CAPÍTULO 6: ADMINISTRACIÓN DEL SISTEMA

6.1. Cree un script que obtenga información de un determinado usuario mediante el comando *id*.

```
#!/bin/bash
# infoUsuario: datos y grupos de un usuario
#

# Comprobación de los argumentos
if [ $# -ne "1" ] ; then
       echo "Uso: `basename $0` usuario"
       exit 2
fi

USUARIO=$1

id $USUARIO 1>/dev/null 2>&1
ERROR=$?

if [ $ERROR -ne 0 ] ; then
  echo "El usuario  $USUARIO no existe"
  exit
fi

#Impresión de información
NOMBRE=`id $USUARIO | cut -f1 -d" "`
echo "(UID) y nombre: $NOMBRE"
GRUPO1=`id $USUARIO | cut -f2 -d" "`
echo "(GID) y grupo primario: $GRUPO1"

if test `id $USUARIO | tr " " "\n" | wc -l` -gt 2 ; then
  GRUPOS2=`id $USUARIO | cut -f3 -d" "`
echo "(GIDs) y grupos secundarios: $GRUPOS2"
fi
```

6.2. Cree el script (*infoGrupo.sh*) que dado un nombre de grupo determine si existe en el sistema, y si es así, presente su nombre, número de grupo (GID), y lista de usuarios que pertenezcan a él, ya sea como grupo primario (consulte */etc/passwd*) o como grupo secundario (consulte */etc/group*).

```
#!/bin/bash
# infoGrupo.sh
# Determina los usuarios en un grupo

# Comprobación de los argumentos
if [ $# -ne "1" ] ; then
       echo "Uso: `basename $0` grupo"
       exit 1
```

```
fi

GRUPO=$1

# Comprobar si el grupo existe previamente (fichero
/etc/group)
cut -d: -f1 /etc/group | grep "$GRUPO" > /dev/null
EXISTE=$?

if [ $EXISTE -ne 0 ] ; then
      echo "El grupo $GRUPO no existe"
      exit 2
fi

#El grupo existe
GID=`grep "$GRUPO:" /etc/group | cut -d: -f3`
echo "El número del grupo $GRUPO es $GID"

# busca usuarios con este grupo primario (fichero
/etc/passwd)
echo "Usuarios en este grupo como primario:"
# corta los campos usuario e id grupo,
# selecciona líneas con $GID al final,
# luego corta el GID, dejando el nombre de usuario
cat /etc/passwd | cut -d: -f1,4 | grep :$GID | cut -d: -f1

# busca usuarios con este grupo secundario (fichero
/etc/group)
echo "Usuarios en este grupo como secundario:"
grep $GRUPO: /etc/group | cut -d: -f4 | tr "," " "
```

6.3. **Utilizando el archivo *`/etc/passwd`* escriba un script (*`listarUsuarios.sh`*) que liste los nombres de usuario, el directorio propio del usuario y el intérprete invocado por defecto de todos los usuarios, ordenados alfabéticamente por nombre de usuario.**

```
#!/bin/bash
# listarUsuarios.sh
# Lista los usuarios del sistema
#
echo "Nombres de usuarios, Directorio propio, intérprete
de comandos"
echo
"============================================================
====="
echo
cat /etc/passwd | cut -d: -f1,6,7 | sort | more
echo
```

6.4. **Utilizando únicamente el archivo *`/etc/group`*, escriba los siguientes scripts.**

- ***`grupo1.sh`*. Lista los nombres y números de grupo y la lista**

de usuarios de cada uno, ordenados por nombre.

```
#!/bin/bash
# grupo1.sh : Listado de grupos y usuarios ordenados
por nombre

clear
echo "Grupos ordenados por nombre:"
echo ------------------------------------------------
---------
echo "login:numero_de_grupo:lista de usuarios"
echo ------------------------------------------------
---------
cat /etc/group | cut -d: -f1,3,4 | sort | more -18
echo ------------------------------------------------
---------
```

- ***grupo2.sh.*** **Igual que el anterior, pero ordenados por número de grupo.**

```
#!/bin/bash
# grupo2.sh : Listado de grupos ordenados por número

echo "Grupos por numero:"
echo ------------------------------------------------
---------
echo "login:numero_de_grupo:lista de usuarios"
echo ------------------------------------------------
---------
cat /etc/group | cut -d: -f1,3,4 | sort -t: -n | more
-18
echo ------------------------------------------------
---------
```

Los dos últimos scripts presentan la información en concordancia a como aparece en el fichero original. Puede ser confusa de leer, por lo que se recomienda formatear la salida mediante awk para facilitar su lectura.

