Universidad Nacional
de la Matanza

Sistemas de Computación 2
Trabajo Práctico N°2
Scripts Básicos

Comisión: 3900.
Dia de Cursada: Miercoles.
Docentes del Curso: Favio Rivalta | Juan Manuel Fera | Soledad Álvarez.
Turno: Noche (de 19 a 23 Hs.).
Fecha Entrega: 19/05/2010.
Curso Lectivo: 2010.

Integrantes del Grupo

Ricardo Bevilacqua	34.304.983
D´Aranno Facundo Nahuel	34.842.320
Moure Pablo	32.031.459
Acha Erica	33.515.479

Índice

1. Ejercicio 1

1.1. ¿Cuál es el objetivo del script?

El objetivo es listar los usuarios del grupo "user" en consola.

1.2. ¿Cuál es el objetivo de la primera línea de cualquier script escrito para los sistemas operativos de la familia UNIX? Ejemplifique al menos 3 diferentes líneas y ¿qué indicarían en cada caso?

El objetivo de la primer línea de los script para los sistemas de la familia UNIX indica el interprete con el cual debe ejecutar el script

1.3. Comente el código del script. No es aceptable poner la descripción de cada uno de los comandos, se debe realizar comentarios que ayuden a leer el script en forma más rápida y facilitar la compresión de la finalidad del mismo.

```
1  #!/bin/bash
2
3  # Nombre: Punto1.sh
4  # Trabajo: Programacion basica de scripts
5  # Numero de ejercicio: 1
6  # Entrega: Primer Entrega
7  #
8  # Grupo NÂ° 63
9  # Acha Erica            33.515.479
10 # D'Aranno Facundo      34.842.320
11 # Moure Pablo           32.031.459
12 # Bevilacqua Ricardo    34.304.983
13
14 #Chequea que no se le pasen parametros al script,
15 #en caso de tenerlos terminamos el mismo.
16 if test $# -ge 1
17 then
18         echo "Ejemplo 1"
19         echo "Uso: este script no necesita parametros."
20         exit 1
21 fi
22
23 #Se toman los datos de los grupos del sistema y se lo guarda en
24 #el archivo /tmp/grupos
25 getent group > /tmp/grupos
26
27 #Se listan del archivo /tmp/grupos los usuarios
28 #que pertenecen al grupo 'users'.
29 IFS=","
30 for dato in `cat /tmp/grupos | grep ^users | cut -d":" -f4`
```

```
31 do
32         echo $dato
33 done

35 #Por ultimo eliminamos el archivo /tmp/grupos
36 rm -f /tmp/grupos

38 exit 0
```

Punto 1

1.4. ¿Es un shell script? ¿Por qué?

Si, es un shell script. Podemos distinguirlo por varias caracteristicas:

- La primer linea del archivo es "#!/bin/bash", lo que indica que el script debe ser leido con el interprete bash.
- Las lineas siguientes describen una serie de comandos que pueden ser ejecutados por un interprete.

Además, en caso de tener extención (lo cual no es obligatorio en UNIX) esta puede ser ".sh", lo que identificaría al archivo como un script.

1.5. Proponga un ejemplo de uso.

```
1 $./Practica1.sh
2 ric
```

Ejemplo de uso

Se puede tambien cambiar el grupo al cual hace refencia el script, pudiendo conocer los usuarios miembros de otros grupos.

1.6. Supongamos que tiene que controlar la ejecución paso a paso y el contenido de las variables del script, ¿qué comando utilizaría? (Tip: vea el comando set y sus parámetros -x, -v, -e, -u).

Utilizaría el comando sed que cambia los parámetros de ejecucion del bash. Sus opciones son las siguientes:

-x Imprime los comandos y sus argumentos antes de ser ejecutados.

-v Imprime las lineas del shell mientras son leidas.

-e Termina la ejecución del shell si alguno de los comandos da error a la salida.

-u Muestra un mensaje de error si el script intenta usar una variable que no esta seteada.

