PRACTICAS DE SISTEMA OPERATIVO UNIX

PRACTICAS DE UNIX

PRACTICA 6. Órdenes de programación shell.

Objetivos
Tener un primer contacto con las herramientas de programación shell en cuanto al control de flujo de ejecución en los guiones shell.

Herramientas
Las herramientas a utilizar serán:
If test case while until for return true/false functionbreak/continue

DESARROLLO DE LA PRACTICA

1. Sentencia IF
   1.1. Codifique un shell script llamado "p6_1-1" que reciba un parámetro posicional, y realice la siguiente gestión:
      a) Verifique que se le ha pasado un argumento como parámetro posicional. Si no es correcto se mostrará el mensaje:
      "La llamada debe ser p6_1-1 argumento"
      b) Si el argumento no es un fichero ordinario debe mostrar el siguiente mensaje:
      "<parámetro> no es un fichero ordinario."
      Y si existe un fichero ordinario en el directorio actual que tenga ese nombre, se indicará si es o no ejecutable, legible, si se puede escribir en él o no y si su tamaño es cero o mayor de cero.
      c) Si existe en el directorio de trabajo un subdirectorio con el mismo nombre indicado en el parámetro, se visualizará un listado con las entradas que existan en el subdirectorio, y si no es un directorio mostrará el mensaje:
      "<parámetro> no es un directorio."

```
if [ $# -lt 1 ]
then
   echo "Introduce un parametro, por favor."
else
   var=`find . -type f -name $1 -print`
   if [ -z "$var" ]
   then
      echo "$1 no es un fichero ordinario"
   else
      var=`find . -type d -name $1 -print`
      if [ ! -z "$var" ]
      then
        echo "$1 es un directorio"
        ll $1
      else
        echo "$1 no es un directorio"
      fi
   fi
fi
```

   1.2. Codifique un shell script que visualice el texto:
   "Se encontró algún archivo llamado prueba."
   si existe al menos un fichero ordinario llamado "prueba" en el sistema de ficheros.

```
if find / -depth -name prueba -print
then
  echo “Se encontró algún archivo llamado prueba.”
fi
```

1.3. Genere un shell script llamado "y" que simule el comportamiento del metacaracter "&&", es decir:
El shell "y" debe recibir dos parámetros posicionales que serán dos instrucciones UNIX. (Si en las instrucciones indicadas como parámetros existen espacios en blanco no olvidar dar el parámetro entre apóstrofos o dobles comillas.)
Si la primera instrucción se ejecuta de forma satisfactoria (consultaremos la variable $?) se ejecutará a continuación la segunda instrucción. En caso contrario no se procesa el segundo parámetro.
El guión devolverá un valor cierto (0) si se ejecutan correctamente las dos instrucciones indicadas como parámetros. Cuando no se ejecute de forma correcta alguna de las dos instrucciones devolverá un valor falso (1).

```
if [ $# -lt 2 ]
then
   echo "y necesita dos parámetros para su ejecución."
else
   valor=1
   $1
   if [ $? -eq 0 ] then
       $2
       if [ $? -eq 0 ] then
          valor=0
       fi
   fi
fi
echo $valor
exit $valor
```

1.4. Ahora cree otro shell script llamado "o" que simule el comportamiento del metacaracter "||".
El guión devolverá un valor cierto (0) si se ejecuta correctamente al menos una de las dos instrucciones indicadas como parámetros. En otro caso devolverá falso (1).

```
if [ $# -lt 2 ]
then
   echo "o necesita dos parámetros para su ejecución."
else
   valor=1
   $1
   if [ $? -eq 0 ] then
       valor=0
    fi
    $2
    if [ $? -eq 0 ] then
          valor=0
       fi
   fi
echo $valor
exit $valor
```

1.5 Realizar un shell script que copie el fichero indicado como primer parámetro posicional de manera que la copia tenga el nombre indicado en el segundo parámetro posicional. Hay que controlar:
a) Que se indiquen dos parámetros.
b) Que exista y sea archivo ordinario el primer parámetro.
c) Que no exista un identificador (fichero ordinario, directorio, etc.) con el mismo nombre que el indicado en el segundo parámetro.

