Capítulo 6

Programación en el *shell* bash Parte V

6.26. Ejemplos de *scripts*

En este capítulo se presentan una serie de *scripts* de un tamaño medio. Estos *scripts* hacen uso de los operadores vistos en los capítulso anteriores.

1. Suponiendo que en nuestro sistema no existe la orden `mv`, hacemos un *script* que la imita utilizando las órdenes `ln` y `rm`. Este se puede ver en el *script* 6.1.

***Script* 6.1** Simulación de la orden `mv`

```
#!/bin/bash

# move:Emula el funcionamiento de la orden mv

case $# in
2) ;;
*) echo "Uso: $0 fich1 fich2"
   exit;;
esac

if [ ! -f $1 ]
then
  echo $0: $1 no se puede abrir
  exit
fi

if [ -f $2 ]
then
  echo -n "$2 existe. Sobreescribir (s/n)?"
  read conf
  case $conf in
  [sS]*) ;;
  *) exit;;
  esac
fi

ln -f $1 $2 && rm $1
```

2. Este *script* pretende mejorar la interfaz de la orden `cal`. Las mejoras que introduce son:

   - Permite dar el mes mediante su nombre (La orden `cal` sólo permite darlo en forma numérica).
   - Si queremos obtener el calendario de un mes del año actual sólo tenemos que darle el nombre o número correspondiente al mes.

***Script* 6.2** Mejora de la orden `cal`

```
#!/bin/bash

# ncal:Mejora la interfaz de cal

case $# in
0)  m=$(date +%m) ; y=$(date +%Y) ;; # no damos parámetros
1)  m=$1; y=$(date +%Y) ;;           # damos 1 parámetro
*)  m=$1; y=$2 ;;                    # damos 2 parámetros
esac

case $m in
[Ee]ne*)     m=1 ;;
[Ff]eb*)     m=2 ;;
[Mm]ar*)     m=3 ;;
abr*|Abr*)   m=4 ;;
may*|May*)   m=5 ;;
jun*|Jun*)   m=6 ;;
jul*|Jul*)   m=7 ;;
ago*|Ago*)   m=8 ;;
se*|Se*)     m=9 ;;
oct*|Oct*)   m=10 ;;
nov*|Nov*)   m=11 ;;
dic*|Dic*)   m=12 ;;
0[1-9]|10|11|12)  ;; # se da el mes en forma numérica
*)  y=$m; m=" " ;;   # no se da el mes como argumento
esac

/usr/bin/cal $m $y       # se ejecuta la orden cal
```

3. *Script* que muestra un menú con los ficheros del directorio de trabajo para que elijamos uno de ellos y después muestra otro menú con una serie de operaciones a realizar con el fichero elegido previamente.

***Script* 6.3** Operaciones con ficheros

```
#!/bin/bash

PS3="Elija número: "

select i in *
do select j in editar renombrar borrar copiar
  do case $j in
  editar)  ${EDITOR:-emacs} $i
           exit;;
  renombrar)  echo -n "Nuevo nombre: "
              read new
              mv $i $new
              exit;;
  borrar)  rm $i
           exit;;
  copiar)  echo -n "Nombre de la copia: "
           read copia
           cp $i $copia
           exit;;
  *)  echo "Elección incorrecta";;
    esac
  done
done
```

4. Esta es una versión de la orden de UNIX `basename`, que se utiliza para devolver la última parte de un nombre de fichero. También se le puede pasar un sufijo para que lo quite del directorio resultante.

***Script* 6.4** Simulación de la orden `basename`

```
#!/bin/bash

# Comprobación de argumentos
if [ $# -eq 0 -o $# -gt 2 ]
then
  echo "Uso: $0 cadena [sufijo]"
  exit 1
fi

# Obtención del nombre base
BASE=${1##*/}

# Comprobar si se ha dado sufijo
if [ $# -gt 1 ]
then
  #Muestra el nombre base sin el sufijo
  echo ${BASE%$2}
else
  #Muestra el nombre base
  echo $BASE
fi
```

5. El siguiente *script* permite borrar de forma «segura»un fichero. En vez de borrarlo lo que hace es moverlo a un directorio llamado `PAPELERA` y guardarlo ahí comprimido por si nos arrepentimos de haberlo borrado.