1.7. Explique y ejemplifique el uso de las estructuras if test y for, mostrando distintas formas de uso.

1.7.1. if test

Ejecuta comandos basado en una condición .
La estructura de la condicion if es:

```
if test ( expresion ) then
...
fi
```

Ejemplo:

```
#comprueba que se envien solo dos parametros y lo informa

  if test $# -eq 2
  then
          echo "se recibieron dos parametros "
  else
          echo "no se recibieron exactamente dos parametros "
  fi
```

1.7.2. for

Es una estructura de iteración.
Ejemplo:

```
#imprime todos los parametros recividos

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

1.8. Explique qué es un parámetro, cómo se pasan parámetros a los script, cómo se los maneja dentro del script. ¿Qué pasa si tengo más de 10 parámetros en un script, cómo puedo acceder ellos?

Un parametro en una cadena que se le envia como dato al script en su llamada, escribiendo estas a continuacion del nombre de este, separadas por un espacio (o la expresion indicada en la variable $IFS). Los scripts reciben los parÃ¡metros que le pasa la shell como variables nombradas de la forma $1, $2, ..., $n. Para saber cuantos parametros hemos recibido se puede utilizar el sÃmbolo $#.

Si se envian mas de diez parametros (contando a $0 , el nombre del script, como uno de estos) no podremos acceder de la forma ya mensionada para ello deberemos utilizar el comando shift, este desplazara los parametros a la

izquierda cuantas veces se le especificique, por lo que en realidad no recibiremos mas de los parametros convencionales, sino que perderemos los primeros para poder acceder a los antes no podiamos, de esta forma si realizamos un shift, el $1 lo perderiamos y en su lugar tendremos el valor que antes se ubicaba en $2, de esta forma el parametro $9 contendra un parametro al que antes no podiamos acceder.

Sn método muy utilizado es acceder siempre a la variable $1 y mediante una iteracion ir realizando un shift para recorrer todos los parametros.

2. Ejercicio 2

Luego de hacer los puntos a, b y c el script modificado queda así:

```
1  #!/bin/bash
2
3  # Nombre: Punto2.sh
4  # Trabajo: Programacion basica de scripts
5  # Numero de ejercicio: 2
6  # Entrega: Primer Entrega
7  #
8  # Grupo NÂ° 63
9  # Acha Erica            33.515.479
10 # D'Aranno Facundo      34.842.320
11 # Moure Pablo           32.031.459
12 # Bevilacqua Ricardo    34.304.983
13
14 #Chequeo que se reciba solo un parametro.
15 #Caso contrario muestra el uso y sale.
16 if test $# -ne 1
17 then
18         echo "Ejemplo 2"
19         echo "Uso: `basename $0` directorio"
20         exit 1
21 fi
22
23 #Comprueba que el parametro pasado sea
24 #un directorio.
25 if test ! -d $1
26 then
27         echo "Ejemplo 2"
28         echo "Uso: `basename $0` directorio"
29         echo
30         echo "ERROR: El parÃ¡metro no es un directorio."
31 fi
32
33 #Se comprueba si se tienen permisos de lectura sobre el
       directorio.
34 if test ! -r $1
35 then
36         echo "ERROR: Permiso denegado al directorio $1."
37         exit 0
```

```
38 fi


41 #Lista los nombres y fechas de creacion de los
42 #directorios dentro de la ruta recibida por parametro.
43 for linea in `find $1`
44 do
45         if test -d $linea # PUNTO B
46         then

48             echo $linea: `date -r $linea`

50         fi
51 done

53 exit 0
```

Punto 2

3. Ejercicio 3

```
1 #!/bin/bash

3 # Nombre: Punto3.sh
4 # Trabajo: Programacion basica de scripts
5 # Numero de ejercicio: 3
6 # Entrega: Primer Entrega
7 #
8 # Grupo NÂ° 63
9 # Acha Erica            33.515.479
10 # D'Aranno Facundo      34.842.320
11 # Moure Pablo           32.031.459
12 # Bevilacqua Ricardo    34.304.983

14 if [ $# -ge 5 ]
15 then


18 fi

20 directorio=`echo $* | cut -d" " -f$#`

22 if [ ! -d $directorio ]
23 then

25     permisos=$directorio
26     directorio="./"

28 else
29     permisos=`echo $* | cut -d" " -f$(($# - 1 ))`

31 fi
```

```
33 case $1 in
34     -M)
35         archivos=`find $directorio -mtime -$( date +%d )`
36         ;;

38     -X) tam=$2
39         if test `echo $tam | grep "[^0-9]" > /dev/null; echo $?`
40         then
41             archivos=`find $directorio -size +$( echo $tam )k`
42         fi
43         ;;

45     -C) cad=$2
46         archivos=`ls $directorio | grep $cad`
47         ;;

