# BASH

THE BOURNE-AGAIN SHELL

Angel Luis Calvo

ESCUELA IT

# En esta sesión:

- Copias de seguridad
- Ejecución al inicio
- Script endemoniado
- Temporizadores

Angel Luis Calvo


## Angel Luis Calvo

angelonx@gmail.com


ESCUELA IT


# Copias de seguridad

Angel Luis Calvo

ESCUELA IT

# Copias de seguridad

Las copias de seguridad son imprescindibles en cualquier sistema informático. Pueden ser la única vía para resolver ciertos problemas de seguridad, y para mitigar otros.
Las diferentes distribuciones de Linux tienen sus propias herramientas de backup, y las hay añadidas, también con entorno gráfico:

- Bacula
- Simple backup
- fwbackups
- BackupPC
- …

Algunas con limitaciones, otras más profesionales

Por ejemplo:
Copias de Ubuntu


# Copias de seguridad

Para línea de comandos, y para scripts, tenemos, entre otras, las utilidades de compresión de archivos.
La más usada: tar

Es el comando más antiguo, probablemente, que está pensado para archivar copias de seguridad.

Por sí solo, *tar* solo empaqueta (concatena archivos, y carpetas) sin comprimir. Para esto trabaja en conjunto con gzip y otros comandos de compresión.

# Copias de seguridad

Sintaxis de *tar*:
tar *opciones fichero ficheros*
El “fichero” es el fichero *tar* que obtenemos, ficheros es el conjunto que queremos empaquetar/comprimir.
Ejemplo:
tar -cf *archivo.tar *.txt* → crea (c) el empaquetado de los ficheros txt

Más sobre sus opciones:
https://linux.die.net/man/1/tar
https://es.wikipedia.org/wiki/Tar

# Copias de seguridad

Un posible ejemplo, para respaldar un directorio:

```
fecha=$(date +%m_%d_%Y)
destino="/mount/copias/copia_web_($_fecha).tgz"
tar cfz $destino /var/www/miweb/
```

Al hacer la copia con la ruta completa, al descomprimir nos dará un aviso, elimina el raíz / para evitar un posible desastre, se puede imponer con el parámetro -P.

# Ejercicio

Crear una tarea que realice copias de seguridad de la carpeta del curso en /mnt/copias/, todos los días a última hora de la tarde (19:00). Un día copia completa, otros seis incrementales. Se llamarán back1 las de la primera semana, back2 la segunda, etc.

```
#!/bin/bash
#contaremos el número de copias en destino, para empezar las nuevas
#Una completa y seis incrementales
destino=/mnt/copias
metadatos=$destino/marca_copia.met #archivo de metadatos con las marcas de tiempo para incrementos
origen=/home/jefe/curso/
fecha=$(date +%Y-%m-%d-%H-%M) #ponemos horas y minutos para poder hacer pruebas
contador=1
for ((num=1; num<=10; num++)); do #hará hasta 10 semanas, podemos poner las que queramos
        copias=$(ls -1 $destino | grep back_$contador | wc -l)
                if [ $copias -lt 7 ] ; then
```

---------- Continúa en la siguiente----------

# Ejercicio

---------- Viene de la anterior ----------

```
                tar cpzf $destino/back_"$contador"_"$fecha" -g "$metadatos""$contador" $origen
                # La p indica que se salven los permisos de archivo
                #con -g indicamos que genere un archivo con metadatos
                #temporales para que haga incrementales
                echo Realizado
                exit
        else
                contador=$(($contador + 1))
        fi
done

#Para el cron: 00 19 * * *
```


ESCUELA IT

# Copias de seguridad

Restauración:

tar xzf /mnt/copias/back1 -g metadatos.inc -C restaurados

x → para extraer

z → formato gzip

/mnt/copias/back1 → copia de seguridad para restaurar

-g metadatos.inc → metadatos de marca horaria para restaurar incremental

-C restaurados → cambio de carpeta donde descomprimir

Orden: primero la completa, luego las siguientes incrementales, por orden de creación

# Copias de seguridad

Otra herramienta es rsync.
Se puede utilizar para sincronizar datos entre dos sitios remotos.
Sintaxis:
rsync [*opciones*] *origen* [*destino*]; si no se pone destino, lista el origen

Ejemplo para sincronizar dos carpetas locales:
rsync -r d1/ d2 → la r indica recursividad, si hay subdirectorios los copia si no se pone la barra /, se copia d1 dentro de d2. Si solo hay que copiar archivos, no hace falta ningún parámetro.

