Guida su bourne shell (bash) di linux.

#!/bin/bash

# Questo script pulisce il terminal2, mostra un saluto e fornisce informazioni

# sugli utenti attualmente connessi. Le due variabili d'esempio vengono

# impostate e mostrate.

celar # pulisce la finestra di terminale

echo "Lo script inizia adesso."

echo "Ciao, $USER!" # Il segno dollaro si usa per ottenere il contenuto di una

#variabile

echo

echo "Ora ti invio un elenco degli utenti connessi:"

echo

w # mostra chi è collegato e

echo # cosa sta facendo

echo "Adesso sto impostando due variabili."

COLORE="nero" # imposta una variabile locale di shell

VALORE="9" # imposta una variabile locale di shell

echo "Questa è una stringa: $COLORE" # mostra il contenuto della variabile

echo "E questo è un numero: $VALORE" # mostra il contenuto della variabile

echo

echo "Ora ti restituisco l'invito."

Echo

env #Visualizza tutte le variabili d’ambiente di sistema

**Quali sono le variabili di ambiente su Linux Le principali variabili di ambiente sul s.o. Linux sono le seguenti**:

HOME la directory dell'utente

EDITOR il programma di default per editare i testi

PWD la directory corrente

OLDPWD la directory precedente

PATH le directory dove cercare i comandi

PS1 il prompt del terminale

LOGNAME il nome di login dell'utente

USER il nome dell'utente

**Per eliminare una variabile:**

unset [ nome variabile ]

Per lavorare su variabili numeriche con operazioni di tipo aritmetico si usa il comando:

(expr espressione) che restituisce il valore dell’espressione aritmetica

Es:

numero=9

numero1=$(expr $numero + 1 ) #oppure numero1=`expr $numero + 1`

echo $numero1 #Il risultato è 10

**Istruzioni di controllo: IF**

Una cosa molto importante da ricordare è che le condizioni vanno sempre inserite all'interno delle parentesi quadre **[** e **]** avendo cura di lasciare uno spazio bianco subito dopo l'apertura e subito prima della chiusura delle parentesi.

Partiamo da un esempio: poniamo di voler porre una domanda e di stampare a video un messaggio a seconda che la risposta sia corretta o meno:

#!/bin/sh

#Faccio la mia domanda

echo -n "Qual'è il nome di Garibaldi? "

#Recupero la risposta

read RISPOSTA

#Stampo a video il risultato

if [ ${RISPOSTA} = "giuseppe" ]

then

echo "Risposta esatta"

else

echo "Risposta sbagliata"

fi

Il costrutto if ha una sintassi molto semplice: si esprime la condizione e se essa risulta vera la shell esegue le istruzioni subito dopo **then**. In caso contrario è possibile specificare delle istruzioni opzionali alternative subito dopo **else**. Ogni costrutto if termina con **fi**.

Da notare che dopo la parentesi siamo andati a capo... ma avremmo anche potuto scrivere così:

if [ condizione ]; then

Invece di andare a capo, quindi, si aggiunge un punto e virgola alla chiusura della parenresi quadra. Il punto e virgola ha infatti la funzione di separatore di comandi che quindi possono essere così mesi silla stessa riga.

Nel caso in cui ci siano **condizioni multiple** è possibile usare **elif** al posto dei vari if ed esprimere una sola volta le istruzioni alterntative con else alla fine. Facciamo un altro esempio:

#!/bin/sh

#Faccio la mia domanda

echo -n "Quanti anni hai? "

#Recupero la risposta

read ANNI

#Stampo a video il risultato

if [ ${ANNI} -lt 10 ]

then

echo "Sei un bambino"

elif [ ${ANNI} -lt 18 ]

echo "Sei minorenne"

else

echo "Sei maggiorenne"

fi

Strutture iterative:

**Il ciclo for**

Il ciclo **FOR** crea un iterazione sulla base di una data lista di valori o di una data condizione. Vediamo un esempio:

#!/bin/sh

for X in 1 2 3 4 5

do

echo $X

done

In questo caso il ciclo for si ripete per ogni valore specificato (nel nostro caso verranno stampati i numeri da 1 a 5, ma avremmo potuto usare anche valori non sequenziali oppure non numerici).

Altri esempi:

for (( X=0; X<10; X++ ))

do

echo $X

done

Oppure:

for X in $(seq 5)

do

echo $X

done

**Il ciclo while**

Altro modo di creare iterazioni all'interno dei nostri script è l'utilizzo di **WHILE**. Grazie a while si ripetono determinate azioni fino a quando la condizione limite risulta soddisfatta.

