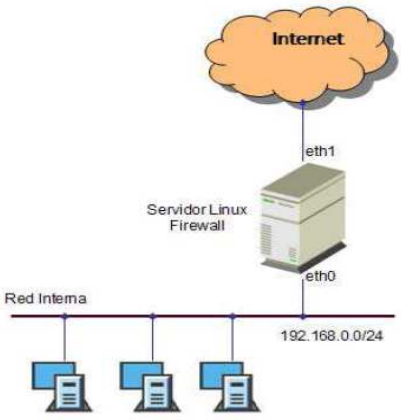
1. Desarrolle un script que genere el fichero */etc/network/interfaces,* para lo cual debe solicitar toda la información a través de la consola.

```
#!/bin/bash
# crearInterfaces.sh
# Creación del fichero /etc/network/interfaces con datos
# obtenidos desde la consola

#Fichero de interfaces
ficheroInterfaces=/etc/network/interfaces

#Obtención de los datos
echo -n "Inserte la dirección IP inicial:"
read ipstart
echo -n "Inserte la dirección IP final:"
read ipend
echo -n "Inserte Gateway:"
read gw
echo -n "Inserte la subred:"
read subnet
echo -n "Inserte Netmask: "
read net
echo -n "Inserte alias: "
read ali

#Asignación de valores
firstIp=`echo "${ipstart%.*}"`
lastIpStart=`echo "${ipstart##*.}"`
lastIpEnd=`echo "${ipend##*.}"`
dif=`echo $(($lastIpEnd-$lastIpStart))`
ip=$lastIpStart
for ((i=$ali;i<=$ali+$dif;i++))
{
   echo "auto eth0:$i" >>$ficheroInterfaces
   echo "iface eth0:$i inet static" >>$ficheroInterfaces
   echo "address $firstIp.$ip" >>$ficheroInterfaces
   echo "network $subnet" >>$ficheroInterfaces
   echo "netmask $net" >>$ficheroInterfaces
   echo "gateway $gw" >> $ficheroInterfaces
   echo " " >> $ficheroInterfaces
   let ip++
}
```

2. Suponiendo el siguiente esquema de red:



Desarrolle un script para configurar el firewall del servidor para que realice las siguientes tareas:

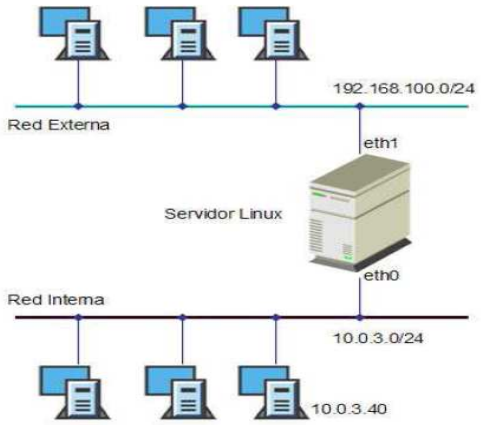
- Permita todos los accesos al servidor Linux desde la red interna.

- Desde internet permitir sólo las conexiones a los servicios HTTP, HTTPS Y SMTP.

```
#!/bin/bash
# firewall1.sh
# Reglas de firewall con el siguiente cometido:
# - Permitir todos los accesos al servidor Linux desde la red interna
# - Desde Internet permitir sólo conexiones a los servicios http, https, y
SMTP

#Se informa del proceso
echo " Configurando el Firewall..."

#Eliminación de las reglas existentes.
echo " -> Eliminado Reglas Existentes..."
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

echo " -> Estableciendo reglas por defecto..."
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

#Definición de las reglas a utilizar
export LAN=eth0
export WAN=eth1

# Se empieza a filtrar
echo " -> Configurando red interna..."
#Se permite el tráfico en localhost
/sbin/iptables -A INPUT -i lo -j ACCEPT
# Al firewall se tiene acceso desde la red local
iptables -A INPUT -s 192.168.0.0/24 -i ${LAN} -j ACCEPT

# Ahora con la regla FORWARD se filtra el acceso de la red local
# al exterior. A los paquetes que no van dirigidos al
# propio firewall se les aplican reglas de FORWARD

# Se acepta que vayan a puertos http (80)
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p tcp --dport 80 -j ACCEPT
# Se acepta que vayan a puertos https (443)
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p tcp --dport 443 -j
ACCEPT
# Se acepta que vayan a puertos smtp (25)
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p tcp --dport 25 -j ACCEPT

# Se acepta que consulten los DNS (necesario para navegar)
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p tcp --dport 53 -j ACCEPT
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p udp --dport 53 -j ACCEPT

# Y se deniega el resto.
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -j DROP

# Se hace enmascaramiento de la red local
# y se activa el BIT DE FORWARDING
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ${WAN} -j MASQUERADE

# Con esto se permite hacer forward de paquetes en el firewall, esto es,
# que otras máquinas puedan salir a través del firewall.
# ACTIVADO A NIVEL KERNEL. EN CASO CONTRARIO -> DESCOMENTAR
# echo 1 > /proc/sys/net/ipv4/ip_forward

## Se crean los accesos indeseados del exterior:
# Nota: 0.0.0.0/0 significa: cualquier red

echo " -> Bloqueando accesos externos..."
# Cerramos el rango de puerto bien conocido
iptables -A INPUT -s 0.0.0.0/0 -p tcp -dport 1:1024 -j DROP
```

```
iptables -A INPUT -s 0.0.0.0/0 -p udp -dport 1:1024 -j DROP

echo " Proceso finalizado"
echo " Verifique su aplicación con: iptables -L -n"

# Fin del script
```