# Copias de seguridad

Unas opciones muy utilizadas:
rsync -avzh *txt textos/ → Guardará los txt en la carpeta textos, con -a conserva los permisos y otros metadatos, con -z comprime (v de verbose y h de humano)

Con esto, la próxima vez que se ejecute el comando, sólo se transmitirán los cambios del origen.

Se puede utilizar la opción -n (--dry-run) para que solo haga una simulación.

# Copias de seguridad

Para sincronizar carpetas remotas:
rsync -avzh ~/curso/local/ usuario@192.168.15.24:~/copias → copiamos el contenido de una carpeta local en un equipo remoto, en su carpeta copias.
Otra alternativa:
rsync -avzh usuario@192.168.15.24:~/copias/  ~/copias/ → a la inversa, hacemos respaldo en el equipo local de lo que hay en el servidor

# Copias de seguridad

Entre los parámetros de *rsync* hay opciones muy interesantes, como las de excluir archivos, o eliminarlos, tanto del origen como del destino.

Una opción buena es utilizar *rsnapshot*, utilidad de copias, basada en la propia *rsync*.

Más info:
https://linux.die.net/man/1/rsync
https://aplicacionesysistemas.com/rsnapshot-backups-en-gnulinux-1/

# Copias de seguridad

Para automatizar por SSH: tendremos que saltar la petición de autenticación
Es necesario crear un par de claves asimétricas en nuestro equipo, y exportar la pública al servidor. (Se supone que tenemos usuario en él)
En nuestro equipo:

```
ssh-keygen -t rsa
```

...
Your identification has been saved in /home/jefe/.ssh/id_rsa
Your public key has been saved in /home/jefe/.ssh/id_rsa.pub → la clave pública
...

# Copias de seguridad

Para exportar la clave pública: (se supone que nuestro usuario en el servidor cuenta con directorio ~/.ssh)
Usamos el comando ssh-copy-id:
ssh-copy-id -i ~/.ssh/id_rsa.pub jefe@192.168.15.24
Nos pedirá, por última vez, la clave y ya está, con ese usuario podemos conectarnos sin contraseña en el servidor por SSH.

# Ejercicio

Hacer un script que sincronice cada día, a las 8:00, la carpeta del curso con un servidor externo, mediante SSH.

```
#!/bin/bash
carpeta=~/curso/
usuario=jefe
rsync -az -e "ssh -p 33030" "$carpeta" "$usuario"@192.168.15.24:/home/jefe/copias/
# -p nº es el puerto de conexión que tenga ssh en el servidor, si no se pone es el 22
#Debe estar instalado rsync en ambos equipos
#Para el cron: 00 8 * * *
```



# Copias de seguridad

Otros tipos de copias más específicas:
La copia de dispositivos enteros, por ejemplo, clonación de discos o copia binaria de sectores de disco.
Comando dd:
dd if=/dev/sdb of=/dev/sdc → copia el disco b en el c
dd if=/dev/sda of=~/backupMBR bs=512 count=1 → copia el MBR (1 bloque de 512 bytes)
dd if=~/backupMBR of=/dev/sda bs=512 count=1 → restaura lo anterior

# Copias de seguridad

Copias de bases de datos.
Por ejemplo, mysql:
Se utiliza el comando mysqldump
Sintaxis
mysqldump [*opciones*] *base_datos* [*tablas*] > copia.sql → para una BBDD pudiendo indicar tablas
mysqldump [*opciones*] --databases(-B) *bases_datos* > copia.sql → para varias BBDD
mysqldump [*opciones*] [--all-databases][-A] > copia.sql → todas las BBDD

# Copias de seguridad

Ejemplos:
mysqldump -h 192.168.15.24 -u root -pClave -A  > copia_total.sql → copia de todas las BBDD del servidor con ese usuario y clave

mysqldump datos_empresa --ignore-table=clientes > copia_empresa.sql → copia de la base de datos *datos_empresa*, excluyendo la tabla clientes

Desde un equipo externo es necesario contar con un cliente mysql (apt install mysql-client) y que el servidor acepte conexiones desde fuera.

