SHELL SCRIPT

Por Cayetano Borja Carrillo

Índice:

Ejercicio 1 – Par o impar

Crea un script que pida un número por pantalla y que diga si el número es par o impar.

Solución

```
#!/bin/bash
clear
echo -n "Introduzca un numero: " #-n es para que no haga un salto de linea
read numero
echo "" #No muestra nada, se usa para hacer un salto de linea
res=$(( $numero % 2)) #La variable "res" será igual a "numero" modulo 2
if [ $res -eq 0 ]; then #-eq significa igual y solo se usa para comparar
numeros. Para cadenas se utiliza ==
    echo "El numero es par"
else
    echo "El numero es impar"
fi
echo ""
read -p "Pulsa cualquier tecla para finalizar"
```

Ejercicio 2 – Mostrar los permisos de un archivo

Crea un script que reciba un nombre de archivo como parámetro e indicar si el archivo es legible, modificable y ejecutable por el usuario

Solución

```
#!/bin/bash
clear
#Comprobamos si se ha introducido algun parametro
if [ $# -eq 0 ]; then #$# es una variable cuyo valor sera el numero de
parametros introducidos
    read -p "Error: No has introducido ningun parametro"
    exit #Sale directamente del programa.
fi

#Comprobamos que solo se haya insertado 1 parametro
if [ $# -ne 1 ]; then #-ne significa distinto y solo se usa para comparar
numeros, para cadenas se utiliza !=
    read -p "Error: Solo se puede introducir un parametro"
    exit
fi

#Ahora comprobamos si el parametro insertado es un fichero o archivo
if [ -f $1 ]; then #$1 contiene el primer parametro. -f devuelve verdadero
si la variable $1 es un archivo existente y falso si no lo es.
    if [ -r $1 ]; then #-r devuelve verdadero si tiene permiso de lectura
        echo "El fichero es legible"
    fi
    if [ -w $1 ]; then #-w devuelve verdadero si tiene permiso de escritura
        echo "El fichero es modificable"
    fi
    if [ -x $1 ]; then #-x devuelve verdadero si tiene permiso de ejecucion
        echo "El fichero es ejecutable"
    fi
    echo ""
    read -p "Pulsa cualquier tecla para continuar"
else
    read -p "Error: El parametro introducido no es un fichero o no existe"
fi
```

Ejercicio 3 – Mostrar el contenido de los archivos

Crea un script que reciba nombres de archivos como parámetros. Después que haga un recorrido de cada parámetro y, si el parámetro es un archivo, que muestre su contenido.

Solución

```
#!/bin/bash
clear
if [ $# -eq 0 ]; then
    read -p "Error: No has introducido ningun parametro"
    exit
fi

#Hacer desde el primer parametro hasta el ultimo
for i in $*; do #$* es una variable que contiene una cadena de texto con
todos los parametros pasados
    clear
    if [ -f $i ]; then
        more $i #"more $i" muestra el contenido de ese archivo
    else
        echo "$i no es un archivo o no existe"
    fi
    echo ""
    read -p "Pulsa cualquier tecla para continuar"
done
```

Ejercicio 4 – Cambiar los permisos de un archivo

Crea un script que reciba un nombre de archivo, verifique que existe, lo convierta en ejecutable para el propietario y el grupo y muestre el modo final.

Solución

```
#!/bin/bash
clear
if [ $# -eq 0 ]; then
    read -p "Error: No has introducido ningun parametro"
    exit
fi

if [ $# -ne 1 ]; then
    read -p "Error: Solo se puede introducir parametro"
    exit
fi

if [ -f $1 ]; then
    chmod ug+x $1 #Comando "chmod" cambia los permisos. "u" significa
usuario y "g" grupo, "+x" le da permiso de ejecucion
    ls -l $1 #Comando "ls" muestra los archivos y carpetas de ese directorio
else
    echo "Error: El parametro introducido no es un archivo o no existe."
fi
echo ""
read -p "Pulsa cualquier tecla para continuar"
```

