

Scripting di Shell

Laboratorio di
Amministratore di Sistema

Impostiamo la password di root

```
xxx@yyy:~$  
~$ su  
Password:  
su: Authentication failure
```

non necessariamente

```
xxx@yyy:~$ sudo passwd root  
Enter new UNIX password:  
Retype new UNIX password:  
passwd: password updated successfully  
xxx@yyy:~$ su  
Password:  
root@yyy:/home/xxx#  
exit
```

Aggiungiamoci come nuovo utente

```
xxx@yy:~$ su
Password:
root@yy:/# adduser alice
Adding user 'alice' (1004) ...
Adding new group 'alice' (1005) with group 'alice'
The home directory '/home/alice' already exists. Not copied
from '/etc/skel'.
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for alice
Enter the new value, or press ENTER for the default
  Full Name []: AliceFullName
  Room Number []: 1
  Work Phone []: 123
  Home Phone []: 123456
  Other []: aaa
Is the information correct? [y/N]
root@yy:/# login alice
```

non necessariamente

Riassunto comandi essenziali

ls	[ls -la] Long directory listing.
echo	[echo "foo"] Does what it says.
cat	[cat /etc/passwd] Dump out the content of a file.
less	[less /etc/passwd] Scroll up/down in a file (q = exit)
head	[head -5 /etc/passwd] Get the 5 first lines of a file.
tail	[tail -7 /etc/passwd] Get the 7 last lines of a file.
grep	[grep x /etc] Dump lines containing x from /etc
chmod	[chmod a+x file] Give everyone executable rights to file
chown	[chown root file] Change owner of file to root.
cd	[cd /etc] Change Directory to /etc

Primo script

```
$ mkdir ~/shell
$ cd ~/shell
$ nano script1
```

```
#!/bin/bash
# nome file: script1
# crea una directory, la rinomina e poi la elimina
# accompagnando le varie fasi con messaggi di traccia
#
mkdir $HOME/shell/niente
echo "Ho creato una directory"
mv $HOME/shell/niente nulla
echo "Ho rinominato la directory"
rmdir $HOME/shell/nulla
echo "Ho eliminato la directory"                                ctrl O + ctrl X
```

Il Path

- Perché non funziona il comando `$ script1 ?`
- Perché il percorso di ricerca dei comandi (PATH) non include la directory corrente – (echo \$PATH) –
- È quindi sufficiente:

```
$ export PATH="$PATH:."
```
- per poter mandare in esecuzione gli script con il solo loro nome (se la directory corrente li contiene!)
oppure
- aggiungere il percorso `./` prima del nome dello script

Primo script

- Ora mando in esecuzione lo script:

```
$ ./script1
```

- ma ottengo ancora un messaggio di errore:

```
-bash: script1: Permission denied
```

- ci mancano i diritti di esecuzione: (ls -all)

```
$ chmod a+rx script1 (ls -all)
```

- Ora funziona.

Secondo script

```
$ nano script2
```

```
#!/bin/bash
# nome file: script2
# gestione delle variabili (testo, numeri, comandi)
# e lettura dalla tastiera
#
txt="variabile di testo"
echo $txt
num=123,5
echo $num
cmd="ls -al"
echo $cmd
echo "Come ti chiami?"
read nome
echo "Ciao, "$nome.
```

Attenzione:
niente spazi




appare il
listato?

Come fare per
farlo apparire?

\$... = il valore di ...

Assegnazione variabili

variabile=valore

- valore viene considerato come una stringa
- 'val ore' ammessi anche spazi 
- "val ore" valutata e sostituita con il risultato 
- `valore` restituisce l'output del comando 
- \$variabile restituisce il valore della variabile
- \${variabile} restituisce il valore della variabile
- "espressione con variabili" valutata
- \$variabile= assegna null a variabile
- \${variabile=default} se la variabile non è ancora stata settata, il suo valore è default
- \$[espressione] restituisce la stringa risultato dell'espress.
- \$((espressione)) come sopra (con calcolatrice)

' = \039

" = \034

` = \096

Riassumendo

script

```
#!/bin/bash
# nome file: esempio
var1=10 ; var2=20
echo valore
echo 'val ore'
echo "val ore"
echo $var1
echo '$var1 + $var2'
echo "$var1 + $var2"
echo ${ var1 + var2 }
echo ${var1=1}
echo ${33+16}
echo $(( 33 - 16 ))
```



output

```
valore
val ore
val ore
10
$var1 + $var2
10 + 20
30
10
49
17
```

Terzo script

- I vettori:

```
#!/bin/bash
# nome file: script3
# utilizzo di variabili vettoriali
vet=(11 12 13 14)
echo "Tutto il vettore: "${vet[*]}
echo "vet[3] ="${vet[3]}
echo "Il vettore è lungo "${#vet[*]}
vet[4]=15
echo "Ho aggiunto un valore"
echo "ora il vettore è lungo "${#vet[*]}
```

e se aggiungo
spazi?

e se aggiungo
vet[45]?

