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 **$** davanti al nome della stessa.
esempio: NOMEVARIABILE="Coccolino_Coccoloso"
Per vederla: **echo "${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 predefinite

Tra le variabili predefinite troviamo:

- HOME indica la directory home
- USER indica il nome dell'utente corrente
- PWD indica la directory di lavoro corrente
- HOSTNAME nome del computer
- UID indica l'ID utente

-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}"

-Esercizio numeri primi

```
#!/bin/bash
read -p "Inserisci un numero: " n
for ((i=1; i<=n; i++)); do
    primo=1
    if ((i<2)); then
        primo=0
    else
        for ((j=2; j*j<=i; j++));
        do
            if ((i%j==0));
            then
                primo=0
                break
            fi
        done
    fi
    if((primo==1));
    then
        echo "$i è un numero primo"
    else
        echo "$i non è un numero primo"
    fi
done
```

-Esercizio voti con switch

```
#!/bin/bash
read -p "Inserisci un voto (3-10): " voto
case $voto in
    10)
        echo "massimo"
        ;;
    9)
        echo "eccellente"
        ;;
```

```
7 | 8)
    echo "buono"
    ;;
6)
    echo "sufficiente"
    ;;
5)
    echo "insufficiente"
    ;;
3 | 4)
    echo "male male"
    ;;
*)
    echo "voto non valido. Inserisci un voto tra 3 e 10"
    ;;
esac
```

- Inserimento di un nuovo utente

Per aggiungere un nuovo utente si usa il comando **useradd**, ma con privilegi di superuser o root.
L'opzione **-m** crea la directory home per il nuovo utente, **-p** fornisce la password

Per esempio:
$ sudo useradd -m nomeutente -p password

PROBLEMA:
Realizzare uno script che aggiunge un nuovo utente al sistema

SVOLGIMENTO:

1) Per prima cosa verifichiamo che sia eseguito come superuser quindi facciamo il confronto con l'identificatore restituito dal comando **id** che come superuser ha valore 0.
   Scriviamo: if [$(id -u) -eq 0]
2) Nella lettura dell'input da tastiera con il comando **read** con l'opzione **-p** per visualizzare il messaggio e **-s** per non visualizzare i caratteri digitati

quando scriviamo la password. Bisogna inoltre controllare se il nome scelto come username sia già utilizzato nel sistema, verificando se troviamo il nome nel file **/etc/passwd.**
Cerchiamo la stringa nel file con il comando **grep**:
grep -q $username /etc/passwd

3) Dopo il comando grep scriviamo:
if [S? -eq 1]
In questo modo andiamo a verificare che la stringa sia stata trovata, valore 0 indica che la stringa è stata trovata altrimenti si ottiene come valore 1

SCRIPT:

```
#!/bin/bash
if [ (id -i) -eq 0 ] ; then
        read -p “Inserisci username: “ username
        read -s -p “Inserisci password: “ password
        grep -q $username /etc/passwd
        if [ $? -eq 1 ]; then
                useradd -m “$username” -p “$password”
                [ $? -eq 0 ] && echo  -e “\nutente aggiunto” || echo -e “n\No”
        else
                echo -e “\nUsername $username già presente!”
                exit 1
        fi
else
        echo “Attenzione! Solo root può aggiungere un utente al sistema”
        exit 2
fi
```

- Creazioni di archivi compressi

In Linux, il comando **tar** (**t**ape **ar**chive) si usa per creare un archivio compresso oppure per estrarre il contenuto dell’archivio.

Per esempio:
tar [options] [archive-file] [file or directory to be archived]
Vi sono svariate opzioni, alcune di esse sono:

- c: crea un archivio;

- x: estrae il contenuto dell'archivio;
- f: crea un archivio con un dato nome;
- t: visualizza l'elenco dei file contenuti nell'archivio;
- z: zip, indica al comando tar di creare il file compresso usando gzip.

PROBLEMA:
Realizzare uno script di shell per effettuare il backup della directory home di un utente creando un archivio compresso con tutte le directory e i file. Il file ottenuto dovrà essere memorizzato nella directory /tmp.

SVOLGIMENTO:
Innanzitutto andiamo a creare il file userbackup.sh con l'editor nano. L'archivio verrà creato grazie al comando tar con le opzioni c, z e f. Inoltre per rendere generico lo script e non legato ad un determinato utente, utilizziamo la variabile user in questo modo:

```
user=$(whoami)
```

(Questa azione si chiama **sostituzione del comando**, ovvero quando l'output di un comando viene assegnato direttamente ad una variabile)
La variabile user è presente anche nella formazione del nome del file compresso, tuttavia in quest'ultimo è inserita tra parentesi graffe perché dopo ci sono altri caratteri che non fanno parte della variabile

```
file=/tmp/${user}_home_$(date +%Y-%m-%d-%H%m%S).tar.gz
```

Il comando tar spesso restituisce questo messaggio:

```
Removing leading '/' from member names
```

Non si tratta di un errore, difatti l'operazione termina con successo, tuttavia il comando tar rimuove lo / iniziale per permettere, durante la decompressione del file di ritrovare la directory e i file in una posizione relativa, ovvero nella directory nella quale siamo in quel momento.
Per evitare di visualizzare questo messaggio verso il #null device, un file virtuale che non esiste e scarta tutti i dati riceventi, in modo da rendere la riga di comando tar:

```
tar -czf $file $inputdir 2> /dev/null
```

dove il 2 indica lo standard error.

SCRIPT:

```
#!/bin/bash
user=$(whoami)
inputdir=/home/$user
```

file=/tmp/${user}_home_$(date +%Y-%m-%d-%H%m%S).tar.gz

tar -czf $file $inputdir
echo “Backup di %inputdir competato. Dettagli sul file di backup creato:”
ls -l $file

- L’esecuzione di azioni pianificate

Una delle solite attività di amministrazione del sistema consiste nel pianificare l’esecuzione di una data azione di un determinato orario e per un dato numero di volte.

Il CRON è un utility di sistema che permette di eseguire un’azione pianificata.

All’avvio del sistema viene avviato il cron che legge il file di configurazione CRONTAB in cui sono memorizzate le informazioni per programmare ogni singola azione da eseguire.
Si può crearne uno proprio con il comando seguente ;

$ crontab -e.

Il comando può essere utilizzato anche per modificare uno già esistente

Il file cro ntab contiene le informazioni per pianificare un’azione. I primi cinque campi specificano i tempi a seguire c’è il comando da eseguire.
Il valore * significa nel campo dom indica tutti i giorni del mese

Campi	**Valore**
Minuti (*m)*	Da 0 a 59
Ore(*h*)	Da 0 a 23
Giorno del mese(*dom*)	Da 1 a 31
Mese(*mon*)	Da 1 a 12
Giorno della settimana(dow)	Da 0 (domenica) a 6 (sabato)

Esempi:

- Lo script deve essere eseguito ogni notte alle due e mezza:
30 02 * * * / path/myscript.sh

-Lo script deve essere eseguito il giorno 15 di ogni mese alle ore 20:
0 20 15 * * /path/myscript.sh