Ejercicio 5 – Menú básico

Escribe un script que muestre un menú como el siguiente:

a) Visualizar el contenido del directorio actual en formato largo
b) Visualizar el contenido del directorio actual
c) Mostrar el contenido de un fichero
d) Salir

Solución

```
#!/bin/bash
while true ; do #Mientras sea verdadero haz
    clear
    echo "a) Visualizar el contenido del directorio actual en formato largo"
    echo "b) Visualizar el contenido del directorio actual"
    echo "c) Mostrar el contenido de un fichero"
    echo "d) Salir"
    echo -n "Escoge una opcion: "
    read opcion
    echo ""
    case $opcion in #Segun $opcion haz
        a) # Se ha escrito a
            ls -l
        ;; # Como el break de C
        b)
            ls
        ;;
        c)
            echo -n "Introduce el nombre del archivo: "
            read fichero
            echo ""
            if [ -f $fichero ]; then #Si es un fichero, entonces
                cat $fichero #"cat" muestra contenido de carpetas o archivos
            else
                echo "El texto introducido no es un fichero"
            fi
        ;;
        d)
            exit
        ;;
        *) # Se escribe cualquier otra opcion
            echo "Opcion incorrecta. Vuelve a intentarlo"
        ;;
    esac # Fin del case
    read enterkey #Espera a que se pulse una tecla
done # Fin del while
```

Ejercicio 6 – Copia de archivos

Crea un script que reciba varios parámetros. El script debe validar que el primer parámetro sea un directorio y que el resto sean archivos, posteriormente, copia esos archivos en ese directorio.

Solución

```
#!/bin/bash
clear
#Primero comprobamos que haya recibido al menos 2 argumentos
if [ $# -le 1 ]; then #-le significa menor o igual
    read -p "Error: Se necesitan al menos 2 argumentos. Ej: Carpeta Archivo1
Archivo2"
    exit
fi

#Segundo copiamos los archivos en el directorio. Antes comprobamos que el
primer argumento es una carpeta
if [ -d $1 ]; then #-d devuelve verdadero si la variable es una carpeta
    carpeta=$1
    numarchivos=0
    for i in $*; do #Desde i=0 hasta fin de los argumentos
    #Comprobamos que el resto de argumentos sean realmente archivos
        if [ -f $i ]; then #-f devuelve verdadero si la variable es un
archivo
            cp $i $carpeta #El comando "cp" sirve para copiar. Copia el
fichero "$i" dentro de carpeta "$carpeta"
            echo "Archivo $i copiado en $carpeta"
            numarchivos=$(( $numarchivos + 1 ))
        else
            if [ $i != $carpeta ]; then #Cuando hay que comparar cadenas no
se utiliza -eq, -ne, etc. Se utiliza ==, !=, etc.
                echo "El argumento $i no es un archivo o no existe"
            fi
        fi
        echo ""
    done
    read -p "Se han copiado $numarchivos archivos"
else
    read -p "Error: El primer argumento introducido no es un directorio"
fi
```

Ejercicio 7 – Mostrar directorios y subdirectorios

Crea un script que reciba un nombre de directorio como parámetro, validar su existencia y condición de directorio y mostrar los nombres de todos los directorios y subdirectorios bajo él en formato de página largo 23.

Solución

```
#!/bin/bash
clear
if [ $# -eq 0 ]; then
    read -p "Error: No has introducido ningun parametro"
    exit
fi

if [ $# -ne 1 ]; then
    read -p "Error: Solo se puede introducir un parametro"
    exit
fi

if [ -d $1 ]; then
    ls -lR $1 | grep "^d" | pr -l 24 | more -24 #Aqui se ejecutan 4
comandos, el simbolo | se llama tuberia y se utiliza para encadenar comandos
(la salida de un comando sera la entrada del otro).

    # Comando ls: Muestra todos los archivos y carpetas de un directorio. -l
lo muestra en formato largo. R muestra todos sus subdirectorios y los
subdirectorios de sus subdirectorios

    # Comando grep: Muestra lineas donde aparece una palabra. "^d" muestra
solo las lineas que empiecen por "d", es decir, los directorios. Si se
escribiera "d" muestra las lineas que contengan una "d" y si se escribe -v
"d" muestra los que no contengan "d". Considerando esto, tambien se podia
haber escrito "grep -v "^-" ya que ocultamos todo lo que no son directorios.

    # Comando pr -l 24: Muestra en cada pantalla 24 lineas.

    # Comando more -24: El comando "pr" muestra todas las lineas separadas
en pantallas de golpe, el comando more -24 muestra las primeras 24 lineas y
despues pide que le pulses un boton para mostrar mas.

else
    read -p "Error: El parametro introducido no es un directorio"
fi
```

