



Introduzione al bash scripting

Laboratorio di Fondamenti di Informatica
Operatore Informatico Giuridico

Facoltà di Giurisprudenza
Università di Bologna



Definizione

Un bash script è uno strumento usato per creare applicazioni permettendo di utilizzare contemporaneamente chiamate di sistema, strumenti, utility e file binari (eseguibili)

Un vero e proprio linguaggio di progettazione che supporta “nativamente” comandi di sistema senza l’aggiunta di particolari plugin



Classico esempio - Hello World

Creo un file testuale con un qualsiasi editor di testo

Scrivo la seguente sintassi:

```
#!/bin/bash  
echo "Ciao Mondo!!!"  
exit
```

Salvo ed esco dall'editor

Do i permessi di esecuzione al file creato

Eseguo il file con "sh nomefile" o "./nomefile"

Consigliato l'uso di ./nomefile



Primo codice

```
#!/bin/bash  
# script che esegue un file  
# Da eseguire come root, naturalmente.
```

```
programma=/home/steve/  
cd $programma
```

```
sh pippo
```

```
echo "Programma eseguito"
```

```
exit
```



Primo codice

`#!/bin/bash`: usato SEMPRE per iniziare uno script

`#`: tutti i caratteri dopo questo simbolo sono commenti che non verranno eseguiti al momento dell'esecuzione dello script

`echo " "`: comando usato per visualizzare caratteri sullo schermo

`programma=/home/steve` : sintassi usata per la dichiarazione di una variabile

`exit`: sintassi utilizzate per terminare uno script



Variabili

Le variabili rappresentano il modo in cui i linguaggi di scripting e di programmazione identificano i dati.

Compaiono nelle operazioni aritmetiche, nelle manipolazioni quantitative e nelle verifiche di stringhe.

Una variabile non è nient'altro che un'etichetta assegnata a una locazione, o a una serie di locazioni, di memoria del computer che contiene un dato.



Variabili

Esempio di sintassi per l'assegnamento di un valore ad una variabile:

```
percorso=/home/steve
```

```
valore1=9
```

```
variabile_multipla="contiene più valori"
```

Quando si assegna il valore ad una variabile, non sono consentiti spazi prima e dopo l'uguale.

```
echo "$valore1" -> visualizza il valore della variabile
```

```
varolex="" -> variabile vuota
```

```
21=$valore1 -> modifica il valore della variabile "valore1"
```



Costrutti condizionali

Il costrutto if/then verifica se l'exit status di un elenco di comandi è 0 (se è zero, la verifica è verificata), può essere utilizzato con qualsiasi comando, o condizione compresa tra le parentesi quadre

```
if [ qualcosa ]  
then  
  comando  
else  
  comando  
fi
```



Costrutti condizionali

```
if cmp pippo nomefile &> /dev/null # non visualizza l'output
then echo "I file pippo e nomefile sono identici."
else echo "I file pippo e nomefile sono diversi."
fi
```

```
if [ `pwd` != "dir-programma" ]
then echo "non sono in $dir-programma"
else sh pippo
fi
```



Costrutti condizionali

È possibile annidare i costrutti if/then

Esempio:

```
if pippo = pluto
```

```
then
```

```
if [ "$UID" = 0 ]
```

```
echo "pippo è uguale a pluto e sei root"
```

```
else
```

```
echo "le condizioni non sono soddisfatte"
```

```
fi
```



Operatori di confronto

Confronto di stringhe:

- = è uguale a - `if ["$a" = "$b"]`
- != è diverso da - `if ["$a" != "$b"]`
- \`<` è inferiore a - `if ["$a" \< "$b"]`
- \`>` è maggiore di - `if ["$a" \> "$b"]`
- -z la stringa ha lunghezza zero
- -n la stringa non ha lunghezza zero



Operatori di confronto

Confronto di interi:

- `-eq` è uguale a - `if ["$a" -eq "$b"]`
- `-ne` è diverso da - `if ["$a" -ne "$b"]`
- `-gt` è maggiore di - `if ["$a" -gt "$b"]`
- `-ge` è $> 0 = a$ - `if ["$a" -ge "$b"]`
- `-lt` è minore di - `if ["$a" -lt "$b"]`
- `-le` è $< 0 = a$ - `if ["$a" -le "$b"]`



Primo codice - controlli

```
#!/bin/bash
# script che esegue un file
# Da eseguire come root, naturalmente.
dir-programma=/home/students/s/sfratepi
if [ `pwd` != "dir-programma" ]
then echo "non sono in $dir-programma"
else sh pippo

echo "Programma eseguito"

fi
exit
```



Primo codice - controlli

```
#!/bin/bash
# sei root?
userroot=0
path=/home/steve
if [ "$UID" = "$userroot" ]; then
    if [ "$(pwd)" == "$path" ]; then
        echo "tutto ok"
    else
        echo "non sei in $path"
    fi
else
    echo "no root"
fi
exit
```



Cicli - for

Un ciclo è un blocco di codice che ripete un certo numero di comandi finché la condizione di controllo del ciclo definita è vera.

```
for qualcosa in qualcosaltro  
do  
  comando(c)  
done
```

qualcosaltro può essere una variabile o una lista



Cicli - while

Il ciclo While verifica una condizione definita all'inizio del ciclo; il ciclo viene mantenuto in esecuzione finché quella condizione rimane vera.

Il ciclo while viene usato in situazioni in cui il numero delle iterazioni non è conosciuto in anticipo.

```
while [condizione]  
do  
  comando  
done
```



Cicli - while

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

```
var0=0
```

```
LIMITE=10
```

```
while [ "$var0" -lt "$LIMITE" ]
```

```
do
```

```
var0=`expr $var0 + 1`
```

```
done
```

```
exit
```



Cicli - until

Until verifica una condizione definita all'inizio del ciclo; il ciclo viene mantenuto in esecuzione finché quella condizione rimane falsa (l'esatto contrario del ciclo while)

```
until [condizione-falsa]  
do  
comando  
done
```



Cicli - until

```
1 #!/bin/bash 2
```

```
CONDIZIONE_CONCLUSIONE=fine
```

```
until [ "$var1" = "$CONDIZIONE_CONCLUSIONE" ]
```

```
do
```

```
echo "Immetti la variabile 1 "
```

```
echo "($CONDIZIONE_CONCLUSIONE per terminare)"
```

```
read var1
```

```
echo "variabile 1= $var1"
```

```
done
```

```
exit 0
```



Riferimenti

- <http://it.wikipedia.org/wiki/>
- <http://www.pluto.it/files/ildp/guide/abs/index.html>



Introduzione al bash scripting

DOMANDE???