3. Suponga que dispone de dos redes locales con varios equipos y un servidor Linux haciendo funciones de routing entre ambas redes, tal y como muestra el siguiente esquema:



Desarrolle un script para configurar el firewall del servidor Linux teniendo en cuenta que debe realizar las siguientes tareas:

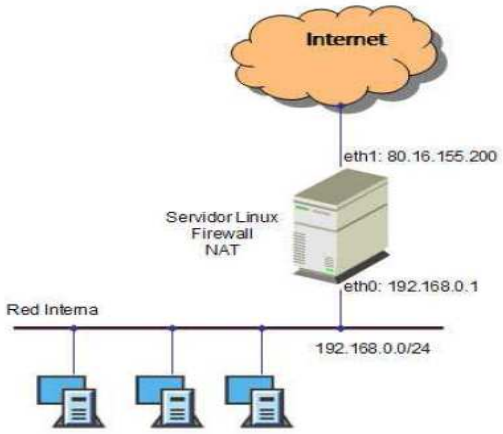
- Realizar un enmascaramiento de la red interna.
- Redireccionar las peticiones Web a la máquina 10.0.3.40, permitiendo el tráfico específico en el firewall.
- Permitir sólo el tráfico de salida de tipo WEB y DNS, y el resto se rechaza.
- Realizar un registro (Log) de intentos de acceso desde la red externa al firewall y a los equipos internos.
- El firewall sólo admitirá conexiones SSH desde la red interna.
- Limitar el tráfico de control ICMP para evitar ataques DoS (hasta un máximo de cinco peticiones por segundo).

```
#!/bin/bash
# firewall2.sh

#Se informa del proceso
echo "Configurando el firewall..."

#Se eliminan las reglas existentes.
echo " -> Eliminado Reglas Existentes..."
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

#Se definen las interfaces que se usan
export INTERNA=eth0
export EXTERNA=eth1
```

```
echo " -> Estableciendo reglas por defecto..."
iptables -P INPUT DROP        # descartar entradas al firewall
iptables -P OUTPUT DROP       # descartar salidas del firewall
iptables -P FORWARD DROP      # descartar reenvíos a través del firewall
iptables -t nat -P PREROUTING  ACCEPT
iptables -t nat -P POSTROUTING  ACCEPT

# Permitir tráfico a localhost (firewall)
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# SNAT (enmascaramiento de lo que sale de red interna 10.0.3.0/24) hacia
exterior
echo " -> Enmascaramiento de salida..."
iptables -t nat -A POSTROUTING -s 10.0.3.0/24 -o ${EXTERNA} -j MASQUERADE

# DNAT (redireccionamiento servicio HTTP [puerto 80] a red interna)
echo " -> Redireccionado tráfico HHTP..."
iptables -t nat -A PREROUTING -i ${EXTERNA} -p tcp --dport 80 \
                              -j DNAT --to-destination 10.0.3.40:80

# Limitar tráfico ICMP (permitir hasta un máximo de 5 peticiones/segundo)
echo " -> Limitando tráfico de control ICMP (5 peticiones/seg)..."
iptables -A INPUT -p icmp -m limit --limit 5/second -j ACCEPT
iptables -A OUTPUT -p icmp -m limit --limit 5/second -j ACCEPT
iptables -A FORWARD -p icmp -m limit --limit 5/second -j ACCEPT

# FILTRADO ENTRADA RED INTERNA
# - permitir paso de servicios redireccionados (peticiones + sus
respuestas)