Ejercicio 8 – Información sobre los usuarios

Crea un script que muestre los nombres de los usuarios conectados. A continuación, el script deberá pedirnos que ingresemos uno de esos usuarios y, si existe, mostrará su nombre completo y la ruta de su directorio personal.

Solución

```
#!/bin/bash
clear
#Primero mostramos los nombres de los usuarios conectados.
echo "Los usuarios conectados son los siguientes:"
echo ""
who | cut -d " " -f1 | nl #Aqui se ejecutan 3 comandos.
#Comando who: Muestra los usuarios conectados en el sistema.
#Comando cut: Divide cada fila en bloques. -d " " quiere decir que dividira
usando un espacio como delimitador. -f1 quiere decir que tomaremos solamente
el primer bloque de todos los separados. Otro uso del cut es por ejemplo
"cut -c1,5", con esto solo se mostrara el caracter 1 y 5 de cada fila.
#Comando nl: Numera cada fila.

#Ejemplo, si escribimos "who" se mostrara:
    # usuario1 pts/0 2010-05-01 18:00 (:0)"
    # usuario2 pts/1 2010-05-01 18:00 (:0)"
# Escribiendo "cut" y utilizando un espacio como delimitador, nos dividira
cada fila en 5 bloques ([usuario1],[pts/0],etc). Como lo unico que nos
interesa el el primer bloque escribimos -f1. Despues de eso se mostrara:
    # usuario1
    # usuario2
# Si escribimos nl nos enumerara cada fila quedando finalmente:
    # 1 usuario1
    # 2 usuario2

#Segundo nos pide que ingresemos uno de esos usuarios.
echo ""
echo -n "Escribe un usuario: "
read Usuario

echo ""
if [ -z $Usuario ]; then #-z devuelve true si la longitud de la cadena es 0
    read -p "Error: No se ha introducido ningun valor"
else
    NombreUsuario=$(grep ^$Usuario /etc/passwd | cut -d : -f5 | cut -d , -
f1) #NombreUsuario sera igual a la cadena de texto resultante despues de
esos comandos.
    #Comando grep: Muestra lineas donde aparece una palabra. Con "^Usuario
/etc/passwd" mostrara la informacion de ese usuario en esa carpeta. El
simbolo ^ quiere decir que solo muestre las filas que empiecen por ahi.
```

```
    #Comando cut: Nos quedaremos con el quinto bloque si separamos la fila
por el delimitador ":"
    #Comando cut: Despues del cut anterior nos quedara "Usuario1,,,", si nos
quedamos con el primer bloque utilizando coma como delimitador nos
quedaremos finalmente con "Usuario" que es lo que queremos.

    HomeUsuario=$(grep ^$Usuario /etc/passwd | cut -d : -f6)
    if [ -z $NombreUsuario ]; then
        read -p "El nombre de usuario escrito no existe"
    else
        echo "El nombre del usuario es: $NombreUsuario"
        echo "Su carpeta personal es: $HomeUsuario"
        echo ""
        read -p "Pulsa cualquier tecla para continuar ..."
    fi
fi
```

Ejercicio 9 – Listado de usuarios

Crea un script que liste los nombres de login, el directorio propio del usuario y el intérprete invocado por defecto de todos los usuarios, ordenados alfabéticamente por nombre de login.