***Script* 6.5** Borrado seguro

```
#!/bin/bash

case $# in
0)  echo "Uso: $0 fichero . . ."
    exit;;
*)  ;;
esac

if [ ! -d ~/PAPELERA ]
then
  echo "PAPELERA no existe, lo voy a crear"
  mkdir ~/PAPELERA
elif [ ! -w ~/PAPELERA ]
then
  echo "PAPELERA no tiene permiso de escritura"
  exit
fi

for FICH in $*
do
  if [ ! -f $FICH ]
  then
    echo "$FICH no existe"
    continue
    elif [ ! -r $FICH -o ! -w $FICH ]
    then echo "$FICH no se puede comprimir"
    continue
  else
    echo "Comprimiendo $FICH"
    gzip $FICH
    echo "Moviendo $FICH.gz"
    mv $FICH.gz ~/PAPELERA
  fi
done
```

6. El *script* 6.6 es una variante del 6.5 que borra de forma «segura»los ficheros. En este caso, no comprime los ficheros borrados, pero admite dos opciones, `-d` para indicar el directorio empleado como `PAPELERA` y `-v` que permite mostrar un mensaje en cada paso que va dando el *script*. Este ejemplo permite ver la utilización de los operadores de sustitución de cadenas, estudiados anteriormente.

***Script* 6.6** Utilización de operadores de sustitución

```
#!/bin/bash

# Nos aseguramos de que no está declarada la variable previamente
unset VISUALIZAR

while getopts :vd: arg
do
  case $arg in
  v) VISUALIZAR=SÍ;;
  d) PAPELERA=$OPTARG;;
  :) echo "Error: Falta el argumento de la opción $OPTARG"; exit 1;;
  \?) echo "Error: Opción $OPTARG incorrecta"; exit 1;;
  esac
done

# Determinamos el directorio PAPELERA.
echo ${PAPELERA:=/tmp} > /dev/null

shift $((OPTIND - 1))

# Comprobamos si las variables necesarias están establecidas
echo ${PAPELERA:?Variable no establecida}

if [ ! -d $PAPELERA ]
then
   echo ${VISUALIZAR:+Creamos directorio $PAPELERA}
   mkdir $PAPELERA
fi
if [ ! -w $PAPELERA ]
then
   echo Error: No tengo permiso de escritura
   exit 1
fi
echo ${VISUALIZAR:+ Moviendo ficheros a $PAPELERA}
for FICHERO in $*
do
    if [ -f $FICHERO ]
    then
        echo ${VISUALIZAR:+Moviendo $FICHERO}
        mv $FICHERO $DIRECTORIO
    else
        echo Error: $FICHERO no existe
    fi
done
echo ${VISUALIZAR:+Operación terminada}
```

7. El *script* 6.7 admite dos opciones, `l` o `f`, ambas exclusivas, y uno o varios directorios. Si se introduce la opción `l` realiza un listado de los enlaces simbólicos de los directorios proporcionados. Con la opción `f` muestra los nombres de los ficheros regulares. Si no se le pasa ningún directorio, muestra un menú con los directorios existentes en el directorio de trabajo.

***Script* 6.7** Uso de funciones, `select` y `getopts`

```
#!/bin/bash

declare -i OPC=0
USO="Formato: $0 [ -l | -f ] [ directorio .... ]"

function error() {
    echo -e "$USO \n $2"; exit $1
}

function simbolico() {
    ls -l $1 | awk 'BEGIN {OFS=" -> "
                           print "ENLACES SIMBÓLICOS EN '$1'";
                           print "----------------------------";}
                    /^l/ {print $9, $11}
                    END  {print "--------------------";}'
}

function fichero() {
    ls -l $1 | awk 'BEGIN {print "FICHEROS EN '$1'";
                           print "--------------------";}
                    /^-/ {print $9}
                    END  {print "--------------------";}'
}

function menu() {
    PS3="Escoja una opción: "
    select DIR in $(ls -l | awk '/^d/ {print $9}') "Salir"
    do
        case $DIR in
        Salir) break;;
        *) if [ -d $DIR -a -r $DIR -a -x $DIR ]
           then
            eval $func $DIR
           else
            echo "$DIR no es un directorio o no tiene permisos adecuados"
           fi;;
        esac
    done
}
```

***Script* 2.26** Continuación

```
while getopts :lf opcion
do
    case $opcion in
    l) [ $OPC == 0 ] && OPC=OPC+1 || error 1 "Opciones incompatibles";;
    f) [ $OPC == 0 ] && OPC=OPC+2 || error 1 "Opciones incompatibles";;
    \?) error 2 "Opción incorrecta";;
    :) error 3 "Falta argumento a la opción";;
    esac
done

shift $((OPTIND - 1))

case $OPC in
    0|2) func=fichero;;
    1) func=simbolico;;
    *) error 1 "Opciones incompatibles";;
esac

if [ $# -eq 0 ]
then
    menu
else
    for DIR
    do
        if [ -d $DIR -a -r $DIR -a -x $DIR ]
        then
            eval $func $DIR
        else
            echo "$DIR no es un directorio o no tiene permisos adecuados"
        fi
    done
fi
```

6.27. Ejercicios

1. Escriba un *script* con el siguiente formato:

   `mostrar` [`-p` *n* | `-f` *n*] [*fichero*]

   donde:

   `-p` *n* muestra las primeras *n* líneas del *fichero*.