6.5. **A la hora de crear usuarios por lotes se puede utilizar el comando *newusers*, el cual crea usuarios a partir de un fichero de texto que contiene la descripción de cada usuario, en el mismo formato que el fichero *passwd* (*usuario:password:uid:gid:comentario:directorio_home:shell*). Por ejemplo:**

homer:HcZ600a9:1008:1000:Homer Simpson:/home/homer:/bin/bash

Cree un fichero de texto (*familia_Simpson.txt*) con el contenido que se lista a continuación y cree el lote de usuarios.

homer:HcZ600a9:1008:1000:Homer Simpson:/home/homer:/bin/bash
marge:1enz733N:1009:1000:Marge Simpson:/home/marge:/bin/csh
bart:1y5eJr8K:1010:1000:Bart Simpson:/home/bart:/bin/ksh
lisa:VGz638i9:1011:1000:Lisa Simpson:/home/lisa:/bin/sh
maggie:5lj3YGQo:1012:1000:Maggie Simpson:/home/maggie:/bin/bash

Este ejercicio es meramente informativo, por lo que carece de complicación. Una vez creado el fichero de texto con el contenido indicado anteriormente, se procederá a la creación del lote de usuarios mediante:

```
$ newusers familia_Simpson.txt
```

6.6. Cree un script *copiabin.sh* que mueva todos los archivos ejecutables del directorio hacia el subdirectorio *bin* del directorio propio del usuario, muestre los nombres de los ejecutables que mueve e indique cuántos ha movido o que no ha movido ninguno. Si el directorio *bin* no existe, deberá ser creado.

```
#!/bin/bash
# copiabin.sh: copia archivos ejecutables hacia $HOME/bin

# si el directorio bin no existe lo crea
if [ ! -d $HOME/bin ] ; then
  mkdir $HOME/bin
fi

# copia de archivos
N=0           # contador de archivos copiados
for ARCH in *
do
  if [ -x $ARCH -a -f $ARCH ]  # ejecutable y archivo
común (no directorio)
  then
    cp $ARCH $HOME/bin
    echo " *  $ARCH fue copiado a $HOME/bin"
    N=`expr $N + 1`
  fi
done
if [ $N -eq 0 ]
then
  echo "No se copió ningún archivo"
```

```
else
  echo "Fueron copiados $N archivos"
fi
```

6.7. **Cree el script *permisosRecursivo.sh* que dado un listado de directorios recorra su contenido y cambie los permisos de todos los archivos y subdirectorios que encuentre de modo recursivo.**

```
#!/bin/bash
# permisosRecursivo.sh
# Cambia los permisos recursivamente a los ficheros de los
directorios indicados

# Listado de directorios a procesar
Directorios[0]="/home/Videos"
Directorios[1]="/home/Musica"
# Puede añadir más directorios si lo necesita
# Directorios[2]="/home/programas"

# Control de fin
export IFS=$'\n'

# Se itera a través de los directorios
for x in ${Directorios[@]}
do
      # Búsqueda de directorios y subdirectorios
      for i in $(find $x -type d)
      do
            # Cambio de permisos
            chmod -c 775 $i
            #-c sólo imprime tras cambio
            #Puede silenciar el proceso eliminado dicho
modificador
            #puede cambiar si desea el propietario
            #chown -c usuario:grupo $i
      done

      # Búsqueda de archivos
      for i in $(find $x -type f)
      do
            # Cambio de permisos
            chmod -c 664 $i
            #puede cambiar si desea el propietario
            #chown -c usuario:grupo $i
      done
done
```