Solución

```
#!/bin/bash
clear
cat /etc/passwd | cut -d : -f1,6,7 | sort | more
#Comando cat: Muestra contenido de carpetas o archivo. Escribiendo "cat
/etc/passwd" nos muestra muchas lineas como las siguientes:
    #colord:x:102:105:colord management daemon,,,:/var/lib/colord:/bin/false
    #messagebus:x:103:107::/var/run/dbus:/bin/false
    #lightdm:x:104:108:Light Display Manager:/var/lib/lightdm:/bin/false
#Con "cut" tomaremos el bloque 1, el 6 y el 7.
#Con "sort" Ordenamos las filas.
#Con "more" nos muestra una pantalla y luego nos pide que pulsemos un boton
para mostrar mas.
echo ""
```

Ejercicio 10 – Menú para solicitar información básica

Crea un script que muestre por pantalla 3 opciones. La opción 1 muestra los nombres, el número de grupo y la lista de usuarios ordenados por nombre. La opción 2 hace lo mismo pero ordenado por número de grupo. La opción 3 muestra el nombre de la máquina, la hora y fecha.

Solución

```
#!/bin/bash
while true; do
    clear
    echo "Opcion 1: Listar nombres, num de grupo y lista de usuarios
ordenados por nombre"
    echo "Opcion 2: Listar nombres, num de grupo y lista de usuarios
ordenados por grupo"
    echo "Opcion 3: Mostrar el nombre de la maquina, la hora y la fecha"
    echo "Opcion 4: Salir"
    echo ""
    echo -n "Elige una opcion: "
    read opcion
    echo ""
    case $opcion in
        1)
            echo "Nombre : Numero de grupo : Lista de usuarios"
            echo ""
            cat /etc/group | cut -d : -f1,3,4 | sort | more -21
        ;;
        2)
            echo "Nombre : Numero de grupo : Lista de usuarios"
            echo ""
            cat /etc/group | cut -d : -f1,3,4 | sort -t: -n -k2
            #Comando "sort -t: -n -k2". Con -t decimos que usaremos ":" de
delimitador. Con -n que lo que se van a ordenar son numeros. Con -k2 que
ordene el bloque 2 que es el grupo. #NOTA: Si no se hubiera puesto -t y hay
los siguientes valores: 1, 2, 11, 16, 22 hubiera ordenado de la siguiente
manera: 1,11,16,2,22, por eso hay que decir que son numeros.
        ;;
        3)
            echo "Nombre de la maquina:" `hostname`
            #El comando "hostname" muestra el nombre de la maquina, se tiene
que poner entre "`" porque sino, en vez de mostrar el nombre, muestra la
palabra hostname literalmente.
            echo "Fecha:" `date`
        ;;
        4)
            exit
        ;;
```

```
        *)
            echo "Opcion incorrecta. Vuelve a intentarlo"
        ;;
    esac
    read enterkey
done
```

Ejercicio 11 – Saludar según la hora del sistema

Crea un script que salude según la hora del sistema. Para ello, se ha de tomar el nombre del usuario que lo ejecute y la fecha. Los saludos corresponden a las siguientes horas: de 05:00 hasta 12:59 decir buenos días, de 13:00 hasta 19;59 decir buenas tardes y de 20:00 hasta 04:59 decir buenas noches. Ejemplo: Buenos días, Juan.

Solución

```
#!/bin/bash
clear
#Primero tomamos el nombre del usuario.
Usuario=$(whoami)
#Otra forma: Usuario=$(grep "^$LOGNAME" /etc/passwd | cut -d : -f5 | cut -d
, -f1)

#Segundo tomamos la hora.
Hora=$((10#$(date +"%H"))) #El 10 se pone para que lo pase a decimal, ya que
si sino lo interpreta como octal
#Otra forma: Hora=$(date | cut -d " " -f5 | cut -d : -f1)

#Tercero, mostrar el mensaje dependiendo de la hora.
#Si la hora es >= que 5 y < que 13.
if [[ ( $Hora -ge 5 ) && ( $Hora -lt 13 ) ]]; then #ge y lt es para numeros
    echo "Buenos dias, $Usuario."
else
    #Si la hora es >= que 13 y < que 20.
    if [[ ( $Hora -ge 13 ) && ( $Hora -lt 20 ) ]]; then
        echo "Buenas tardes, $Usuario."
    else
        #Si la hora es >= que 20 o >= que 0 y < que 5.
        if [[ ( $Hora -ge 20 ) ]] || [[ ( $Hora -ge 0 ) && ( $Hora -lt 5 )
]]; then
            echo "Buenas noches, $Usuario."
        fi
    fi
fi

read enterkey
```