Si se produce alguna de las situaciones anteriores se visualizará un mensaje de error indicativo de tal error.

```
if [[ ! $# -eq 2 ]]
  then
     echo "Debes indicar dos parámetros."
  else
```

```
if [[ -f $1 ]]
   then
     if [[ ! -a $2 ]]
        then
           cp $1 $2
        else
           echo "Ya existe un id. llamado $2."
     fi
   else
      echo "$1 no es fichero ordinario."
fi
```

2. Sentencias CASE

2.1 Utilizando una sentencia "case", construya un shell script que, tomando como valores los ficheros de su directorio de conexión, organice una estructura de subdirectorios, de la forma:

Ficheros con extensión ".c" se mueven al directorio "prog_c".
Ficheros con extensión ".f" se mueven al directorio "prog_for".
Ficheros con extensión ".p" se mueven al directorio "prog_pas".
El resto de ficheros no se mueven.

Los directorios indicados únicamente se crearán si existen archivos para ser movidos a tales directorios.

```
for archivo in $HOME/*
  do
    case of $archivo
       *.c) if  [ ! test -d $HOME/prog_c ]
                  then
                    mkdir $HOME prog_c
              fi
              mv $HOME/$archivo $HOME/prog_pas/$archivo
       *.f) if  [ ! test -d $HOME/prog_for ]
                  then
                    mkdir $HOME prog_for
              fi
              mv $HOME/$archivo $HOME/prog_c/$archivo
       *.p) if  [ ! test -d $HOME/prog_pas]
                  then
                    mkdir $HOME prog_pas
              fi
              mv $HOME/$archivo $HOME/prog_pas/$archivo
    esac
  done
```

2.2 Generar un shell script que muestre el siguiente menú de opciones y ejecute los tratamientos correspondientes:

1. Mostrar un listado con los usuarios de su grupo.
2. Mostrar el nombre de los usuarios de su mismo grupo junto al directorio de conexión que tienen asociado.
3. Hacer un listado con el nombre de los grupos dados de alta y el número de usuarios que tiene cada grupo.
4. Salir.

Se abandonará el shell script cuando se seleccione la opción 4.

```
function menu
{
echo Usuarios de FT-22
echo =============
echo   1.- Lista users
echo   2.- Dirs. $HOME
echo   3.- Lista Grupos
echo   4.- Salir al shell\n\n
echo        Opción: \c
```

```
read opcion
case opcion in
  1) listado;;
  2) directorios;;
  3) grupos;;
  }
function listado
{
echo " Listado usuarios del FT22"
echo "====================="
for i in ´grep /etc/group "ft22$" | cut -f4 -d":" | cut -d","´
  do
    echo "$i\n"
  done
}
function directorios
{
echo " Listado usuarios del FT22 con directorios de conexión"
echo "======================================="
for i in ´grep /etc/group "ft22$" | cut -f4 -d":" | cut -d","´
  do
    dir=echo `grep /etc/passwd "ft22$" | cut -f6 -d":"
    echo "$i\t\t$dir\n"
  done
}
function grupos
{
echo " Listado grupos con # de usuarios"
echo "========================"
for i in ´cat /etc/group´
    do
       echo $i | cut -f1 -d":"\c
       echo $i | wc | -f4 -d":"\n
    done
}
opcion=
until [$opcion=4]
  do
    menu
  done
```

3. Sentencias WHILE y UNTIL

3.1 Construya un shell script que lea un valor desde teclado, hasta que se obtenga la cadena "fin".

```
until [$cadena=fin]
do
  read cadena
done
```

3.2 Genere un shell script que admita varios, uno o ningún argumento.
Si se indica algún argumento, este será el nombre de algún fichero. Mientras exista un argumento se mostrará el siguiente menú de opciones para realizar el tratamiento correspondiente con el argumento:

1. Comprobar si el fichero existe en su directorio actual.
2. Mostrar su nombre y tamaño.

En caso de no existir ningún argumento, visualizará el mensaje:
"no existen más argumentos".

```
function menu
{
 clear
echo "Fichero $1"
```

```
echo "1. Comprobar existencia del fichero."
echo "2. Mostrar nombre y tamaño.\n\n"
read opcion
case opcion in
  1) buscar
  2) mostrar
}
function buscar
{
if [ -f $1 ]
then
   echo "Existe $1."
else
   echo "No existe $1."
fi
}
function mostrar
{
ls -l $1 | tr -s" " | cut -f2,5 -d" "
}
while [ ! -z $# ]
do
  menu
  shift
done
echo "No existen más argumentos."
```

3.3 Escriba un shell script que procese los números enteros del 1 al 20 y liste en tres columnas: el número, su cuadrado y su cubo.

```
clear
num=1
while [$num -le 20]
do
   ((cua=$num*$num))
   ((cub=$cua*$num))
   echo "$num\t$cua\t$cub"
done
```

3.4 Cree un shell script que borre todos aquellos ficheros que se hayan introducido como argumentos. Se borrarán todos los ficheros de su propiedad que tengan el nombre especificado. Si los argumentos son directorios, se visualizará su nombre junto con el mensaje "es un directorio".

```
for i in $*
  do
    if [ -d $i ]
      then
         echo "$i es un directorio."
      else
         rm $i
    fi
  done
```

3.5 Escriba un shell script que lea números hasta encontrar un cero y escriba si el número leido es primo o no.

4. Sentencia FOR

4.1 Utilice un bucle "for" para eliminar de su directorio actual todos los ficheros que comiencen por "a". Adicionalmente, lleve el nombre de cada fichero borrado a un fichero llamado "borrados".

```
for i in ´ls a*´
```

```
  do
    rm $i
    echo $1 >> borrados
  done
```

4.2 Busque en el file system completo todos los ficheros que usted especifique como argumentos de su actual shell script y visualice por cada argumento un listado de la siguiente manera:
El archivo ordinario <argumento> se encuentra en los directorios:
<directorio1>
<directorio2>
.....
donde <directorio?> será la ruta de acceso absoluta a dicho archivo (sin el nombre del archivo).

```
for nombre in $*
  do
    echo "El archivo ordinario $nombre se encuentra en los
                                                    directorios:"
    for dir in `find / -depth -type f $nombre`
      do
        echo "\t`dirname $dir`"
      done
  done
```

4.3 Utilizando un bucle "for", obtenga los cuadrados de todos los números que especifique como parámetros del guión.

```
for i in $*
  do
    let a= $i * $i
    echo $a
  done
```

4.4 Utilizando un bucle "for", distinga cuales de las anotaciones de su directorio actual, corresponden con un directorio y visualice su nombre.

```
for i in ls -1 *
  do
    if [ -d $i ]
      then
        echo "$i es un dir"
    fi
  done
```

4.5 Genere un shell script que muestre por pantalla los nombres y números de líneas que contiene cada fichero ordinario de su directorio de trabajo de la forma:
nombre: <NOMBRE>
contiene: <NUMERO> líneas

```
for i in ls -1 *
          do
            if [ -f $i ]
              then
                echo "nombre: $i"
                echo "contiene: `cat $i | wc -l` lineas"
              fi
          done
```

4.6 Genere un shell script que permita borrar aquellos ficheros cuyos nombres especifique como argumentos. Verifique anteriormente que existen y que son ordinarios. (Como argumento se especificará el directorio junto con el nombre del fichero)

```
if [ $# -lt 1 ]
then
   echo "Introduce un parametro, por favor."
else
   if [ !-f $1 ]
   then
      echo "$1 no es un fichero ordinario"
   else
      rm $1
   fi
fi
```

4.7 Genere un shell script que permita enviar un mensaje con "mail" a los usuarios que estén conectados al sistema en este momento, saludándolos.

```
for  cuenta in `who | tr -s " " | cut -f1 -d" "`
   do
       echo ¡Hola! | mail $cuenta
   done
```