   `-f` *n* muestra las últimas *n* líneas del *fichero*.

   Si *fichero* no se especifica, entonces debe pedirse al usuario el nombre del fichero. Sólo se permite una opción, y si no se especifica ninguna se mostrarán las primeras 5 líneas.

2. Escriba un *script* en el lenguaje de programación del *shell* bash con el siguiente formato:

   `usuario -b` *login* | `-a` *login* [*GID*]

   que permita dar de alta (`-a`) o de baja (`-b`) usuarios en el sistema.

   Para realizar este *script* necesitará crearse la siguiente estructura:

   - Crear en el directorio de casa un directorio llamado `home`, que será empleado como directorio base donde alojar los directorios de casa de los usuarios que dé de alta.
   - Crear en el directorio de casa un directorio llamado `etc`, y copiar en él los ficheros `/etc/passwd` y `/etc/group` y el directorio `/etc/skel`. Estos ficheros serán empleados como ficheros de usuarios y de grupos en el *script*.

   Con la opción `-b` se da de baja a un usuario del sistema. Para ello deberá comprobar que el usuario existe. En ese caso, se borrará del fichero `~/etc/passwd`, y se borrarán su directorio de casa y los ficheros creados debajo de él.

   La opción `-a` permite dar de alta a un nuevo usuario. La información que se necesita para ello es:

   - Nombre de usuario: Se le debe dar como argumento de la opción. Debe comprobar que es único en el sistema.
   - Contraseña: Debe dejar vacío el campo correspondiente.
   - UID: Se tomará como UID uno más que el máximo que exista entre los usuarios del sistema.

- GID: Se puede dar opcionalmente en la línea de órdenes; si no se proporciona se preguntará. En cualquier caso, debe comprobarse que existe ese grupo.
- Información variada: Se preguntará por el nombre y apellidos del usuario.
- Directorio de casa: El camino del directorio de casa de los usuarios será: `~/home/login`.
- *Shell*: Se tomará como *shell* predeterminado `/bin/bash`.

Una vez introducida la información sobre el usuario en `~/etc/passwd`, deberá crearse el directorio de casa del usuario y copiarse los ficheros de entorno del directorio `~/etc/skel`. Durante estas operaciones debe evitar la aparición de cualquier mensaje de error.

No debe realizar control de permisos sobre los ficheros puesto que el *script* lo debería ejecutar el administrador del sistema.

Al finalizar todas estas operaciones se debe mostrar un mensaje avisando de que es necesario cambiar el propietario y grupo del directorio de casa y de los ficheros copiados, indicando la línea de órdenes que debería introducirse.

3. Escriba un *script* en el lenguaje de programación del *shell* bash con el siguiente formato:

   `grupo -b` *nombre-grupo* | `-a` *nombre-grupo* [*usuario1 ...*]

   que permita dar de alta (`-a`) o de baja (`-b`) grupos en el sistema.

   Para realizar este *script* necesitará crearse la siguiente estructura:

   - Crear en el directorio de casa un directorio llamado `etc`, y copiar en él los ficheros `/etc/passwd` y `/etc/group`. Estos ficheros serán empleados como ficheros de usuarios y de grupos en el *script*.

   Con la opción `-b` se da de baja a un grupo del sistema. Para ello deberá comprobar que el grupo existe. En ese caso, se borrará del fichero `~/etc/group`, y dará un mensaje de aviso con los nombres de login de los usuarios que tenían como grupo principal el eliminado y que tendrán que ser cambiados de grupo.

   La opción `-a` permite dar de alta a un nuevo grupo. La información que se necesita para ello es:

   - Nombre del grupo: Se le debe dar como argumento de la opción. Debe comprobar que es único en el sistema.
   - Contraseña: Debe poner un asterisco en este campo.

- GID: Se tomará como GID uno más que el máximo que exista en el sistema.
- Nombres de usuario: Si no se proporcionan nombres de usuario al llamar al *script*, este campo se dejará vacío. Si se dan, habrá que introducirlos separados por comas.

No debe realizar control de permisos sobre los ficheros puesto que el *script* lo debería ejecutar el administrador del sistema.