El script anterior no realiza los cambios en aquellos archivos que contengan espacios en el nombre. Para solventarlo se puede utilizar algo del estilo de:

```
find ~/ -type f -exec chmod 755 '{}' \;
```

6.8. **Los archivos *core* se producen como resultado de programas que no han terminado su ejecución de modo correcto. Son generados por el sistema operativo con la información necesaria para hacer una depuración de los problemas que han ocasionado el final inesperado del programa (volcado de memoria de cuando ocurrió el problema, estado del programa y del procesador en el momento del fallo). Estos archivos suelen ser de gran tamaño y muchas veces se acumulan en el disco . Cree un script que localice todos los ficheros *core* a partir del directorio en el que se encuentre y elimínelos.**

```
$ find / -name core -exec rm -f {} \;# listarUsuarios.sh
```

6.9. **Localice todos los ficheros *.bak* a partir del directorio en el que se encuentre y elimínelos pidiendo confirmación al usuario.**

```
$ find . -type f -name "*.bak" -exec rm -i {} \;
```

6.10. **Cambie los permisos a todos los ficheros del directorio *~/código* a 0700.**

```
$ find ~/codigo -exec chmod 0700 {} \;
```

6.11. **Localice los ficheros, a partir de un directorio dado, con el bit *setgid* a *on*.**

```
#!/bin/bash
# findsetgidon.sh
# Localiza los ficheros con el bit setgid activo que
cuelguen del directorio dado

if [ $# -ne "1" ] ; then
       echo "Uso: `basename $0` directorio"
       exit 1
fi

DIRECTORIO="$1"
find $DIRECTORIO -xdev -type f -perm +g=s –print
```

6.12. **Dado un fichero de origen y un listado de archivos a excluir, desarrolle un script que elimine todos los ficheros de dicho directorio con una fecha de modificación superior a siete días, pidiendo confirmación de todos los ficheros a eliminar.**

```
#!/bin/bash
```

```
# removefiles.sh
#
DIR=/home/usuario
LISTA=/home/usuario/DELETE_LIST
EXCLUSIONES=/home/usuario/EXCLUSIONES

# El script realiza lo mismo que:
# find $DIR -type f -mtime +7 | grep ^./ | grep -v -f
$EXCLUSIONES -exec rm {} \;
# Pero pide confirmación a la hora de eliminar los
ficheros.

#Confirma la eliminación del archivo
function CONFIRMAR
{
       echo "Se va a eliminar $1"
       echo "Desea continuar? -s/n"
       read OPCION
       if [ $OPCION = 's' ] ; then
              rm -f $1
       elif [ $CHOICE = 'n' ] ; then
              exit 1
       else
              echo "Opción inválida"
              exit 2
       fi
}

#Busca los archivos a eliminar
function FIND_FILES
{
       cd $DIR
       find -type f -mtime +7 | grep ^./ | grep -v -f
$EXCLUSIONES > $LISTA

       for i in `cat $LISTA`
       do
              CONFIRMAR $i
       done
}


#Localizar los ficheros a eliminar
FIND_FILES
Exit
```