Ejercicio 12 – Información sobre el sistema con funciones

Crea un script que muestre la siguiente información de configuración del sistema:

1. Usuario que ha iniciado la sesión actualmente y su nombre.
2. Tu shell actual.
3. Tu directorio home.
4. El tipo de tu sistema operativo.
5. La configuración actual del path.
6. Tu directorio actual de trabajo.
7. El número total de usuarios que han iniciado sesión.
8. La información sobre el SO: versión del núcleo y versión SO.
9. Todos los shells disponibles.
10. La configuración del ratón.
11. Información sobre la CPU: Tipo, velocidad, etc.
12. Información sobre la memoria del sistema.
13. Información sobre el sistema de archivos (si está montado).

Escribir un menú que nos permita la consulta e introducir cada punto en una función.

Solución

```
#!/bin/bash

################ IMPLEMENTAMOS LAS FUNCIONES ###############
function opcion1 {
    echo "Nombre del usuario: $USER"
    echo "Nombre de login: $LOGNAME"
    #Otra forma:
    #echo "Nombre del usuario: `grep "$USER" /etc/passwd | cut -d : -f1`"
    #echo "Nombre de login: `grep "$LOGNAME" /etc/passwd | cut -d : -f1`"
}

function opcion2 {
    echo "Shell actual: $SHELL"
    #Otra forma:
    #echo "Shell actual: `grep "$LOGNAME" /etc/passwd | cut -d : -f7`"
}

function opcion3 {
    echo "Directorio home: $HOME"
    #Otra forma:
    #echo "Directorio home: `grep "$LOGNAME" /etc/passwd | cut -d : -f6`"
}

function opcion4 {
    echo "El tipo del sistema operativo es: $OSTYPE"
    #Otra forma:
    #echo "El tipo del sistema operativo es: `uname -o`"
```

```
    #Comando uname muestra informacion del sistema. -o muestra informacion
del sistema operativo.
}

function opcion5 {
    echo "La configuracion actual del path es: $PATH"
}

function opcion6 {
    echo "El directorio actual es: `pwd`"
}

function opcion7 {
    usuarios=`who | wc -l`
    #Comando  who:
    #Comando wc: Cuenta dependiendo de un patron. -l cuenta las lineas.
    echo "Numero de usuarios conectados: $usuarios"
    #Otra forma:
    #echo "Numero de usuarios conectados: `who | wc -l`"
}

function opcion8 {
    echo "Version del nucleo: `uname -r`"
    echo "Version del sistema operativo `uname -v`"
    #Posiblemente la version del sistema operativo tambien funcione con: cat
/etc/issue
    #Escribiendo "cat /proc/version/" tambien aparece la version.
}

function opcion9 {
    echo "Los shell disponibles son los siguientes: "
    cat /etc/shells
}

function opcion10 {
    echo "Opcion no terminada"
}

function opcion11 {
    echo "La informacion sobre la CPU es la siguiente: "
    cat /proc/cpuinfo
}

function opcion12 {
    echo "La informacion sobre la memoria del sistema es la siguiente: "
    #cat /proc/meminfo   ##Supongo que este valdra pero no me gusta mucho.
    free #Este me gusta mas, muestra estadisticas del uso de la memoria
}
```

```
function opcion13 {
    echo "Opcion no terminada"
}

function opcion14 {
    echo "Opcion no terminada"
}

################## EMPEZAMOS EL PROGRAMA #################
opcion=0
while [ $opcion -ne 15 ]; do
    clear
    echo "1) Usuario que ha iniciado la sesion actualmente y su nombre."
    echo "2) Tu shell actual."
    echo "3) Tu directorio home."
    echo "4) El tipo de tu sistema operativo."
    echo "5) La configuracion actual del path."
    echo "6) Tu directorio actual de trabajo."
    echo "7) El numero total de usuarios que han iniciado sesion."
    echo "8) La informacion sobre el SO: version del nucleo y version del
SO."
    echo "9) Todos los shells disponibles."
    echo "10) La configuracion del raton."
    echo "11) Informacion sobre la CPU: Tipo, velocidad, etc."
    echo "12) Informacion sobre la memoria del sistema."
    echo "13) Informacion sobre el disco duro: Tamano, cache, modelo, etc."
    echo "14) Informacion sobre el sistema de archivos (si esta montado)."
    echo "15) Salir."
    echo ""
    echo -ne "Escoge una opcion: "
    read opcion
    echo ""
    case $opcion in
        1)
            opcion1
        ;;
        2)
            opcion2
        ;;
        3)
            opcion3
        ;;
        4)
            opcion4
        ;;
        5)
            opcion5
```

```
        ;;
        6)
            opcion6
        ;;
        7)
            opcion7
        ;;
        8)
            opcion8
        ;;
        9)
            opcion9
        ;;
        10)
            opcion10
        ;;
        11)
            opcion11
        ;;
        12)
            opcion12
        ;;
        13)
            opcion13
        ;;
        14)
            opcion14
        ;;
        15)
            exit
        ;;
        *)
            echo "Opcion invalida, vuelve a intentarlo"
        ;;
    esac
    read enterkey
done
```

