# CAPÍTULO 5: PROGRAMACIÓN DE TAREAS

## 5.1. Programe una tarea *cron* que a las 20 horas imprima en el terminal el siguiente mensaje "*Descanso de 5 minutos*".

```
0 20 * * * echo "Descanso de 5 minutos"
```

## 5.2. Programe una tarea que escriba cada 10 minutos los siguientes datos en el fichero */var/log/historial.txt.*

```
19:44:11 up   2:36,   3 users,   load average: 0,10, 0,18,
0,22
USER     TTY      FROM               LOGIN@   IDLE   JCPU
PCPU WHAT
gonzalo  tty1                       17:11    2:31m 37.07s
0.00s xinit /etc/X11/
gonzalo  pts/0    :0.0              17:13    0.00s  5.11s
0.00s w
gonzalo  pts/1    :0.0              19:08   19:55   0.20s
0.20s bash
```

La información anterior se refiere a los usuarios que han iniciado sesión en el sistema y su actividad en el mismo. Dicha información se puede obtener mediante el comando ***w***. Por lo tanto, tan sólo habrá que redireccionar la salida de dicho comando al fichero indicado mediante un script:

```
#! /bin/bash
# historial.sh
# Guarda información de la actividad de los usuarios

fichero="/var/log/historial.txt"
echo "------------------------------------------------" >>
$fichero
w >> $fichero
echo "------------------------------------------------" >>
$fichero
```

Y posteriormente se debe crear la entrada adecuada en el crontab:

```
*/10 * * * * /pathAlScript/historial.sh
```

## 5.3. Programe una tarea que apague el equipo todos los días a las 23:59.

```
59 23 * * * shutdown –h now
```

5.4. **Escriba un script que se ejecute cada día a las 1:00 horas y que cumpla las siguientes condiciones:**

- **Si es domingo se genera una copia completa de los directorios *_/home_*, */etc/* y */root* en la carpeta */copia_seguridad/completa*.**
- **Si es cualquier otro día se realiza una copia diaria del directorio */etc* y se guarda en la carpeta */copia_seguridad/incremental/mes_dia_año* donde se indica la fecha del sistema.**
- **Las copias de seguridad se ejecutan con el comando *tar*.**
- **Siempre que se genera una copia en la misma carpeta se genera el fichero *fecha.txt* que contiene la fecha exacta del sistema.**

```
#!/bin/bash
#backup condicional

COMPLETO="/home /etc /root"
DIARIA="/etc"

set $(date)

if test "$1" = "dom" ; then
      tar cfz
/copia_seguridad/completa/backup_completo_$3-$2-$6.tgz
$COMPLETO
      date >/copia_seguridad/completa/fecha.txt
else
      tar -Puvf
/copia_seguridad/incremental/backup_incremental_-$3-$2-
$6.tgz $DIARIA
      date >/copia_seguridad/incremental/fecha.txt
fi
```

Ya sólo queda definir la tarea en *crontab* con la siguiente entrada:

```
0 1 * * * /pathAlScript/backup.sh
```

**Nota.** Mediante el comando *set* se obtienen los campos del comando

*date* parametrizados, de modo que dichos campos se podrán utilizar a conveniencia.

5.5. **El demonio *mcelog* se encarga de monitorizar errores en sistemas Linux x86, tanto de 32 como de 64 bits, especialmente errores relacionados con la memoria y la CPU. Este demonio *mcelog* almacena la información de los posibles errores en el fichero de log */var/log/mcelog*. En versiones de kernel antiguas no estaba configurado como un demonio, si no que se debía programar como una tarea *cron*. Por ello, de cara a trabajar con sistemas antiguos, programe un script que notifique los errores hardware de un servidor mediante correo electrónico.**

```
#!/bin/bash
# monitorHardware.sh
# Detecta errores hardware mediante el demonio mcelog y
los notifica vía mail

LOGGER=/usr/bin/logger
FILE=/var/log/mcelog

EMAIL="gonzalo@server1"
ASUNTO="Error HARDWARE- $(hostname)"
MENSAJE="Warning - Errores hardware detectado en
$(hostname) @ $(date). Vea el fichero de log
/var/log/mcelog para más detalles."
OK_MENSAJE="$0 - OK: NO se han detectado errores
hardware."
WARNING_MENSAJE="$0 - ERROR: Error de hardware detectado."

die(){
      echo "$@"
      exit 1
}

warn(){
      echo $MENSAJE | email -s "${ASUNTO}" ${EMAIL}
      $LOGGER "$WARNING_MENSAJE"
}

[ ! -f "$FILE" ] && die "Error - No existe el fichero
$FILE o mcelog no está configurado"
[ $(grep -c -i "hardware error" $FILE) -gt 0 ] && warn ||
$LOGGER $OK_MENSAJE
```