Más info:
https://geekytheory.com/como-permitir-el-acceso-remoto-a-una-base-de-datos-mysql

# Ejercicio

Automatizar copias de seguridad de la base de datos completa del servidor 192.168.15.24, desde nuestro equipo.

```
#!/bin/bash
# definimos las variables
usuario=jefe
clave=Clave
servidor=192.168.15.24
directorio=/home/jefe/mysql_copias/ #tenemos que tenerlo creado
fecha=$(date +%d-%b-%Y)
# ejecución del volcado de la base de datos
mysqldump –u $usuario –p$clave –h $servidor -A > “$directorio”back_$fecha.sql


#Quedaría definir el cron
```

# Copias de seguridad

Más opciones:

mysqlbinlog → para leer archivos binarios, puede servir para hacer copias incrementales
(activar log_bin en my.cnf)

Otras opciones: usar aplicaciones estilo XtraBackup
También existe la opción de utilizar SSH.

Más info:
http://download.nust.na/pub6/mysql/doc/refman/5.0/es/mysqlbinlog.html

# Ejercicio

Automatizar copias de seguridad de una base de datos del servidor 192.168.15.24, desde nuestro equipo, copiando el script en el servidor.
Usaremos dos scripts, uno lanzará al otro, se puede aprovechar el lanzador para otros.



```
#!/bin/bash
# definimos las variables
usuario=jefe
clave=Clave
servidor=localhost
directorio=/home/jefe/mysql_copias/ #debe estar creado en el servidor
bd=fop2
fecha=$(date +%d-%b-%Y)
mysqldump -u $usuario -p$clave -h $servidor $bd > "$directorio"back_"$bd"_"$fecha".sql
```

***lanzadera.sh***

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

---------- Continúa en la siguiente----------


ESCUELA IT

# Ejercicio

---------- Viene de la anterior ----------

```
scp -P 33030 48_mysql_lanzada.sh jefe@192.168.15.24:/home/jefe
ssh jefe@192.168.15.24 -p 33030  "/home/jefe/48_mysql_lanzada.sh"
#Quedaría definir el cron
```


ESCUELA IT


# Ejecución al inicio

Angel Luis Calvo

ESCUELA IT

# Ejecución al inicio

Anteriormente vimos cómo la cláusula @reboot de *cron* nos sirve para lanzar scripts al inicio del sistema.
Hay otras formas de hacerlo.
El archivo /etc/rc.local (o /etc/rc.d/rc.local) era históricamente un archivo donde colocar comandos y scripts para ejecutar en el inicio. En distribuciones modernas está siendo sustituido por demonios del sistema, se puede consultar su estado con systemctl status rc-local.

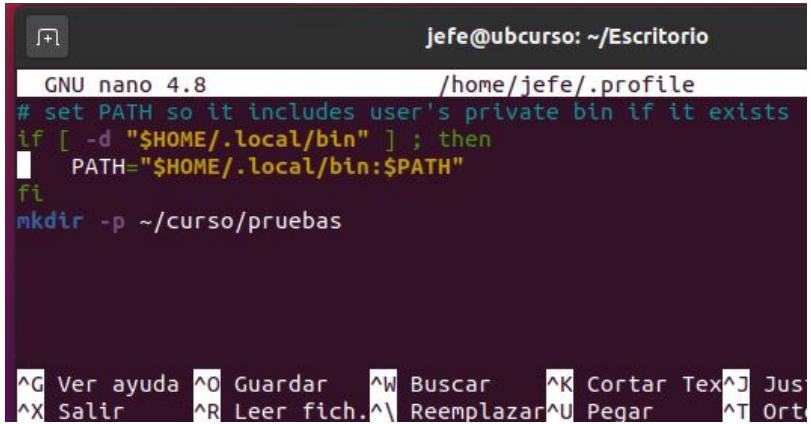
# Ejecución al inicio

Otra opción es utilizar los archivos de configuración de los usuarios (/etc/profile y /etc/bashrc por ejemplo), para ejecutar scripts al iniciar sesión.
Si es solo para un usuario concreto, se utilizarán los de su perfil:
Por ejemplo, añadir echo HOLA al final del archivo ~/.bashrc, colocará un mensaje en cada sesión bash que se abra.


# Ejecución al inicio

Para que se ejecute con el inicio de sesión, y no con el shell, se puede utilizar el archivo ~/.profile
(O el /etc/profile)

Añadiendo al final el comando mkdir, creará una carpeta al iniciar sesión este usuario.