49     -G) grp=$2
50         archivos=`find $directorio -group $grp`
51         ;;

53       -?)     echo -e "ERROR: Parametros incorrectos."
54               echo  "`basename $0` Version 1.0"
55               echo  "Cambia permisos a subdirectorios y archivos
     dentro de directorio."
56               echo  "USO: `basename $0` [opciones...] permisos [
     directorio]"
57               echo  "OPCIONES"
58               echo  "-M           Cambia permisos si fue
     modificado en el mes actual."
59               echo  "-X <tamano>  Cambia permisos si el archivo
     supera el tamano."
60               echo  "-C <cadena>  Cambia permisos si el nombre
     contiene la cadena."
61               echo  "-G <grupo>   Cambia permisos si el archivo
     pertenece al grupo."
62               echo  "-?           Muestra la ayuda."

64           exit 0
65         ;;

67       *) archivos=`ls $directorio`
68         ;;
69 esac

71 for archivo in $archivos
72 do
73     chmod $permisos $archivo > /dev/null
74 done

76 exit 0

78 #
79 # FIN SCRIPT
```

```
80 #
```

Punto 3

4. Ejercicio 4

```
1  #! /bin/bash

3  # Nombre: Punto4.sh
4  # Trabajo: Programacion basica de scripts
5  # Numero de ejercicio: 4
6  # Entrega: Primer Entrega
7  #
8  # Grupo NÂ° 63
9  # Acha Erica            33.515.479
10 # D'Aranno Facundo      34.842.320
11 # Moure Pablo           32.031.459
12 # Bevilacqua Ricardo    34.304.983

14 # | ($# = 2 & $1 = "-R" & ! -d $2)'

16 # la siguiente funcion muestra el uso del script en pantalla y
       termina el proceso
17 function error(){

19           echo -e "ERROR: Parametros Incorrectos.
20                     \nUSO: 'basename $0 ' [-R] directorio
21                   \n     -R    Indica si el script debe
       ejecutarce en forma recursviva
22                   \n           cambiando los nombres del los
       subdirectorios del directorio indicado\n"

24           exit 1;
25 }

27 #corroboro que los parametros enviados sean correctos, en caso
       correcto llamo a la funcion error
28 case $# in

30       1)
31             if [ ! -d $1 ]
32             then
33                   error

35             else
36                   # establesco los valores de las variables
       desde los argumentos
37                   # (caso de un solo parametro)

39                   directorio=$1
40                   recursividad=0
```

```
42              fi
43          ;;

45          2)
46              if [  $1 != -R ]
47              then
48                  error

50              else

52                  if [ ! -d $2 ]
53                  then
54                      error

56                  else
57                      # establesco los valores de las
    variables desde los argumentos
58                      #  (caso de dos parametro)
59                      directorio=$2
60                      recursividad=1

62                  fi
63              fi
64          ;;

66          *)
67              #cualquier otro parametro indicara Error
68              error

70          ;;

72  esac


75  echo "\nEl directorio a renombrar es: " $directorio "y la
    recursividad esta en estado : " $recursividad "\n"

77  #recorro el directorio enviado como parametro renombrando sus
    archivos en forma normalizada

79    for archivo in `ls $directorio`
80    do

82              if test  -d $archivo -a $recursividad -eq 1
83              then
84                      # efectua recursividad sobre las
    subcarpetas
85                      # en caso de que el parametro -R
    halla sido enviado
86                      echo hola!!!!
87                      #  bash ./Ejercicio -R $archivo
88              fi
```

```
90          #obtengo el nuevo nombre del archivo y lo almaceno en
    $archivonuevo
91          #en caso de no ser el mismo nombre que antes ($archivo)
    efectuo el cambio

93          archivonuevo=`echo $archivo | sed -r 's/([A-Z])/\L&/g'|
    sed -r 's/(^.|[.;\)\(* _-,].)/\U&/g'`

95              if test $archivo !=  $archivonuevo
96              then
97                                      echo "se cambiara el
    nombre de " $archivo "a " $archivonuevo
98                                      mv $archivo $archivonuevo
99              fi

