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
```

Login su server dedicato al corso

putty | ssh

Collegarsi al server
ammsis-2014

Accreditarsi come
login: root
Password: ammsis2014

CTRL – ALT - T

Aggiungiamoci come nuovo utente

Per vedere tutti gli utenti esistenti:

```
xxx@yyy:~$ cat \etc\passwd
          oppure
xxx@yyy:~$ ls \home
```

per eliminare definitivamente un utente

```
xxx@yyy:~$ sudo userdel -r pippo
```

Aggiungiamoci come nuovo utente

```
xxx@yyy:~$ su
Password:
root@yyy:/# adduser alice
Adding user 'alice' (1004) ...
Adding new group 'alice' (1005) with group 'alice'
The home directory '/home/alice' already exists. Not copyng
from '/etc/skel'.
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated succwssfully
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@yyy:/# login alice
```

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+rwx 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 ?
```

stringhe

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

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

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 $1: $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
```

Addendum: stringhe

lunghezza di una stringa

```
str=pippo53;  echo ${#str}
```

gestione sottostringhe con indice →

${*stringa*:*posizione*}

${*stringa*:*posizione*:*lunghezza*}

```
str=pippo53; echo ${#str}    → 7
echo ${str:0} → pippo53   echo ${str:0:2} → pi
echo ${str:1} → ippo53    echo ${str:1:1} → i
echo ${str:2} → ppo53     echo ${str:2:2} → pp
echo ${str: -1} → 3       echo ${str: -1:2} → 3
echo ${str: -2} → 53      echo ${str: -2:3} → 53
```

Addendum: stringhe

Scrivere un programma che esegua i seguenti comandi su stringhe, ricorrendo ad una interfaccia di testo

```
1. Confronta due stringhe
2. Concatena stringhe
3. Lunghezza di una stringa
4. Scomponi una stringa
5. Inverti una stringa
6. Esci
Cosa scegli (1-6)?
```

```
clear
i="s"
t=0
while [ $i = "s" ]
do
  clear
  echo "1. Confronta due stringhe"
  ...
  echo "Cosa scegli (1-6)?"; read comando
  case $comando in
  esac
  echo "Vuoi continuare (s/n)?"; read i
  if [ $i != "s" ]
  then
    exit
  fi
done
```

soluz. 1/4

```
clear
echo "1. Confronta due stringhe"
echo "2. Concatena due stringhe"
echo "3. Lunghezza di una stringa"
echo "4. Scomponi una stringa"
echo "5. Inverti una stringa"
echo "6. Esci"
echo "Cosa scegli (1-6)?"; read comando
case $comando in
    1)# Confronto
     echo "Prima stringa"; read s1
     echo "Seconda stringa"; read s2
     if [ $s1 = $s2 ]
     then
        echo "Le due stringhe sono uguali"
     else
        echo "Le due stringhe sono diverse"
     fi;;
```

soluz. 2/4

```
2)# Concatenazione
 echo "Prima stringa"; read s1
 echo "Seconda stringa"; read s2
 t=$1$2
 echo $t;;
3)# Lunghezza
 echo "Inserire una stringa"; read s1
 t=${#s1}
 echo "Lunghezza di "$s1" = "$t;;
4)# Scomposizione
 echo "Inserire una stringa"; read s1
 t=${#s1}
 i=0
 while [ $t != $i ]
 do
   echo ${s1:$i:1}
   let i++
 done;;
```

soluz. 3/4

```
5)# Invertire
  echo "Inserire una stringa"; read s1
  t=${#s1}
  i=""
  while [ $t -ge 0 ]
  do
    i=$i${s1:$t:1}
    let t--`
  done
  echo "originale: "$s1" invertita: "$i;;
6)exit;;
*)echo "<<< valore errato >>>";;
```

soluz. 4/4

Random

```
#!/bin/bash
# $$ restituisce il valore del PID dello script

dado=(“uno” “due” “tre” “quattro” “cinque” “sei”)
let “i=$$ % 6”

echo ${dado[$i]}
```

Lanciare N volte il dado, e calcolarne la distribuzione dei risultati dei lanci

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
```

altezza

larghezza

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
```

Cambiare!!

```
$ 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
```