# Ejecución al inicio

En los sistemas SysV antiguos podía hacerse:
Colocar el script en /etc/init.d/ y ejecutar update-rc
O más artesanalmente crear los enlaces simbólicos en cada run-level (inicio, apagado, reiniciado) correspondiente:
ln -s /home/jefe/curso/script.sh /etc/rc0.d/K99enlace-simbólico
Actualmente se impone Systemd

Más info:
https://www.linuxito.com/gnu-linux/nivel-alto/42-iniciar-servicios-automaticamente-en-ubuntu


# Script endemoniado

Angel Luis Calvo

ESCUELA IT

# Script endemoniado

Los scripts pueden ser lanzados con el inicio del sistema. Ya depende del propio script el comportamiento del mismo.

Los demonios son programas que quedan en segundo plano, ejecutándose según su propia programación. Podemos hacer que los scripts se comporten así.

Dos formas de plantearlo:

- Como script lanzado de inicio (rc.local, cron, etc.)
- Como demonio del sistema

# Script endemoniado

Un script lanzado con el inicio del sistema, se puede considerar un demonio mientras siga en ejecución, en espera de algún evento que le obligue a hacer algo.
Un comando útil en esa situación puede ser *trap*.
Con *trap* se pueden capturar señales de interrupción del sistema.
Para ver un listado de esas señales podemos ejecutar kill -l o trap -l.
Sintaxis:
trap *comandos señal/es* → se ejecutan esos comandos cuando se detecte una de esas señales

# Script endemoniado

La idea es que si se produce algún tipo de interrupción sobre el script, tener controlado cómo terminar el mismo (generando un log, eliminando archivos temporales, cualquier otra cosa)

Ejemplo de trap:

```
#!/bin/bash
trap 'echo "TRL-C está deshabilitado"' SIGINT
#se podía haber puesto 2 en vez del SIGINT, o abreviar con INT
until ! : ;   do
        echo Esto se para poniendo \"parar\" en el archivo control.txt
        sleep 1
        if [ "$(cat control.txt)" = parar ] ; then
                echo Por fin
                exit
        fi
done
```

# Ejercicio

Automatizar con un script el movimiento de los logs del directorio /var/log/mis_logs, al directorio /home/jefe/curso/mis_logs en el momento en que se apague o reinicie el equipo. (Para que sea como demonio colocamos el script en el cron con la cláusula @reboot)

@reboot root /home/jefe/scripts/mover_logs.sh

***mover_logs.sh***

```
#!/bin/bash
logs="/var/log/mis_logs"
destino="/home/jefe/curso/mis_logs/"
mover () {
date >> "$destino"milog
mv $logs/* $destino
trap - TERM INT #reasignamos las señales por defecto
exit
}
```



---------- Continúa en la siguiente----------


ESCUELA IT

# Ejercicio

---------- Viene de la anterior ----------

```
while : ; do
trap mover TERM INT #capturamos interrupciones típicas
done
```


ESCUELA IT

# Script endemoniado: systemd

Lo más efectivo es crear nuestros propios demonios de sistema para ejecutar los scripts.

En Systemd, la mayoría de demonios se encuentran en los enlaces simbólicos de los subdirectorios de /etc/systemd, que apuntan a /lib/systemd

También puede haberlos en los perfiles de usuario, limitados a los privilegios de cada usuario:
~/.config/systemd/user/

# Script endemoniado: systemd

Para crear uno, partimos de que tenemos un script que queremos ejecutar como demonio.

Para ello crearemos un “archivo de unidad” de tipo *.service* en /etc/systemd/system
Si se quiere se puede hacer un enlace aquí y crear el archivo en /lib/systemd/system, como en la mayoría de servicios.

Más info:
https://wiki.archlinux.org/index.php/Systemd_(Espa%C3%B1ol)#Escribir_archivos_de_unidad

# Script endemoniado: systemd

La estructura de un archivo de unidad es:

- [Unit]
  - Description
  - After: la unidad se inicia después de las unidades indicadas.
  - Before: lo opuesto al anterior.
  - Wants: intenta activar las unidades indicadas aquí.
  - Requires: configura dependencias sobre otras unidades. Si no logra activarlas, ésta no se activará tampoco.