#   [necesario al usar DROP por defecto]
echo " -> Filtrando entrada a la red interna..."
iptables -A FORWARD -i ${EXTERNA} -d 10.0.3.40 -p tcp --dport 80 -j ACCEPT
iptables -A FORWARD -i ${INTERNA} -s 10.0.3.40 -p tcp --sport 80 \
                    -m state --state ESTABLISHED,RELATED -j ACCEPT

# - log de otros accesos a red interna (se denegarán por defecto)
echo " -> Creando registro de accesos a la red interna..."
iptables -A FORWARD -i ${EXTERNA} -d 10.0.3.0/24 -j LOG --log-prefix
"Acceso red
interna:"

# # FILTRADO SALIDA RED INTERNA
echo " -> Filtrando salida de la red interna..."

# - se permiten conexiones salientes HTTP + sus respuestas
iptables -A FORWARD -i ${INTERNA} -s 10.0.3.0/24 -p tcp --dport 80 -j
ACCEPT
iptables -A FORWARD -o ${INTERNA} -d 10.0.3.0/24 -p tcp --sport 80 \
                    -m state --state ESTABLISHED,RELATED -j ACCEPT

# - se permiten las consultas DNS salientes + sus respuestas
iptables -A FORWARD -i ${INTERNA} -s 10.0.3.0/24 -p tcp --dport 53 -j
ACCEPT
iptables -A FORWARD -o ${INTERNA} -d 10.0.3.0/24 -p tcp --sport 53 \
                    -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A FORWARD -i ${INTERNA} -s 10.0.3.0/24 -p udp --dport 53 -j
ACCEPT
iptables -A FORWARD -o ${INTERNA} -d 10.0.3.0/24 -p udp --sport 53 \
                    -m state --state ESTABLISHED,RELATED -j ACCEPT

# - se rechazan los demás intentos de salida (por defecto se haría DROP)
iptables -A FORWARD -i ${INTERNA} -s 10.0.3.0/24 -j REJECT --reject-with
icmp-portunreachable

# FILTRADO CONEXIONES HACIA EL FIREWALL
echo " -> Permitiendo tráfico SSH de red interna..."
# - se permite entrada y salida de SSH desde red interna, resto bloqueado
por política por defecto
iptables -A INPUT -i ${INTERNA} -s 10.0.3/24 -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -o ${INTERNA} -d 10.0.3/24 -p tcp --sport 22 -j ACCEPT

# - log de los intentos de acceso al firewall desde exterior (serán
denegados)
echo " -> Creando log de accesos externos al firewall..."
```

```
iptables -A INPUT -i ${EXTERNA} -j LOG --log-prefix "Acceso firewall:"

# Habilitar retransmisión de paquetes (LA TENEMOS DESHABILITADA EN KERNEL)
#DESCOMENTAR EN CASO CONTRARIO
#echo 1 > /proc/sys/net/ipv4/ip_forward
```

4. Suponga que dispone de una red local con varios equipos y un servidor Linux haciendo las funciones de routing de dicha red local e Internet, tal y como muestra el siguiente esquema:



Desarrolle un script de modo que el firewall cumpla las siguientes restricciones:

- Proporcione acceso a Internet a los equipos internos a través de SNAT con la dirección del servidor Linux.
- No debe permitir a los equipos internos:
  - Hacer ping a un equipo externo.
  - Acceder por FTP a un equipo externo.
  - Acceder por SSH al servidor Linux.
- No debe permitir al servidor Linux:
  - Recibir ninguna petición a los puertos privilegiados (1-1024) desde la red local.
- Se permite todo lo demás (política ACCEPT).

```
#!/bin/bash
#firewall3.sh

# SCRIPT PARA ACTIVAR EL ENRUTAMIENTO Y EL NAT

# Reglas de NAT con el siguiente cometido:
# - Proporcionar acceso a Internet a los equipos internos a través de SNAT
con
#   la dirección IP pública de nuestro servidor Linux
# - No permitir a los equipos internos
#   - Hacer un ping externo
#   - Acceder por ssh al servidor Linux
# - No permitir al servidor Linux
#   - Recibir ninguna petición a los puertos privilegiados (1-1024) desde la
red local
# - Se permite todo lo demás (política ACCEPT)
```

```
#Se informa del proceso
echo " Configurando el Firewall..."

#Se eliminan las reglas existentes.
echo " -> Eliminado Reglas Existentes..."
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

echo " -> Estableciendo reglas por defecto..."
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

#Se definen las reglas a utilizar
export LAN=eth0
export WAN=eth1

# Comienza el filtrado
echo " -> Configurando red interna..."
#permitimos el tráfico en localhost
/sbin/iptables -A INPUT -i lo -j ACCEPT

# Al firewall se deniega el acceso ssh (22) desde la red local
echo " -> Denegando acceso SSH al servidor desde la red interna..."
iptables -A INPUT -s 192.168.0.0/24 -i ${LAN} -p tcp --dport 22 -j DROP

# Ahora con regla FORWARD se filtra el acceso de la red local
# al exterior. A los paquetes que no van dirigidos al
# propio firewall se les aplican reglas de FORWARD

# Se deniegan que vayan a puertos ftp (21)
echo " -> Denegando FTP a equipos externos..."
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p tcp --dport 21 -j DROP
# Se deniega el ping a equipos externos (protocolo icmp)
echo " -> Denegando ping a equipos externos..."
iptables -t filter -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p icmp --icmp-
type echo-request -j REJECT
iptables -t filter -A FORWARD -s 192.168.0.0/24 -i ${LAN} -p icmp --icmp-
type echo-replay -j REJECT

# Se acepta el resto.
iptables -A FORWARD -s 192.168.0.0/24 -i ${LAN} -j ACCEPT

# Se hace el enmascaramiento de la red local
# y se activa el BIT DE FORWARDING
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ${WAN} -j MASQUERADE

# Con esto se permite hacer forward de paquetes en el firewall, o sea
# que otras máquinas puedan salir a través del firewall.
# ACTIVADO A NIVEL KERNEL. EN CASO CONTRARIO -> DESCOMENTAR
# echo 1 > /proc/sys/net/ipv4/ip_forward

echo " -> Proporcionando acceso a internet con SNAT..."
iptables -t nat -A POSTROUTING -s 192.168.0.0/24 -o ${WAN} -j SNAT --to
80.16.155.200

echo " -> Bloqueando accesos internos a puertos privilegiados..."
# Cerramos el rango de puertos de 1 a 1024 desde la red interna
iptables -A INPUT -s 192.168.0.0/24 -p tcp -dport 1:1024 -j DROP
iptables -A INPUT -s 192.168.0.0/24 -p udp -dport 1:1024 -j DROP

echo " Proceso finalizado"
echo " Verifique su aplicación con: iptables -L -n"

# Fin del script
```

5. Realice un script de elimine toda la configuración establecida por el ejercicio anterior.

   El proceso de desactivación consiste únicamente en borrar todas las tablas y establecer las reglas por defecto:

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

# SCRIPT PARA DESACTIVAR EL ENRUTAMIENTO Y EL NAT

#Se informa del proceso
echo " Configurando el Firewall..."

#Se eliminan las reglas existentes.
echo " -> Eliminado Reglas Existentes..."
iptables -F
iptables -X
iptables -Z
iptables -t nat -F

echo " -> Estableciendo reglas por defecto..."
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT
iptables -t nat -P PREROUTING ACCEPT
iptables -t nat -P POSTROUTING ACCEPT

echo " Proceso finalizado"
echo " Verifique su aplicación con: iptables -L -n"
```

6. Desarrolle un script que cree copias de seguridad todas las bases de datos MySQL completas, salvo aquellas que se indiquen para su exclusión de la copia. El fichero resultante sólo podrá ser accedido por el root.

```
#!/bin/bash
# backupExclusiones.sh
#
# Realiza una copia de seguridad de todas las bases de datos MySQL salvo
las
# incluidas en un listado de exclusiones.
# El fichero resultante sólo podrá ser accedido por el root.

