Guida bash

La programmazione **Bash** (Bourne Again Shell) è un linguaggio di scripting utilizzato principalmente per automatizzare operazioni in ambienti Unix-like, come Linux e macOS. Bash è una shell, ovvero un'interfaccia a linea di comando che permette di eseguire comandi e script. È particolarmente utile per interagire con il sistema operativo tramite terminale, ma offre anche funzionalità di programmazione come variabili, cicli e condizionali.

### **1. Comandi di Base**

Bash consente di eseguire comandi di sistema come ls, cd, echo, cat, cp, mv, rm, e così via. Questi comandi sono la base per qualsiasi operazione in Bash, e vengono usati anche all'interno degli script.

### **2. Script Bash**

Un **script Bash** è un file di testo che contiene una sequenza di comandi Bash. Si crea un file con estensione **.sh** (anche se non è obbligatorio), si rendono eseguibili i permessi del file e si esegue direttamente.

Per esempio:

bash

#!/bin/bash

echo "Ciao, mondo!"

La prima riga (**#!/bin/bash**) è chiamato "shebang" e indica al sistema che il file deve essere eseguito con Bash.

### **3. Variabili**

Le variabili in Bash non richiedono dichiarazioni di tipo. Si assegnano semplicemente con un segno di uguale:

bash

nome="Mario"

echo "Ciao, $nome"

Nota: non ci sono spazi tra il nome della variabile, il segno di uguale e il valore. Per usare una variabile nel comando, si usa il simbolo $.

**3.1 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

### **4. Condizioni (if-else)**

Le condizioni in Bash funzionano in modo simile ad altri linguaggi di programmazione. Si utilizza l'istruzione if per verificare una condizione:

numero=10

if [[ $numero -gt 5 ]]; then

echo "Il numero è maggiore di 5"

else

echo "Il numero è minore o uguale a 5"

fi

### **5. Ciclo for in Bash**

Il ciclo for in Bash viene utilizzato per ripetere una serie di comandi un numero definito di volte. La sintassi di base è la seguente:

for ((i=0; i<3; i++));

do

istruzione

done

**6. Esempio di codice con commenti**

In questo pezzo di codice possiamo vedere un for è un if con lo scopo di vedere se un numero è primo o meno.Il numero entra, nel ciclo l’unica differenza con altri codici di programmazione è l’uso delle doppie parentesi tonde.

#!/bin/bash

V=14

C=0

for((i=2;i<=V;i++)) ; do

if [[ $((V % i)) -eq 0 ]] ; then

C=1;

break

fi

done

if [ $C -eq 1 ]; then

echo "Il numero è primo"

else

echo "Il numero non è primo"

fi

**7. 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 ;;

\*)

istruzione ;;

esac

esempio di esercizio con dei voti

#!/bin/bash

echo "Hello World!"

V=5

case $V in

1)

echo "Hai preso un bruttissimo voto";;

2)

echo "Hai preso un bruttissimo voto";;

3)

echo "Almeno non hai preso 2";;

4)

echo "Almeno non hai preso 3";;

5)

echo "Sei quasi arrivato alla sufficienza ";;

6)

echo "Con un po di impegno puoi prendere di più";;

7)

echo "Ok";;

8)

echo "Bel voto";;

9)

echo "Sei quasi alla perfezione";;

10)

echo “La perfezione ";;

esac

**8. Codice login**

#!/bin/bash

#

# This script creates a new user on the local sytem.

# You will be prompted to enter the username (login), the person name and a password.

# The username, password, and host for the account will be displayed.

# Make sure the script is being executed with superuser privileges.

if [[ "${UID}" -ne 0 ]]

then

echo 'please run with sudo or as root'

exit 1

fi

# Get the username (login)

read -p 'Enter the username to create: ' USER\_NAME

# Get the real name (content for the description field)

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

# Get the password

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

# Create the account

useradd -c "${COMMENT}" -m ${USER\_NAME}

# Check to see if the useradd command succeeded

# We don't want to tell the user that an account was created when it hasn't been

if [[ "${?}" -ne 0 ]]

then

echo 'The account could ot be created'

exit 1

fi

# Set the password

echo ${PASSWORD} | passwd --stdin ${USER\_NAME}

if [[ "${?}" -ne 0 ]]

then

echo 'The password for the account could not be sent'

exit 1

fi

# Force password change on first login

password -e ${USER\_NAME}

# Display the username, password, and the host where the user vas created

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!"

**9. Creazione di sequenze casuali**

Qui creiamo una password casuale prima con un numero randomico poi prendiamo la data in nanosecondi , poi usiamo dei numeri per controllare i file se sono uguali in questo caso sha256sum e infine uniamo le varie soluzioni aggiungendo 1 carattere speciale randomico all'inizio e alla fine della password

!/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}"