# Script endemoniado: systemd

La estructura de un archivo de unidad es:

- [Service]
  - ExecStart: comando a ejecutar al arrancar la unidad
  - ExecStop: comando al parar la unidad
  - Type: tipo de arranque
  - RemainAfterExit: se considerará el proceso como activo después de que haya salido.

# Script endemoniado: systemd

La estructura de un archivo de unidad es:

- [Install]
  - WantedBy: Indica el *target* al que pertenece esta unidad. Con systemctl enable se crean los enlaces simbólicos necesarios dentro de target.wants/ sin tener que hacerlo a mano.

Los *target* son los entornos de trabajo de los servicios

# Script endemoniado: systemd

Después de crear el archivo hay que ejecutar:

- systemctl daemon-reload: para recargar los demonios, nos puede informar de problemas de sintaxis
- systemctl enable *demonio*.service: para que arranque con el sistema
- systemctl start *demonio*.service: para arrancarlo a mano

# Script endemoniado: systemd

Ejemplo de demonio para ejecutar en el apagado algo sencillo:

[Unit]

Description="Prueba de apagado"

Before=shutdown.target → para que se ejecute antes de esto

[Service]

Type=oneshot → solo una ejecución

RemainAfterExit=true → esta y la siguiente para que el servicio quede activo

ExecStart=/bin/true → simplemente algo válido para la sintaxis

ExecStop="ls /home/jefe/curso >> /home/jefe/listado_final.log" → al cerrar hace esto

[Install]

WantedBy=multi-user.target → entorno de usuario con todos los servicios activos

# Ejercicio

Crear un demonio en Systemd que ejecute una copia de seguridad al apagar o reiniciar el equipo. (script de copia tar_back.sh)

***apagando.service***

[Unit]
Description="Script para copias de seguridad en el apagado"
Wants=network.target → necesitamos servicios de red para el copiado
Before=unmount.target → debe actuar antes de que se desmonte la unidad de red

[Service]
Type=oneshot → para que se ejecute una vez
RemainAfterExit=true → al arranque que continúe activo
ExecStart=/bin/true → no queremos que haga nada al arrancar, solo que quede activo
#Copiaremos antes de apagar
ExecStop=/home/jefe/scripts/tar_back.sh

---------- Continúa en la siguiente----------


ESCUELA IT

# Ejercicio

---------- Viene de la anterior ----------

[Install]
WantedBy=shutdown.target → es el entorno de apagado


ESCUELA IT

# Script endemoniado: systemd

Para buscar posibles problemas podemos usar:

journalctl -u *demonio.service*
O también:
systemctl status *demonio.service*

Además de los logs de /var/log/messages y syslog

# Ejercicio

Crear un demonio en Systemd que ejecute un ping continuo para verificar el estado de un servidor.

***alerta_servidor.service***

```
[Unit]
Description="Script de ping al servidor"
After=network-online.target

[Service]
Type=simple
ExecStart=/home/jefe/scripts/ping_cont.sh
```

→ nuestro script

```
[Install]
WantedBy=multi-user.target
```

→ es el entorno de todo activado

***ping_cont.sh***

---------- Continúa en la siguiente----------

# Ejercicio

---------- Viene de la anterior ----------

```
#!/bin/bash
servidor=192.168.15.24
fecha=$(date)
while :
do
        contador=0
        sleep 10m #Lo prueba cada 10 minutos
        for num in {1..5} ; do
                ping -c 1 "$servidor" &> /dev/null
                if [ $? -eq 1 ]; then
                        sleep 2 #falló el ping
                        contador=$(($contador + 1))
                fi
```

---------- Continúa en la siguiente----------


ESCUELA IT

# Ejercicio

---------- Viene de la anterior ----------

```
        done
        if [ "$contador" -eq 5 ]; then
                echo Servidor caído "$fecha" >> /home/jefe/alerta_servidor
                mail -s "Servidor caído" usuario@gmail.com <<< "Alerta de caída"
        fi
done
```


ESCUELA IT


# Temporizadores

Angel Luis Calvo

ESCUELA IT

# Temporizadores

Para ejecutar procesos temporizados se puede utilizar *cron*. Pero los demonios de Systemd también cuentan con *timers*.