6.13. **Las copias de seguridad, no sólo de los datos personales, sino de la configuración son imprescindibles en todo sistema. Por ello, desarrolle un script que realice una copia de los directorios de datos del sistema *(/home, /root, /usr/local/httpd*) y de configuración del mismo *(/etc, /var/lib*). Una vez a la semana, los domingos, la copia será completa y los demás días incremental. Las copias se realizarán en un disco externo que montará en */mnt/backup.***

```
#!/bin/bash
# backupSystem.sh
# Copia de los archivos fundamentales del sistema

DATOS="/home /root /usr/local/httpd"
CONFIGURACION="/etc /var/lib"
LISTA="/tmp/backlist_$$.txt"

#montado del disco destino de la copia
mount /mnt/backup
#obtención de los datos de la fecha
set $(date)

#tenga cuidado con el formato/idioma de la fecha
if test "$1" = "dom" ; then
        # Copia semanal completa
        # datos
        tar cfz "/mnt/backup/DATOS/DATOS_full_$6-$2-
$3.tgz" $DATOS
        rm -f /mnt/backup/DATOS/DATOS_diff*
        # configuración
        tar cfz
"/mnt/backup/CONFIGURACION/CONFIGURACION_full_$6-$2-
$3.tgz" $CONFIGURACION
        rm -f
/mnt/backup/CONFIGURACION/CONFIGURACION_diff*
else
        # Copia diaria incremental
        # datos
        find $DATOS -depth -type f \( -ctime -1 -o -mtime
-1 \) -print > $LISTA
        tar cfzT "/mnt/backup/DATOS/DATOS_diff_$6-$2-
$3.tgz" "$LISTA"
        rm -f "$LISTA"
        # configuración
        find $CONFIGURACION -depth -type f  \( -ctime -1 -
o -mtime -1 \) -print > $LISTA
        tar cfzT
"/mnt/backup/CONFIGURACION/CONFIGURACION_diff_$6-$2-
$3.tgz" "$LISTA"
        rm -f "$LISTA"
fi

#Desmontar el disco de copias
umount /mnt/backup
```

Programación cron. Dada su funcionalidad, puede programar este script para su ejecución diaria mediante cron, ubicándolo directamente en el directorio /etc/cron.daily o bien programándolo directamente en la crontab

Archivos con espacios en el nombre. Estructuras como tar zcvf archivo.tgz `find /home –ctime 1 –depth –print` producen errores en aquellos archivos que contenga espacios en su nombre. Esto se solventa

redirigiendo la salida de la búsqueda a un fichero o mediante la opción –T de tar.

6.14. Cree un script que lleve a cabo una copia de seguridad de todos los directorios y ficheros listados en un fichero dado (*~/.mybackup)* y que envíe el resultado por correo electrónico con formato *tar.gz*.

Nota*: Para poder enviar datos adjuntos en un correo debe tener instalado *mutt.

```
#!/bin/bash
# backupMail.sh
#
# Realiza una copia de seguridad de los ficheros indicados
en el archivo
# ~/.mybackup, lo comprime en formato .tar.gz y
# lo envía por correo electrónico

# Indique en ese archivo los ficheros/directorios a
salvaguardar
#
# Uso   : ./backupMail
#


FICHERO=~/.mybackup
FECHA=`date +"%d-%m-%Y"`
FORMATO="`echo $USER.$HOSTNAME`.$FECHA.tar.gz"
TAR=`which tar`

# Configuración del correo
DESTINATARIO="usuario@dominio.com"
ASUNTO="Backup (`echo $USER @ $HOSTNAME`) en `date`"
MENSAJE=~/tmp/mybackup.txt
MATT=~/tmp/$FORMATO

# Los datos se almacenan en su directorio tmp y no en
/tmp
[ ! -d ~/tmp ] && mkdir ~/tmp || :
if [ -f $FICHERO ]; then
       IN="`cat $FICHERO | grep -E -v "^#"`"
else
       echo "$FICHERO no existe"
       exit 3
fi

if [ "$IN" == "" ]; then
       echo "$FICHERO está vacío"
       echo "Por favor, añada ficheros/directorios a copiar
en el backup"
       exit 2
fi
```

```
$TAR -zcf ~/tmp/$FORMATO $IN >/dev/null

# Creación del mensaje de correo
echo "Backup realizado con éxito. Por favor, vea el
documento adjunto." > $MENSAJE
echo "" >> $MENSAJE
echo "Backup: $FORMATO" >> $MENSAJE
echo "" >> $MENSAJE

# Sólo se pueden enviar adjuntos si tiene mutt instalado
which mutt > /dev/null
if [ $? -eq 0 ]; then
      # Enviar el fichero con el backup adjunto
      mutt -s "$ASUNTO" -a "$MATT" $DESTINATARIO <
$MENSAJE
else
      echo "mutt no se encuentra en el sistema"
      echo "No se puede enviar el fichero adjunto"
fi

# Borrar ficheros temporales
/bin/rm -f $MATT
/bin/rm -f $MENSAJE
```