Ejercicio 13 – Examen 2008

Crea un script llamado informe.sh que muestre la información sobre el uso del sistema. El script tendrá la siguiente sintaxis: *informe.sh opción [usuario]*

Las opciones pueden ser -u y -a. Si la opción es -u, deberá recibir como segundo argumento un nombre de usuario y mostrará la información sobre ese usuario. Si la opción es -a mostrará la información total del sistema.

La información que mostrar es la siguiente:

- Nombre del usuario (solo si la opción es -u).
- Numero de procesos que ha lanzado.
- Nombre del primer proceso que lanzo.
- Listado de todos los procesos que ha lanzado.
- Numero de directorios que tiene.
- Numero de ficheros regulares que tiene.
- Uso del espacio en disco que ocupa.
- Tanto por ciento de espacio en disco usado.

Solución

```
#!/bin/bash

################ IMPLEMENTAMOS LAS FUNCIONES ###############
#Funcion Error: Muestra un error.
function Error
{
    echo ""
    echo "Utiliza '$0 -u usuario' para presentar un informe de ese usuario."
    echo "Utiliza '$0 -a' para presentar un informe total del sistema."
    echo "Si se meten mas parametros de los necesarios, seran omitidos."
    echo ""
    exit
}

#Funcion NombreUsuario: Muestra el nombre de usuario.
function NombreUsuario
{
    clear
    echo "El nombre del usuario es: $NomUsuario"
    read enterkey
}

#Funcion NumProcesos: Muestra el numero de procesos en ejecucion.
function NumProcesos
{
    clear
```

```
    #Si se introdujo "-u" muestra el num de procesos del usuario, sino los
totales.
    if [[ $Eleccion == "-u" ]]; then
        NumProcesos=$( ps -U $NomUsuario | wc -l )
        #Comando "ps": Muestra los procesos en ejecucion. -U muestra solo
los de ese usuario.
        #Comando "wc": Cuenta dependiendo de un patron. -l cuenta las lineas
        NumProcesos=$(( $NumProcesos - 1 )) #Hay que quitar 1 ya que la
linea 1 es la leyenda no un proceso.
        echo "Numero de procesos que ha lanzado $NomUsuario: $NumProcesos"
    else
        NumProcesos=$( ps -e | wc -l )
        NumProcesos=$(( $NumProcesos - 1))
        #Comando "ps": -e muestra todos los procesos del sistema.
        echo "Numero de procesos totales cargados en el sistema: `ps -e | wc
-l`"
    fi
    read enterkey
}

#Funcion NomPrimerProceso: Muestra el nombre del primer proceso que se ha
ejecutado.
function NomPrimerProceso
{
    clear
    if [[ $Eleccion == "-u" ]]; then
        #Hay que saber antes cuantos procesos ha ejecutado ese usuario por
si son 0 omitir esta operacion.
        NumProcesos=$( ps -U $NomUsuario | wc -l )
        NumProcesos=$(( $NumProcesos - 1 ))
        if [[ NumProcesos -ne 0 ]]; then
            echo "Primer proceso que ha lanzado $NomUsuario: `ps -F -U
$NomUsuario | head -2 | tail -1 | awk '{print $11}'`"
            #Comando "ps": -F Muestra informacion completa.
            #Comando "head": Muestra las 10 primeras lineas. -2 solo muestra
las 2 primeras lineas.
            #Comando "tail": Muestra las 10 ultimas lineas. -1 solo muestra
la ultima linea.
            #Comando "awk":  El comando "cut" es bueno cuando los bloques de