# Datos de conexión
USUARIO="usuario"
PASS="password"
SERVER="localhost"

# Ubicación de los comandos necesarios
MYSQL="$(which mysql)"
MYSQLDUMP="$(which mysqldump)"
CHOWN="$(which chown)"
CHMOD="$(which chmod)"
GZIP="$(which gzip)"

# Directorio destino del backup
DEST="/backup"

# Subdirectorio para las copias de mysql
MBD="$DEST/mysql"

# Obtención del nombre del host local
HOST="$(hostname)"

# Formateo de la fecha (dd-mm-yyyy)
NOW="$(date +"%d-%m-%Y")"

# Fichero para almacenar el backup actual
FILE=""
# Almacenamiento para el listado de bases de datos
DBS=""

# Bases de datos a excluir
EXCLUIRBDS="test"

# Si no existe el fichero destino del backup, se crea
[ ! -d $MBD ] && mkdir -p $MBD || :
```

```
# Sólo el root tendrá acceso al mismo
$CHOWN 0.0 -R $DEST
$CHMOD 0600 $DEST

# Obtener el listado de todas las bases de datos
DBS="$($MYSQL -u $USUARIO -h $SERVER -p$PASS -Bse 'show databases')"

# Exclusión de las bases de datos indicadas
for db in $DBS
do
    skipdb=-1
    if [ "$EXCLUIRBDS" != "" ];
    then
   for i in $EXCLUIRBDS
   do
       [ "$db" == "$i" ] && skipdb=1 || :
   done
    fi

    if [ "$skipdb" == "-1" ] ; then
   FILE="$MBD/$db.$HOST.$NOW.gz"
   # Todo el trabajo se lleva a cabo en un cauce
   # Conexión a mysql mediante mysqldump para seleccionar la base de datos
   # la cual se envía comprimida mediante gzip al directorio de destino
        $MYSQLDUMP -u $USUARIO -h $SERVER -p$PASS $db | $GZIP -9 > $FILE
    fi
done
```

7. Cree un script que permita añadir bases de datos, usuarios y contraseñas a un servidor MySQL, y que admita además acceso el mismo de modo inmediato mediante la asignación de los permisos pertinentes.

```
#!/bin/bash
# addBD.sh
#
# Script que permite añadir bases de datos, usuarios y password a una base
de
# datos MySQL.
# Permite acceso "al vuelo" a la vez que se crea la base de datos.

# Parámetros para la creación de la base de datos
BD="$1"
USUARIO="$2"
PASS="$3"
HOSTS="$4"
PERMISOS="$5"

## Ruta de los binarios de MySQL ##
mysql="/usr/bin/mysql"

## Datos de conexión ##
ADMINUSER='root'
ADMINPASS='MySQL-PassWord'
SERVER='localhost'

# Se comprueba la entrada de al menos tres permisos: BD, usuario, password
[[ $# -le 2 ]] && { echo "Uso: $0 'NombreBD' 'Usuario' 'Contraseña'
['remoto1|remoto2|remotoN'] ['Permisos']"; exit 1; }

# Si no se indican permisos, se asignan todos
[[ -z "${_PERMISOS}" ]] && _PERMISOS="ALL"

# Construcción de las consultas MySQL
_uamq="${mysql} -u "${ADMINUSER}" -h "${SERVER}" -p'${ADMINPASS}' -e
'CREATE DATABASE ${BD};'"
_upermq1="${mysql} -u "${ADMINUSER}" -h "${SERVER}" -p'${ADMINPASS}' -e
\"GRANT ${PERMISOS} ON ${BD}.* TO ${USUARIO}@localhost IDENTIFIED BY
'${PASS}';\""

# Ejecución de las consultas MySQL
$_uamq
$_upermq1
```

```
# Lectura en bucle de los host a los que asignar permiso de acceso
IFS='|'
for  i in ${HOSTS}
do
   _upermq2="${mysql} -u "${ADMINUSER}" -h "${SERVER}" -p'${ADMINPASS}' -e
\"GRANT ${PERMISOS} ON ${BD}.* TO ${USUARIO}@${i} IDENTIFIED BY
'${PASS}';\""
   $_upermq2
done
```

El modo de uso del script es muy sencillo. Por ejemplo, mediante:

```
./addBD.sh personas gonzalo puga
```

Se creará la base de datos *personas*, a la cual se añade el usuario *gonzalo* con password *puga*.

Si además se añade un conjunto de direcciones IP:

```
./addBD.sh personas gonzalo puga '192.168.1.5|192.168.1.11'
```

Dichas direcciones serán autorizadas para acceder al servidor de base de datos con el usuario *gonzalo*.

Para finalizar, se pueden indicar los permisos que dicho usuario tendrá sobre la base de datos mediante:

```
./addBD.sh personas gonzalo puga '192.168.1.5|192.168.1.11'
'SELECT,INSERT,UPDATE,DELETE'
```

8. Implemente un script que realice una copia de una base de datos MySQL y la envíe a otro servidor haciendo uso de SSH. Las condiciones que debe cumplir el script son las siguientes:

   - Exportar la estructura y los datos de la base de datos en un archivo SQL con MySqlDump.
   - Comprimir el archivo SQL.
   - Envío del archivo a un servidor remoto mediante SSH (opcional).
   - Guardar también la copia en un directorio local. Si el número de copias almacenadas supera un valor (variable *MaxBackups*) se eliminarán las versiones más antiguas.

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

# Script para realizar backup de la base
# de datos MySQL y su envío mediante ssh


# Configuración de la base de datos
DbUser=usuario
DbHost=127.0.0.1
DbPass=password
```

```
DbName=baseDatos

# Configuración del número máximo de backups a guardar
MaxBackups=7

# Configuración de comandos
MySqlDump_cmd=/usr/bin/mysqldump
Tar_cmd=/bin/tar
LOCAL_SCP_CMD=/usr/bin/scp
LOCAL_SSH_CMD=/usr/bin/ssh

# Configuración de directorios
DirTmp=/tmp
DirBackup=/backups

# Configuración archivo de backup
HOY=`date +"%d-%m-%Y_%H_%M_%S"`
FileNameBackup=backup_$HOY

# Configuración archivos temporales y backups (OJO! no cambiar la extensión
del archivo)
FileTmpBackup=$DirTmp/$FileNameBackup.sql
FileBackup=$DirBackup/$FileNameBackup.tgz
BackupsFilePath=$DirTmp/backupspaths

# Configuración SSH para enviar backup a otro servidor
ENABLED_REMOTE_BACKUP_SSH=1
CLAVE_SSH=/usr/local/bin/ssh_keys/id_rsa
IP_REMOTE_SSH=192.168.0.1
PORT_REMOTE_SSH=22
USER_SSH=root
HOY_SSH=`date +"%u"`
FileNameBackup_SSH=backup_$HOY_SSH.tgz
REMOTE_FILE_SSH=/ backups/ $FileNameBackup_SSH
LOCAL_FILE_SSH=$FileBackup

# Inicio del proceso de backup

echo "Generando backup de la base de Datos MySQL..."

# Comprobación de existencia de los directorios necesarios
if [ ! -d "$DirTmp" ]; then
    echo "Error, el directorio temporal '$DirTmp' no existe."
    exit
fi

if [ ! -d "$DirBackup" ]; then
    echo "Error, el directorio para las copias '$DirBackup' no existe."
    exit
fi

# Crear archivo SQL con estructura y datos de la base de datos MySql
$MySqlDump_cmd -u $DbUser --host $DbHost --password=$DbPass $DbName >
$FileTmpBackup
chmod 777 $FileTmpBackup

# Comprime el script de backup de la base de datos MySql
$Tar_cmd czvf $FileBackup $FileTmpBackup &> /dev/null
rm $FileTmpBackup

# Borra los backups antiguos
echo "Realizado limpieza de backups antiguos..."

find $DirBackup -name '*.tgz' | sort -r > $BackupsFilePath
chmod 777 $BackupsFilePath

i=1

while read file; do
    if [ $i -gt $MaxBackups  ]; then
        echo "Eliminando backup antiguo: $file "
        chmod 777 $file
        rm $file
    fi

    i=`expr $i + 1`
```

```
done < $BackupsFilePath

rm $BackupsFilePath

# Envío del backup al servidor remoto si la función esta activada
if [ $ENABLED_REMOTE_BACKUP_SSH = 1 ]; then
       echo "Copiando backup al servidor remoto..."

      $LOCAL_SCP_CMD -P $PORT_REMOTE_SSH \
      -i $CLAVE_SSH $LOCAL_FILE_SSH \
      $USER_SSH@$IP_REMOTE_SSH:$REMOTE_FILE_SSH
fi

echo "Se ha completado el backup de la base de Datos MySql."
```