6.15. **Para completar el ejercicio anterior, realice un script que añada ficheros y directorios al archivo que contiene el listado de ficheros y directorios a salvaguardar.**

```
#!/bin/bash
# myBackupAddFiles.sh
# Añade ficheros/directorio a ~/.mybackup, que es el
fichero contenedor
# de todos los ficheros/directorios a salvaguardar
#
# Uso   : ./myBackupAddFiles ~/public_html/
#

FICHERO=~/.mybackup
MYH=~
CWD=`pwd`
SRC=$1

if [ "$SRC" == "" ]; then
  echo "Debe indicar un fichero/directorio"
  exit 1
fi

# Si el fichero de listado no existe, se crea
[ ! -f $FICHERO ] && touch $FICHERO || :

# Se comprueba que el fichero/directorio indicado exista
if [ ! -f $SRC ]; then
   if [ ! -d $SRC ]; then
      echo "$SRC no existe"
      exit 2
```

```
   fi
fi

# Comprueba que no se dupliquen entradas en el listado
cat $FICHERO | grep -w $SRC > /dev/null
if [ "$?" == "0" ]; then
   echo "$SRC ya existe en $FICHERO"
   exit 3
fi

# Se añade el fichero/directorio al listado
echo "$SRC" >> $FICHERO
echo "$SRC añadido a $FICHERO"
```