todas las lineas son del mismo tamano, pero cuando son diferentes es mejor
usar el comando "awk". "awk '{print $x}'" muestra el bloque x.
            read enterkey
        fi
    else
        echo "Primer proceso lanzado por el sistema: `ps -e -F | head -2 |
tail -1 | awk '{print $11}'`"
        read enterkey
    fi
```

```
}

#Funcion ListadoTodosProcesos: Muestra un listado de todos los procesos en
ejecucion.
function ListadoTodosProcesos
{
    clear
    if [[ $Eleccion == "-u" ]]; then
        NumProcesos=$( ps -U $NomUsuario | wc -l )
        if [[ NumProcesos -ne 1 ]]; then
            echo "El listado de todos los procesos que ha lanzado
$NomUsuario es:"
            echo ""
            ps -F -U $NomUsuario | tail -$NumProcesos | awk '{print $11}' |
more -d -21
            #Si no escribimos "tail" se muestra la leyenda como primer
proceso y no lo es, de esta forma se oculta.
            read enterkey
        fi
    else
        NumProcesos=$( ps -e | wc -l)
        echo "El listado de todos los procesos que hay cargados en el
sistema es:"
        echo ""
        ps -e -F | tail -$NumProcesos | awk '{print $11}' | more -d -21
        read enterkey
    fi
}

#Funcion NumDirectorios: Muestra el numero de directorios y subdirectorios.
function NumDirectorios
{
    clear
    if [[ $Eleccion == "-u" ]]; then
        carpetausuario=`cat /etc/passwd | grep ^$NomUsuario | cut -d: -f6`
            echo "El numero de directorios y subdirectorios de $NomUsuario
es: `find $carpetausuario -type d | wc -l`"
            #Comando  "find": Busca archivos y carpetas. -type d busca
directorios
    else
        find /. -type d | wc -l >> temporal.$$$ #Almaceno el resultado en un
fichero. Lo hago asi para que no se muestre si existe algun permiso
denegado.
        clear
        echo "El numero de directorios y subdirectorios del sistema es:
`more temporal.$$$`" #Muestro el valor del fichero
        rm temporal.$$$ #Borro el fichero.
    fi
```

```
    read enterkey
}

#Funcion NumFicheros: Muestra el numero de ficheros.
function NumFicheros
{
    clear
    if [[ $Eleccion == "-u" ]]; then
        carpetausuario=`cat /etc/passwd | grep ^$NomUsuario | cut -d: -f6`
            echo "El numero de ficheros de $NomUsuario es: `find
$carpetausuario -type f | wc -l`"
    else
        find /. -type f | wc -l >> temporal.$$$ #Almaceno el resultado en un
fichero. Lo hago asi para que no se muestre si existe algun permiso
denegado.
        clear
        echo "El numero de ficheros del sistema es: `more temporal.$$$`"
#Muestro el valor del fichero
        rm temporal.$$$ #Borro el fichero.
    fi
    read enterkey
}

#Funcion EspacioEnDisco: Muestra el espacio utilizado en disco.
function EspacioEnDisco
{
    clear
    if [[ $Eleccion == "-u" ]]; then
        carpetausuario=`cat /etc/passwd | grep ^$NomUsuario | cut -d: -f6`
            echo "El espacio en disco ocupado por $NomUsuario es: `du -sh
$carpetausuario | awk '{print$1}'`"
        #Comando "du": Calcula el espacio usado desde determinada carpeta. -
