -c1)
echo "${PASSWORD}"
echo "${S_C}${PASSWORD}"