Dada la importancia de esta monitorización se debería ejecutar cada minuto:

```
* * * * * /pathAlScript/monitorHardware.sh
```

## 5.6. Programe un script que monitorice todos los procesos en ejecución en el sistema cada 30 segundos durante 3 minutos.

Para la realización de este script no es necesaria la programación mediante *cron* sino una temporización interna dentro del mismo. Para ello se ciclará 3 veces y se dormirá el proceso por periodos de 30 segundos.

```
#!/bin/bash
# monitorProcesos.sh
# Monitoriza los procesos del sistema cada 30 segundos
durante 3 minutos

for r in 1 2 3 4 5 6
do
       #Se muestran los procesos del sistema
       echo "**************************** x^x^x
****************************"
       ps -e
       echo "**************************** x^x^x
****************************"
       #dormir 30 segundos
       sleep 30
done
```

## 5.7. Cree un script que monitorice la ejecución de MySQL en un servidor y reinicie dicho servicio en caso de que se encuentre detenido. Envíe un email al usuario informando de dicha eventualidad.

```
#!/bin/bash
# monitorMySQL.sh
#
# Reinicia MySQL si el servidor no está funcionando y
envía un email al
# usuario informando de dicha eventualidad.
# Conviene programarlo mediante cron para controlar el
servidor MySQL

# CONFIGURACION DEL SERVIDOR DE MYSQL #

# Usuario root/administrador de MySQL
MUSER="root"
# Contraseña de administrador/root de MySQL
MPASS="password-de-root"
# Servidor MySQL
MHOST="localhost"
# Ubicación del demonio MySQL para servidores
Debian/Ubuntu.
# Modificar de acuerdo a su distribución.
```

```
MSTART="/etc/init.d/mysql start"
# Email de notificaciones
EMAILID="notificaciones@donde.com"
# path al programa de mail
MAILCMD="$(which mail)"
# path a mysqladmin
MADMIN="$(which mysqladmin)"

#### NO CAMBIE NADA DE LA PARTE INFERIOR ####
MAILMESSAGE="/tmp/mysql.fail.$$"

# Comprobación del servicio
$MADMIN -h $MHOST -u $MUSER -p${MPASS} ping 2>/dev/null
1>/dev/null
if [ $? -ne 0 ]; then
      echo "" >$MAILMESSAGE
      echo "Error: El servidor MySQL Server no está
respondiendo las peticiones de ping">>$MAILMESSAGE
      echo "Hostname: $(hostname)" >>$MAILMESSAGE
      echo "Fecha & Hora: $(date)" >>$MAILMESSAGE
      # Se intenta reiniciar MySQL
      $MSTART>/dev/null
      # Se comprueba su reinicio
      o=$(ps cax | grep -c ' mysqld$')
      if [ $o -eq 1 ]; then
            sMess="MySQL Server reiniciado correctamente"
      else
            sMess="MySQL Server FALLÓ al ser reiniciado"
      fi
      # Notificación vía mail del proceso
      echo "Estado Actual: $sMess" >>$MAILMESSAGE
      echo "" >>$MAILMESSAGE
      echo "*** Email automático generado por $(basename
$0) ***" >>$MAILMESSAGE
      echo "*** Por favor, no responda a este email de
notificaciones ***" >>$MAILMESSAGE
      # send email
      $MAILCMD -s "MySQL server" $EMAILID < $MAILMESSAGE
else # MySQL está en ejecución por lo que no se hace nada
      :
fi
# Eliminación del fichero de correo temporal
rm -f $MAILMESSAGE
```

Dada la importancia de esta monitorización se debería ejecutar cada minuto:

```
* * * * * /pathAlScript/monitorMySQL.sh
```