s suma el total. -h te lo pone en formato KB, MB, ...
    else
        #Para calcular el espacio por disco, se podria hacer de la misma
forma que el anterior poniendo que la carpeta es la raiz "/." pero solo sera
valido si estamos conectados como administrador ya que, si no, no se podra
calcular el espacio en carpetas protegidas con permisos. Para solucionar
esto se utilizara la siguiente sintaxis:
        echo "El espacio de disco total ocupado es: `df -h | head -2 | tail
-1 | awk '{print$3}'`"
        #Comando df: Muestra informaci□n sobre el espacio ocupado por el
disco duro. -h te lo pone en formato humano (KB, MB, ...)
    fi
    read enterkey
}

#Funcion TantoPorCiento: Muestra el tanto por ciento ocupado en disco.
```

```
function TantoPorCiento
{
    clear
    if [[ $Eleccion == "-u" ]]; then
        #Primero calculo el espacio ocupado por el usuario en KB.
        carpetausuario=`cat /etc/passwd | grep ^$NomUsuario | cut -d: -f6`
            espacioocupado=$( du -s $carpetausuario | awk '{print$1}' )
        #Ahora calculo el espacio total de disco en KB.
        tamanodisco=$( df | head -2 | tail -1 | awk '{print$2}' )
        #Ahora hacemos una regla de 3 para calcular el tanto por ciento.
        #porcentaje=$(( ( $espacioocupado * 100 ) / $tamanodisco )) #Esta
expresion seria lo normal, pero no muestra decimales. Hay que usar bc.
        porcentaje=$( echo "scale=2;$espacioocupado*100/$tamanodisco" | bc )
#scale=2 utiliza 2 decimales.
        echo "El porcentaje de espacio de disco ocupado por $NomUsuario es:
$porcentaje %"

    else
        echo "El porcentaje de espacio de disco total ocupado es: `df | head
-2 | tail -1 | awk '{print$5}'`"
    fi
    read enterkey
}

################## EMPEZAMOS EL PROGRAMA #################
clear
#Comprobamos que se hayan introducido al menos un parametro.
if [[ $# -eq 0 ]]; then
    echo "Error: No has introducido ningun parametro"
    Error
fi

#Comprobamos que el primer parametro sea "-u" o "-a".
if [[ ( $1 == "-u" ) || ( $1 == "-a" ) ]]; then
    Eleccion=$1
    #Comprobamos que si el primer parametro es "-u", se haya introducido un
segundo parametro.
    if [[ ( $1 == "-u" ) && ( $2 == "" ) ]]; then
        echo "Error: No has introducido ningun usuario"
        Error
    else
        if [[ $1 == "-u" ]]; then
            NomUsuario=$2
            #Comprobamos que el segundo parametro sea un usuario real.
            if id $NomUsuario > /dev/null 2>&1; then
            #Otra forma de hacerlo:
            #NomUsuario=$(grep -w ^$2 /etc/passwd | cut -d : -f1) #-w
significa que contenga exactamente.
```

```
            #if [[ $NomUsuario != "" ]]; then
                NombreUsuario
            else
                echo "Error: El usuario introducido no existe o no esta
conectado"
                Error
            fi
        fi
        NumProcesos
        NomPrimerProceso
        ListadoTodosProcesos
        NumDirectorios
        NumFicheros
        EspacioEnDisco
        TantoPorCiento
    fi
else
    echo "Error: El parametro introducido no es -u ni -a"
    Error
fi
exit
```