Cómo crear una temporización:
Es necesario crear un archivo .service (no necesita sección *Install*) y un archivo .timer (con el mismo nombre, por claridad)

Ubicación: la misma que cualquier demonio

# Temporizadores

Hay dos formas de temporizar:

- Clásica: de calendario, usa la instrucción *OnCalendar*
- Rítmica: se ejecuta cada cierto tiempo desde el arranque, comandos como *OnBootSec* y *OnActiveSec*

Los timers se activan como los servicios (systemctl *enable-start* etc.)

# Temporizadores

Formato OnCalendar:
*Día_de_la_semana Año-mes-día Hora:minuto:segundo*

Ejemplo de archivo *.timer*:
[Unit]
Description=Tempo una vez al día
[Timer]
OnCalendar=*-*-* 22:00 → a diario a esa hora
Persistent=yes → se ejecuta en el arranque si anteriormente no pudo (por estar apagado)
#Unit=nombre.service → solo si el servicio tiene otro nombre
[Install]
WantedBy=timers.target

# Temporizadores

Ejemplos de OnCalendar:
(Se pueden poner tantas sentencias como se quiera, los asteriscos pueden obviarse si no hay ambigüedad)
Mon-Sat *-*-* 12:30:15 → de lunes a sábado a esa hora
*-1,4-1 * → cada primer día de enero y abril
*-*-1..7 12:00 → los días del 1 al 7, a mediodía
weekly → semanalmente, equivale a → Mon *-*-* 00:00:00
*:0/10 → cada 10 minutos (a y 10, 20, 30,...)
*:30 → todos los días a y media

# Temporizadores

Cuando se usan expresiones especiales como *hourly* (*-*-* *:0:0), o *minutely* (*-*-* *:*:0) pueden coincidir con los especificados en otros *timers*, por lo que conviene usar en ese caso *RandomizedDelaySec*:
RandomizedDelaySec=5min → aproximadamente retarda x minutos la ejecución para no saturar el equipo.

Más info:
https://wiki.archlinux.org/index.php/Systemd_(Espa%C3%B1ol)/Timers_(Espa%C3%B1ol)

# Temporizadores

Ejemplo de temporización rítmica:

[Unit]
Description=”Script de prueba”

[Timer]
OnBootSec=5min → arranca este tiempo después del arranque del sistema
OnUnitActiveSec=1w → cada semana se vuelve a ejecutar

[Install]
WantedBy=timers.target

# Ejercicio

Crear una tarea que cada 20 minutos revise el contenido de la carpeta del curso, obteniendo un listado de la misma en /home/jefe/carpeta_curso.txt.
Script: listar.sh; tarea: listado_curso.service y listado_curso.timer

***listar.sh***

```
#!/bin/bash
#Simplemente, saca un listado del contenido de curso
ls -l /home/jefe/curso > /home/jefe/carpeta_curso.txt
```



***listado_curso.service***

```
[Unit]
Description="Script para listar contenido de curso"
#El momento de ejecución ya no se decide aquí, lo hace el timer

[Service]
```

---------- Continúa en la siguiente----------


ESCUELA IT

# Ejercicio

---------- Viene de la anterior ----------

```
Type=oneshot
ExecStart=/home/jefe/scripts/listar.sh
```

***listado_curso.timer***

```
[Unit]
Description="Para hacer listado de curso cada 20min"

[Timer]
OnCalendar=*:0/20

[Install]
WantedBy=timers.target
```

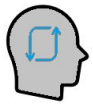

ESCUELA IT

# Temporizadores

Existe la opción de generar temporizadores transitorios, algo semejante a lo que hace el comando *at*. Se hace con el comando systemd-run.

systemd-run --on-active=*30min /home/jefe/mi_script.sh* → para ejecutar un script al cabo de un tiempo

Para consultar los temporizadores configurados:
systemctl list-timers



Recuerde:
Los temporizadores son también demonios, se pueden usar los comandos relacionados con ellos.

# Ejercicios

1.- Automatizar con un script la liberación de una unidad RAM (/tmp/ram) para que se elimine al reducirse el espacio libre en RAM hasta los 256MB. Debe ponerse a salvo su contenido en /mount/servidor/. Si se apaga o reinicia el equipo debe liberarse igualmente.

52_

2.- Crear un servicio que arranque con el sistema, de forma que cada hora revise el contenido del archivo syslog, y si encuentra el comentario “Revisar ahora”, envíe un correo al administrador indicándolo.