101   done

103 exit 1;
```

Punto 4

5. Ejercicio 5

```
1  #!/bin/bash

3  # Nombre: Punto5.sh
4  # Trabajo: Programacion basica de scripts
5  # Numero de ejercicio: 5
6  # Entrega: Primer Entrega
7  #
8  # Grupo NÂ° 63
9  # Acha Erica            33.515.479
10 # D'Aranno Facundo      34.842.320
11 # Moure Pablo           32.031.459
12 # Bevilacqua Ricardo    34.304.983

14 #Funcion ayuda imprime un texto con la ayuda.
15 function ayuda(){

17     echo -e " `basename $0`
18             \n Crea Back Ups del directorio pasado por parametro
    .
19             \n USO: `basename $0 ` directorio\n"

21 }

23 #Compruebo que haya solo un parametro.
24 if [ ! $# -eq 1 ]; then

26     echo "ERROR: Parametros incorrectos."
27     ayuda
28     exit 1
```

```
29 fi

31 #Compruebo si el parametro es la ayuda.
32 if [ $1 = "-?" ]; then
33     ayuda
34     exit 0
35 fi

37 #Compruebo que el parametro sea un directorio.
38 if [ ! -d $1 ]; then

40     echo "ERROR: No es un directorio."
41     ayuda
42     exit 1
43 fi

45 #Extraigo el directorio pasado por parametro y el nombre base
       sin
46 #la ruta absoluta o relativa del directorio.
47 directorio=$1
48 dirName="`basename $1`"

50 #Chequeo que exista el archivo bkup.conf en el directorio donde
       se ejecuta el script.
51 if [ -f ./bkup.conf ]; then
52     #Intento tomar el parametro backupdir del archivo de
       configuracion.
53     bkupDir="`grep "backupdir" bkup.conf | cut -d"=" -f2`"

55     #Chequeo que el parametro no sea vacio.
56     if [ $bkupDir = "" ]; then
57         echo "ERROR: bkup.conf: El parametro backupdir no existe
       o es incorrecto."
58         exit 1
59     fi

61     #Chequeo que el parametro tomado sea un directorio.
62     if [ ! -d $bkupDir ]; then
63         echo "ERROR: bkup.conf: El parametro backupdir no existe
       o es incorrecto."
64         exit 1
65     fi

67 else #Si no existe el archivo de configuracion, pregunto si
       desea crearlo.
68     echo "ERROR: bkup.conf: No existe archivo de configuracion."
69     echo "Desea crearlo? (y/n)"; read -n 1 answer

71     if [ $answer = "y" ]; then
72         echo "Creando archivo bkup.conf y directorio de Back Ups
       por defecto..."
73         echo "backupdir=./bkup" > bkup.conf
74         mkdir bkup
75         bkupDir="./bkup"
```

```
76          echo "Listo! Creando Back Up..."
77      else
78          echo "Saliendo."

80          exit 1
81      fi
82  fi

84  # Formato de archivo de Back Up: nombredirectorio%fecha%.tar.gz
85  #Obtengo el nombre del archivo del Back Up.
86  bkupName="$dirName%‘date +%d-%m-%y-%H:%M‘%.tar.gz"

88  #Compruebo si hay Back Ups anteriores.
89  if [ ‘ls $bkupDir | grep "$dirName" > /dev/null ; echo $?‘ = "0"
        ]; then
90      #Rescato el nombre del ultimo Back Up.
91      ultimoBkUp="$bkupDir/‘echo $(ls -t $bkupDir) | cut -d" " -f1
        ‘"
92      #Busco los archivos modificados desde el ultimo Back Up.
93      archMod="‘find $directorio -newer $ultimoBkUp -type f‘"

95      #Chequeo si hay algun archivo modificado.
96      if [ ! "$archMod" = "" ]; then
97          #Creo el nuevo Back Up.
98          tar -caf "$bkupDir/$bkupName" $archMod

100         #Informo al usuario.
101         echo "Creado Back Up $bkupName"
102         echo "Archivos:"
103         echo $archMod | sed 's/ /\n/g'
104     else
105         #Si no hay archivos modificados, no se hace Back Up.
106         echo "No hay archivos modificados. No se crea Back Up."
107     fi

109 else
110     #Creo el Back Up de todo el directorio.
111     tar -caf "$bkupDir/$bkupName" $directorio

113     #Informo al usuario los archivos agregados al Back Up.
114     echo "Creado Back Up $bkupName"
115     echo -e "Archivos: \n" ; find $directorio

117 fi


120 exit 0


123 #
124 # FIN
125 #
```

Punto 5