#!/bin/sh

X=1

while [ $X -le 5 ]

do

echo $X

X=`expr $X + 1` #oppure X=$(expr $X + 1)

done

Nel nostro caso il ciclo continerà fino a quando la condizione limite (X minore o uguale a 5) risulta vera stampando, quindi, una sequenza numerica da 1 a 5.

**break e continue**

Due comandi molto importanti nella gestione dei loop sono **break** e **continue**.

Il comando *break* serve per interrompere il ciclo in modo arbitrario. Vediamo un esempio di ciclo *while* interrotto con *break*:

#!/bin/sh

X=1

while [ $X -le 20 ]

do

echo $X

if [ $X -eq 5 ]

then

break

fi

X=`expr $X + 1`

done

Nel nostro esempio il ciclo verrà interrotto quando la nostra variabile contatore avrà raggiunto il valore di 5.

Attraverso *break* è anche possibile interrompere più cicli annidiati:

#!/bin/sh

for X1 in 1 2 3

do

for X2 in 0 5

do

if [ $X1 -eq 2 -a $X2 -eq 0 ]

then

break 2

else

echo "$X1 $X2";

fi

done

done

In questo esempio abbiamo introdotto una doppia condizione, quando verificata questa porterà allo stop di entrambi i cicli, per farlo si usa la sintassi:

break n

Dove "n" indica il ciclo da cui si desidera uscire (n deve essere maggiore o uguale di 1).

Mentre *break* interrompe il ciclo, il comando *continue* lo fa proseguire semplicemente saltando l'iterazione corrente. Il funzionamento e la sintassi sono i medesimi visti per *break*.

**Operatori Matematici**  
Si utilizzano solo ed esclusivamente con valori numerici

|  |  |
| --- | --- |
| + | Somma |
| - | Sottrazione |
| \\* | Moltiplicazione |
| / | Divisione |
| \*\* | Esponenzia il primo numero al secondo |
| % | Ritorna il resto di una divisione |

Per effettuare operazioni matematiche sulle variabili è necessario utilizzare alternativamente:

* il comando esterno **expr** tra apici gravi
* le doppie parentesi tonde **((** e **))**
* il costrutto **let**

Vediamo qualche esempio:

#!/bin/sh

#Operazioni matematiche con numeri interi

#Definiamo due variabili numeriche

X=10

Y=4

#Uso di expr tra apici gravi

R1=`expr $X - $Y`

#Uso le doppie parentesi

R2=$((X-Y))

#Uso di let

let "R3 = X - Y"

#Stampiamo a video i risultati

echo "risultato di expr: "$R1

echo "risultato delle dopie parentesi: "$R2

echo "risultato di let: "$R3

Da ricordare che expr funziona solo con i numeri interi. Se volete lavorare con numeri decimali utilizzate il comando bc (binary calculator) in questo modo:

echo "2.11 + 8.7" | bc

Riprendiamo lo script matematico visto sopra ponendo che le due variabili X e Y siano numeri decimali:

#!/bin/sh

#Operazioni matematiche con numeri decimali

X=7.3

Y=4.22

Z=`echo "$X + $Y" | bc`

echo $Z

**Operatori di Confronto (Numerico)**  
Si utilizzano solo ed esclusivamente con valori numerici

|  |  |
| --- | --- |
| -eq | Equivalenza |
| -ne | Disuguaglianza |
| -gt | Maggiore |
| -ge | Maggiore o Uguale |
| -lt | Minore |
| -le | Minore o Uguale |

**Operatori di Confronto (Stringhe)**  
Si utilizzano solo ed esclusivamente con variabili stringa

|  |  |
| --- | --- |
| -n | Verifica che la stringa sia di lunghezza maggiore di zero |
| -z | Verifica che la stringa abbia lunghezza zero |
| = | Verifica che due stringhe siano uguali |
| != | Verifica che due stringhe siano diverse |
| < | Verifica che la prima stringa sia lessicograficamente minore della seconda |
| > | Verifica che la prima stringa sia lessicograficamente maggiore della seconda |

**Operatori Logici**

|  |  |
| --- | --- |
| ! | Operatore di negazione, inverte il valore logico dell'espressione al quale viene applicato. !(vero) = falso |
| -a | Operatore AND. Ritorna vero se entrambi gli argomenti passati lo sono. |
| -o | Operatore OR. Ritorna vero se almeno uno degli argomenti lo è. |