Quanto è ora
la lunghezza
totale?

- I vettori cominciano dall'indice == 0.
- I vettori in realtà sono un elenco di variabili.

Quarto script

- Gli argomenti della linea di comando:

```
#!/bin/bash
# nome file: script4 X Y
# utilizzo degli argomenti di linea di comando
# $0 = nome dello script in esecuzione
# $1 = valore del primo argomento
# $# = numero argomenti della linea di comando
# $* = tutti gli argomenti della linea di comando
#
echo Ho inserito $# argomenti: $*
echo "Il primo vale = $1"
echo 'Il secondo vale '$2
```

Quinto script

- Gli operatori aritmetici

```
#!/bin/bash
# nome file: script5 X Y
# utilizzo del comando let
# e degli operatori aritmetici
#
let var=$1-$2 ; echo $1 - $2 = $var
let var=$1/$2 ; echo $1 / $2 = $var
let var=$1%2 ; echo $1 % 2 = $var
let var=$1,var+=1 ; echo $1 += 1 = $var
echo $1 * $1 = $(( $1 * $1 ))
```

Sesto script

- Gli operatori di confronto:

```
#!/bin/bash
# nome file: script6 N1 N2
# dove N1 ed N2 sono due attributi numerici
# utilizzo di operatori di confronto numerico
#
if [ $1 -eq $2 ] # comando di selezione
then
    echo "arg1 è uguale a arg2"
elif [ $1 -lt $2 ]
then
    echo "arg1 è minore di arg2"
else
    echo "arg1 è maggiore di arg2"
fi
```

Gli operatori di confronto

numeri

```
A -eq B → A == B?  
A -ne B → A != B?  
A -lt B → A < B?  
A -le B → A <= B ?  
A -gt B → A > B ?  
A -ge B → A >= B ?
```

```
case selettore in  
    val1) comandi1 ;;  
    val2) comandi2 ;;  
    *) comandiDefault ;;  
esac
```

stringhe

```
strA == strB  
strA > strB  
strA < strB  
strA != strB  
-z strA → length(strA)==0  
-n strA → length(strA)!=0
```

boolean

```
! (NOT)  -a (AND)  -o (OR)
```

Cicli reiterativi

```
for elemento in lista  
do  
    istruzioni  
done
```

```
for i in 1234567  
do  
    echo $i  
done
```

```
while [condizione]  
do  
    istruzioni  
done
```

```
while 1234567  
do  
    echo $i  
done
```

```
a=1  
maxA=7  
while [ $a -le $maxA ]  
do  
    echo $a  
    a=$((a+1))  
done
```


Settimo script

- Controllo valori degli argomenti da riga di comando

```
#!/bin/bash
# nome file: script7 N1 N2
# controlla i valori della riga di comando
# compresi tra 1 e 100
i=1
for var in $1 $2
do
    if [ $var -lt 1 -o $var -gt 100 ]
    then
        echo valori num $i non adeguato
    else
        echo valore num $i corretto = $var
    fi
    let i++
done
```

Attenzione agli
spazi!!!

Ottavo script

- Creare uno script che produca una lista numerata dei file della directory corrente che soddisfano ad una stringa-condizione

```
#!/bin/bash
# nome file: script8 condizione
# ad esempio ./script8 "*.txt"
# enumera i file della directory corrente che
# soddisfano alla condizione specificata
#
let i=0
for file in `ls $1`
do
    let i++
    echo $i: $file
done
```

Esercizi

1. Immettere due variabili numeriche da tastiera (non da linea di comando) e visualizzare la più grande
2. Immettere due variabili numeriche da tastiera (non da linea di comando) e confrontarle (max)
3. Come il primo, ma con tre variabili stringa

Soluzione 1

1. Immettere due variabili numeriche da tastiera (non da linea di comando) e visualizzare la più grande

```
#!/bin/bash
# nome file: esercizio1
# massimo tra due numeri
#
echo -n 'Dammi un numero ' ; read num1
echo -n 'Dammi un secondo numero ' : read num2
if [ $num1 -ge $num2 ]
then
    echo Il massimo è $num1
else
    echo Il massimo è $num2
fi
```