6.16. **En algunas ocasiones cuando se trabaja con transmisiones FTP son necesarias comunicaciones con un nivel de seguridad mucho más alto, sobre todo si la información que se intercambia es comprometida. Por lo tanto, se deben aplicar todos los esfuerzos para evitar intrusiones o robo de información. Un modo de asegurar las conexiones FTP es mediante GPG (GNU Privacy Guard - software de cifrado en sistemas Linux/ Unix).**
GnuPG es un programa de software de cifrado híbrido que utiliza una combinación convencional de criptografía de claves simétricas para la velocidad, y la criptografía de clave pública para facilitar el intercambio seguro de claves, por lo general mediante el uso de la clave pública del destinatario para cifrar una clave de sesión que sólo se utiliza una vez. Este modo de operación es parte del estándar OpenPGP y ha sido parte del PGP desde su primera versión.
Desarrolle un script que permita el envío de datos entre equipos haciendo uso de GPG para asegurar la transferencia.

```
#!/bin/bash
# secureBackup.sh
#
# Realiza un backup del directorio indicado a un servidor
FTP remoto,
# subiendo la información de forma segura, para lo cual
utiliza gpg con
# el propósito de encriptar el fichero .tar.gz resultante
antes de
# enviarlo.
#
# Para su correcta ejecución debe tener instalado:
# - /usr/bin/ncftpput
# - /bin/tar
# - /usr/bin/mail
# - /usr/bin/gpg
#
# El script manda un correo de confirmación al
```

```
administrador
# informando del éxito o fracaso del proceso
#
# Instalación:
# Personalice el script de acuerdo a sus parámetros de
configuración
# indicando el servidor FTP, usuario, contraseña, etc
#
# Deberá generar las claves gpg para su usuario e importar
la clave pública
# de cara a la encriptación de los ficheros.

# PARÁMETROS DE CONFIGURACIÓN

# Directorios a copiar (puede indicar varios separándolos
con espacios)
BACKUP="/home"

# Servidor FTP
FTPH="servidor"

# Usuario FTP
FTPU="ftpusername"

# Password FTP
FTPP="secreta"

# gpg userID
GPGU="gpguserID"

# Directorio remoto, blanco para directorio por defecto
# Si no existe será creado automáticamente por ncftpput
# Asegúrese de tener los permisos necesarios para ello
FTPD="backup/"

# Directorio temporal para almacenar el fichero tar.gz y
procesarlo
TMPD="/tmp"

# Email del administrador
MTO="admin@mail.com"
# Asunto
MSUB="Informe de Backup de $(hostname)"
# Información de administrador
ADMIN_INFO="Para contactar con el administrador:
admin@mail.com"

# Ubicación de los binarios
NCFTP="/usr/bin/ncftpput"
TAR="/bin/tar"
MAILC="/usr/bin/mail"
GPG="/usr/bin/gpg"

##################################################################
#############
# NO CAMBIE NADA DE ABAJO
##################################################################
#############
```

```
FILE="$(hostname).$(date +"%d-%m-%Y").tar.gz"
OUT="$TMPD/$FILE"
FOUT="$OUT.gpg"
MFILE="/tmp/ftpout.$$.txt"
MESS=""

if [ ! -x $TAR ]; then
  echo "$TAR comando no encontrado, $ADMIN_INFO"
  exit 1
fi

if [ ! -x $NCFTP ]; then
  echo "$NCFTP comando no encontrado, $ADMIN_INFO"
  exit 1
fi

if [ ! -x $GPG ] ; then
  echo "$GPG comando no encontrado, $ADMIN_INFO"
  exit 1
fi

$TAR -zcf $OUT $BACKUP
if [ $? -ne 0 ]; then
   MESS="$TAR falló al crear el fichero. No se enviará
nada al servidor FTP $FTPH"
else
   # Encriptar el fichero .tar.gz antes de su envío
   $GPG -e -r $GPGU -o $FOUT $OUT
   $NCFTP -m -u "$FTPU" -p "$FTPP" "$FTPH" "$FTPD" "$FOUT"
   OSTAT="$?"
   case $OSTAT in
        0) MESS="OK.";;
        1) MESS="Imposible conectar al servidor remoto
$FTPH.";;
        2) MESS="Imposible conectar al servidor remoto
$FTPH - timed out.";;
        3) MESS="Fallo en la transferencia.";;
        4) MESS="Fallo en la transferencia - timed out.";;
        5) MESS="Fallo al cambiar de directorio.";;
        6) MESS="Fallo al cambiar de directorio - timed
out.";;
        7) MESS="URL incorrecta.";;
        8) MESS="Error de uso. Actualice su versión de
ncftpput ($NCFTP)";;
        9) MESS="Error en el archivo de configuración.";;
        10)MESS="Inicialización incorrecta de
librerias.";;
        11) MESS="Fallo en el inicio de sesión.";;
        *) MESS="Error desconocido, $ADMIN_INFO";;
   esac
fi

>$MFILE
echo "Estado de Backup de $(hostname) en $(date):"
>>$MFILE
echo "" >>$MFILE
echo "Fichero de Backup : $FOUT" >>$MFILE
echo "Servidor FTP : $FTPH" >>$MFILE
```

```
echo "Estado del Backup : $MESS" >>$MFILE
echo "" >>$MFILE
echo "-- Generado automáticamente por $(basename $0)"
>>$MFILE

# Envío de mail al administrador
$MAILC -s "$MSUB" $MTO <$MFILE
# Eliminación de archivos temporales
[ -f $MFILE ] && rm -f $MFILE || :
[ -f $FOUT ] && rm -f $FOUT || :
[ -f $OUT ] && rm -f $OUT || :
```

6.17. **Dada la potencialidad que aporta el uso de claves en scripts que hacen uso de SSH, permitiendo llevar a cabo conexiones sin necesidad de introducir contraseñas, es necesario un mecanismo para proceder al intercambio de las llaves entre equipos. Por lo tanto, cree un script que envíe la clave pública de SSH a un equipo remoto.**

```
#!/bin/bash
# enviarKey.sh

# Uso: enviarKey.sh equipo_remoto

# Añade la clave pública SSH en la lista authorized_keys
de un servidor remoto
# (Se asume que por defecto todas las claves se encuentran
en su ubicación por defecto)

set -v                          # salida verbose
usuario="usuario"         # coloque aquí el nombre de
usuario
servidorRemoto=$1         # Asigna el primer comando al
$servidorRemoto


# Se envía la clave pública introducida a través de un
cauce
# Se verifica la existencia del listado en el equipo
remoto y se concatena allí
cat ~/.ssh/id_dsa.pub | ssh ${usuario}@${servidorRemoto}
"touch ~/.ssh/authorized_keys && cat - >>
~/.ssh/authorized_keys"

exit 0
```