53_

3.- Automatizar la copia de los archivos de /mnt/servidor al arranque y cada dia, ya que son archivos importantes que se eliminan cada día, de tal forma que se almacenen en /home/jefe/curso/almacen. Las copias de más de seis meses se guardarán en /home/jefe/curso/almacen/reserva, las de más de un año se eliminarán.

54_


ESCUELA IT


1.-

***liberar_ram.sh***

```
#!/bin/bash
ram="/tmp/ram"
servidor="/mnt/servidor/"
liberar (){
mv $ram/* $servidor 2> /dev/null
mountpoint $ram > /dev/null
if [ $? -ne 1 ] ; then
        umount /tmp/ram
fi
}
while true ; do
        sleep 3
        mem_libre=$(free | grep Memoria | tr -s " " | cut -d " " -f 4)
        if [ $mem_libre -lt 262144 ] ; then
                liberar
```

---------- Continúa en la siguiente----------

ESCUELA IT


---------- Viene de la anterior ----------

```
        fi
done
```

***liberar_ram.service***

```
[Unit]
Description="Script para salvar el disco ram si escasea memoria"
Wants=network.target
After=network-online.target mount.target

[Service]
Type=simple
ExecStart=/home/jefe/scripts/liberar_ram.sh

[Install]
WantedBy=multi-user.target
```

---------- Continúa en la siguiente----------

ESCUELA IT


---------- Viene de la anterior ----------

***liberar_ram_apagando.service***

```
[Unit]
Description="Script para salvar el disco ram al apagado"
Wants=network.target
Before=unmount.target

[Service]
Type=oneshot
RemainAfterExit=true
ExecStart=/bin/true
ExecStop=/home/jefe/scripts/liberar_ram.sh

[Install]
WantedBy=shutdown.target
```

ESCUELA IT

2.-

***revisar.sh***

```
#!/bin/bash
#Comprobar en syslog una determinada cadena de caracteres.
#para enviar un correo en ese caso
mensaje="Revisar ahora"
grep "$mensaje" /var/log/syslog > miro_syslog
if [ $? -eq 0 ]
   then
        echo Se encontró el mensaje
        mail -s "$mensaje" usuario@gmail.com <<< "Se recibió mensaje en syslog"
        systemctl stop revision.timer
fi
```

***revision.service***

---------- Continúa en la siguiente----------


---------- Viene de la anterior ----------

```
[Unit]
Description="Script para revisar el registro syslog"

[Service]
Type=oneshot
ExecStart=/home/jefe/scripts/revisar.sh
```

***revision.timer***

```
[Unit]
Description="Para avisar de si pasa algo con syslog, cada hora"
[Timer]
OnCalendar=hourly
RandomizedDelaySec=6min
[Install]
WantedBy=timers.target
```

ESCUELA IT


3.-

***copi_total.sh***

```
#!/bin/bash
#Copiamos todo del servidor, porque lo eliminan cada día
#Los mantenemos en un almacén seis meses, los trasladamos a una reserva
#y al cabo de un año se borran
servidor=/mnt/servidor
almacen=/home/jefe/curso/almacen
reserva=/home/jefe/curso/almacen/reserva
#copia de archivos
cp  "$servidor"/* "$almacen"/
#comprobaciones
#usamos el comando find, la f indica fichero
#mtime, fecha de modificación en días
for archivo in $(find "$almacen" -type f -mtime +180)
do
```

---------- Continúa en la siguiente----------

ESCUELA IT


---------- Viene de la anterior ----------

```
        mv "$archivo" "$reserva"
        echo Movido archivo "$archivo" >> "$almacen"/movidos_borrados.log
done

for viejo in $(find "$reserva" -type f -mtime +360)
do
        rm "$viejo"
        echo Borrado archivo "$viejo" >> "$almacen"/movidos_borrados.log
done
```

***copion.service***

```
[Unit]
Description="Script para copias y eliminación de las antiguas"

[Service]
```

---------- Continúa en la siguiente----------


---------- Viene de la anterior ----------

```
Type=oneshot
ExecStart=/home/jefe/scripts/copi_total.sh
```

***copion.timer***

```
[Unit]
Description="Para copia de seguridad, cada día"

[Timer]
OnBootSec=15min
OnUnitActiveSec=1d

[Install]
WantedBy=timers.target
```

ESCUELA IT


BASH

THE BOURNE-AGAIN SHELL

# GRACIAS!!

Angel Luis Calvo

ESCUELA IT