Soluzione 2

1. Immettere due variabili numeriche da tastiera (non da linea di comando) e confrontarle (max)

```
#!/bin/bash
# nome file: esercizio2
# confronto di due numeri
#
echo -n 'Dammi un numero ' ; read num1
echo -n 'Dammi un secondo numero ' : read num2
echo -n 'Tra $num1 e $num2 il maggiore è '
if [ $num1 -gt $num2 ]
then echo $num1
elif [ $num2 -gt $num1 ]
then echo $num2
else echo $num1 "(sono uguali!)"
fi
```

Soluzione 3

1. Immettere tre variabili intere da tastiera (non da linea di comando) e visualizzare la più grande

```
#!/bin/bash
# nome file: esercizio3
# ricerca del massimo tra tre valori interi
#
echo -n 'Scrivi un intero: ' ; read num1
echo -n 'Scrivi un secondo intero: ' : read num2
echo -n 'Scrivi un terzo intero: ' : read num3
let max=$num1
if [ $num2 > $max ] ;then
let max=$num2; fi
if [ $num3 > $max ] ; then
let max=$num3; fi
echo 'Il massimo è $max'
```

Backup

- Facciamo il backup della directory /home :

```
#!/bin/bash
# nome file: backup
# effettua il backup compresso della directory /home
#
sorg="/home/"
dest="/var/backup/"
nome=home-$(date +%d-%m-%Y).tgz
tar -czf $dest$nome $sorg
```

Comando multipli

- Diverse funzioni in base alle opzioni :

```
#!/bin/bash
# nome file: multiplo
# esegue funzioni diverse in base all'opzione indicata
# provare con $ ./multiplo
# provare con $ ./multiplo --port 127.0.0.1 21
# provare con $ ./multiplo --time
# provare con $ ./multiplo --help
case $1 in
  --port) telnet $3 $2 ;;
  --time) date ;;
  --help|*)
    echo "Uso: $0 [--help] [--port <port>] host [--time]" ;;
  esac
```

Un ambiente grafico?

```
#!/bin/bash
# nome file: menu1
# utilizzo del comando whiptail
#
if whiptail --yesno "Vuoi proseguire?" 10 30
then
    echo "Bene :-)"
else
    echo "Male ;-( "
fi
```

Diagramma di annotazione: un rettangolo giallo con "altezza" ha una freccia che punta al valore "10" nel comando whiptail. Un altro rettangolo giallo con "larghezza" ha una freccia che punta al valore "30".

Sostituisci i due testi con:

```
whiptail --msgbox "Risposta esatta!" 10 24
whiptail --msgbox "Risposta errata!" 10 24
```

Un ambiente grafico?

```
$ whiptail --title "Messaggio" --msgbox "Ciao mondo" 5 20
```

```
$ whiptail --title "Messaggio" --yesno
  "Pensi di essere fortunato?" 6 25
```

Diagramma di annotazione: un rettangolo giallo con "Cambiare!!" ha una freccia che punta al valore "6" nel comando whiptail.

```
$ whiptail --infobox "Aspetta" 7 30 ; sleep 3
```

```
$ whiptail --inputbox "Come ti chiami?" 8 40 2>tempfile
```

```
$ whiptail --textbox tempfile 2 70
```

by <http://www.linuxjournal.com/article/2807>

Un ambiente grafico?

```
$ whiptail --menu <text> <height> <width> <menu-height>
[<tag><item>]
```

```
#!/bin/bash
# nome file: menu2
# utilizzo del comando whiptail
#
whiptail --title "MENU" \
        --menu "Scegli un'opzione" \
        0 0 0 \
        1 prima \
        2 seconda \
        3 terza \
        4 quarta 2>.tempfile
echo `cat .\tempfile`
```

← altezza larghezza altezza_menu

<OK> → restituisce il <tag> selezionato

<CANCEL> → restituisce nulla

Un ambiente grafico?

```
#!/bin/bash
# nome file: directory
whiptail --title "Scelta" \
        --inputbox "Nome directory:" 8 40 `pwd`\
        2>./tempfile
if [ $? = 1 ]; then # controlla lo stato d'uscita [CANC]
    clear ; exit 0
fi
ris=`cat ./tempfile`
ls -al $ris > ./tempfile
whiptail --title "contenuto di "$ris \
        --textbox ./tempfile 22 75
clear
rm -f ./tempfile
exit 0
```