Esto mismo se puede conseguir de un modo mucho más sencillo como sigue:

```
$ ssh-copy-id –i ~/.ssh/id_dsa.pub usuario@servidor
```

6.18. **Suponiendo que dispone de claves SSH generadas de cara a una comunicación sin contraseñas entre equipos, desarrolle un script que lleve a cabo el replicado de un servidor local, salvo los ficheros de log relativos al acceso (*"*access.log"*) y a los errores (*"*error.log"*) en dos discos diferentes de un servidor remoto.**

```
#!/bin/bash
# rsyncRemoto.sh
#
# Replicación (Backup) de un servidor en remoto mediante
rsync


# Directorio local a copiar
LOCALBAKPOINT=/disk3
LOCALBAKDIR=/remote/home/httpd/

# Datos del servidor SSH remoto
# usuario
SSHUER=usuario

# Host/servidor IP
SSHSERVER=10.10.11.12

#Directorio de backup remoto
SSHBACKUPROOT=/disk2.backup/hot/

rsync --exclude '*access.log*' --exclude '*error.log*' -
avz -e 'ssh ' ${SSHUER}@${SSHSERVER}:${SSHBACKUPROOT}
${LOCALBAKPOINT}${LOCALBAKDIR}

# Creación de un fichero de log del proceso en
/var/log/messages
[ $? -eq 0 ] && logger 'RSYNC BACKUP : OK' || logger
'RSYNC BACKUP : FALLO!'

# Duplicación del backup en /disk1 y /disk2
SRC=${LOCALBAKPOINT}${LOCALBAKDIR}
DST="/disk1/remote /disk2/remote"
for d in $DST
do
 [ ! -d $d ] && mkdir -p $d || :
 rsync -avr $SRC $d
done
```

6.19. **Una opción que se puede valorar, a la hora de llevar a cabo la conexión SSH es la utilización de claves creadas ex proceso para dicha comunicación, que una vez acabada la misma se eliminan. Se podrían definir como *claves temporales*. Basándose en esta premisa, implemente un script que cree unas claves SSH, las envíe al equipo remoto de cara a llevar a cabo la comunicación y, una vez finalizada la misma, las elimine.**

```
#!/bin/bash
# claveTemporal.sh

# Conexión remotas ssh con clave temporal


keyTemporal="1" #usar de clave temporal o no
identificadorSesion="option" #usado para indicar que
ejecución es
servidorRemoto="usuario@servidor.com"
usuario="root"

if [ "$keyTemporal" -eq "1" ] ; then
      key="/root/.ssh/tempsshkey_id_${identificadorSesion}
"
      publickey="/root/.ssh/tempsshkey_id_${identificadorS
esion}.pub"
      authkeys="/root/.ssh/authorized_keys"
      ssh-keygen -t rsa -q -f ${key} -N ''
      chmod 600 ${key}
      chmod 600 ${publickey}
      pubkeystring=`cat ${publickey} | sed -e 's/\//.*/g'`
      ssh-copy-id -i ${publickey} root@${servidorRemoto}
2>&1 1>/dev/null
      sshstring="ssh -i ${key}
${usuario}@${servidorRemoto}"
else
      sshstring="ssh ${usuario}@${servidorRemoto}"
fi

${sshstring} "uptime"

if [[ -n "$keyTemporal" ]] ; then
      echo "Eliminando claves ssh temporales..."
      ${sshstring} "cat ${authkeys} | sed -e
'/$pubkeystring/d' > ${authkeys}"
      rm ${key} ${publickey}
fi
```

El script anterior crea unas nuevas claves en un único fichero, con lo que se previene la destrucción de las claves anteriores, y les asigna los permisos apropiados para su copiado en la máquina remota. A la hora de ejecutar el script (en este ejemplo el comando *uptime*) se captura el comando *ssh* en una variable, lo que permite ejecutarlo con o sin claves, minimizando el código empleado.
Con la condición *if* final se eliminan los posibles cambios sobre claves que se hayan podido llevar a cabo.