## 5.8. Desarrolle un script que monitorice el porcentaje de utilización de los discos del sistema. En caso de que alguno de ellos supere una ocupación del 90% envíe un email al administrador informando de tal hecho.

```
#!/bin/bash
# monitorDiscos.sh
# Monitoriza el uso de los discos del sistema.
# Si el uso supera un umbral de alerta, informa vía mail
al administrador

# Establecer el email de notificaciones
ADMIN="email@notificaciones.com"
# Establecer el nivel de alerta
ALERT=90

df -H | grep -vE '^S.archivos|tmpfs|cdrom' | awk '{ print
$5 " " $1 }' | while read output;
do
  usep=$(echo $output | awk '{ print $1}' | cut -d'%' -f1
)
  partition=$(echo $output | awk '{ print $2 }' )
  if [ $usep -ge $ALERT ]; then
    echo "Disco sobreutilizado \"$partition ($usep%)\" en
$(hostname) con fecha $(date)" |
     mail -s "Alerta: Disco sobreutilizado $usep" $ADMIN
  fi
done
```

**Nota.** Este script utiliza comparación de patrones para la eliminación de las líneas de texto devueltas por el comando ***df.*** Dichas cadenas dependen del sistema y del idioma, por lo que puede necesitar modificarlas para evitar errores al comparar el porcentaje de utilización del disco con el nivel de alerta.

## 5.9. Cree un script que recoja un mensaje predefinido por el usuario para enviárselo al mismo como recordatorio por correo posteriormente. Para el envío del mensaje haga uso de la programación *at.*

```
#!/bin/bash
# recordatorio.sh Manda un correo de recordatorio con el
# mensaje predefinido por el usuario

echo "Introduzca su mensaje recordatorio.
Finalice con una línea conteniendo punto (.)
(O presione Ctrl-C o DEL para salir.)"

while :
do
    read MESSAGE
    if [ "$MESSAGE" = "." ]
    then
        break
    else
        echo $MESSAGE >> $HOME/Msgs/message.$$
```

```
    fi
done

cat << !!
Introduzca el momento en que
quiere recibir el mensaje, por ejemplo:

      0815am Jan 24
      8:15am Jan 24
      now + 1 day
      5 pm Friday

Y presione Intro.

!!
read TIME

echo "at $TIME mail $LOGNAME $HOME/Msgs/message.$$"

at $TIME  << !!
mail $LOGNAME < $HOME/Msgs/message.$$
rm -f $HOME/Msgs/message.$$
!!
exit 0
```

El script enviará un mensaje al usuario en el momento que éste defina. Para el almacenamiento temporal del fichero se ha creado un directorio temporal en la cuenta de usuario (*/$HOME/Msgs*) lo que dota de mayor seguridad al proceso (aunque se podría utilizar algún fichero temporal del sistema como puede ser */usr/tmp)*. Una vez enviado el correo el mensaje se elimina de dicho directorio.

A continuación puede ver un ejemplo de uso:

```
programacion@shell:~$ ./recordatorio.sh
Introduzca su mensaje recordatorio.
Finalice con una line conteniendo punto (.)
(O presione Ctrl-C o DEL para salir.)
Prueba de recordatorio
.
Introduzca el momento en que
quiere recibir el mensaje, por ejemplo:

      0815am Jan 24
      8:15am Jan 24
      now + 1 day
      5 pm Friday

Y presione Intro.

now + 1 min
at now + 1 min mail gonzalo
/home/gonzalo/Msgs/message.10769
warning: commands will be executed using /bin/sh
```

```
job 4 at Sat Jul 23 13:37:00 2011
programacion@shell:~$ atq
4      Sat Jul 23 13:37:00 2011 a gonzalo
programacion@shell:~$ atq
Tiene correo nuevo en /var/mail/Gonzalo
```

Y puede ver, además, como el contenido del correo recibido es el esperado:

```
Return-Path: <gonzalo@server1>
X-Original-To: gonzalo@server1
Delivered-To: gonzalo@server1
Received: by server1 (Postfix, from userid 1000)
       id 56612201697; Sat, 23 Jul 2011 13:37:00 +0200
(CEST)
To: <gonzalo@server1>
X-Mailer: mail (GNU Mailutils 2.1)
Message-Id: <20110723113700.56612201697@server1>
Date: Sat, 23 Jul 2011 13:37:00 +0200 (CEST)
From: gonzalo@server1 (gonzalo)
Status: RO
Content-Length: 23
Lines: 1

Prueba de recordatorio
```