Memoria Práctica 4

Administración de Servidores

Gestión de Sistemas de ficheros y dispositivos

Almudena García Jurado-Centurión

Índice

0. Introducción

En esta práctica aprenderemos a crear y gestionar diferentes sistemas de ficheros.
Los sistemas de ficheros son los encargados de indexar los ficheros dentro del disco duro, para que estos puedan ser encontrados por el sistema operativo.

El disco duro, a su vez, puede estar dividido en particiones. En el sistema MBR clásico, se permiten 4 particiones primarias y una extendida. La partición extendida es aquella que agrupa a varias particiones dentro de ella, siendo estas llamadas “particiones lógicas”

A lo largo de los ejercicios de esta práctica, iremos creando particiones, y sistemas de ficheros dentro de ellas; desde la propia línea de comandos.

También realizaremos algunas personalizaciones sobre el sistema de ficheros, como eliminar el espacio reservado para root, o asignar etiquetas a las particiones.

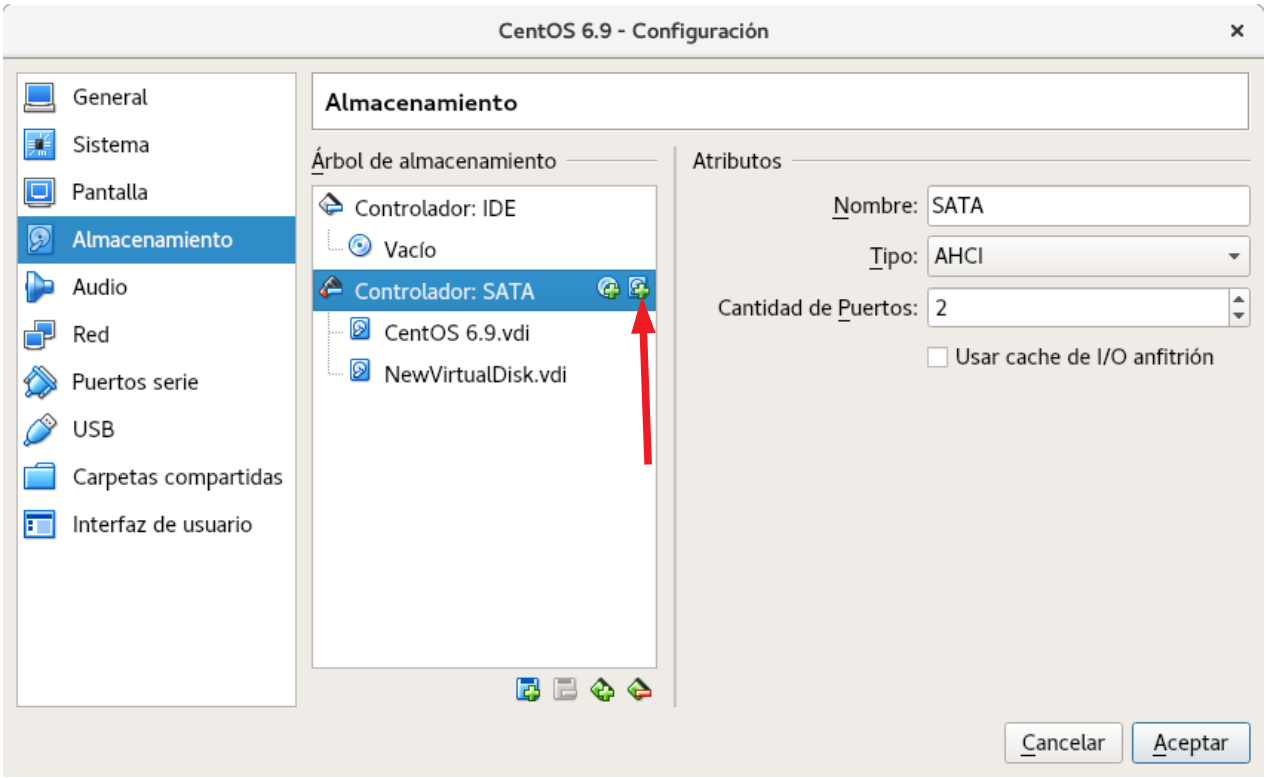
Y, finalmente, en la última parte de la práctica, aprenderemos a crear un sistema de ficheros cifrado sobre una partición; lo cual nos permitirá añadir un extra de seguridad a nuestros archivos.

1. Crear un nuevo disco duro

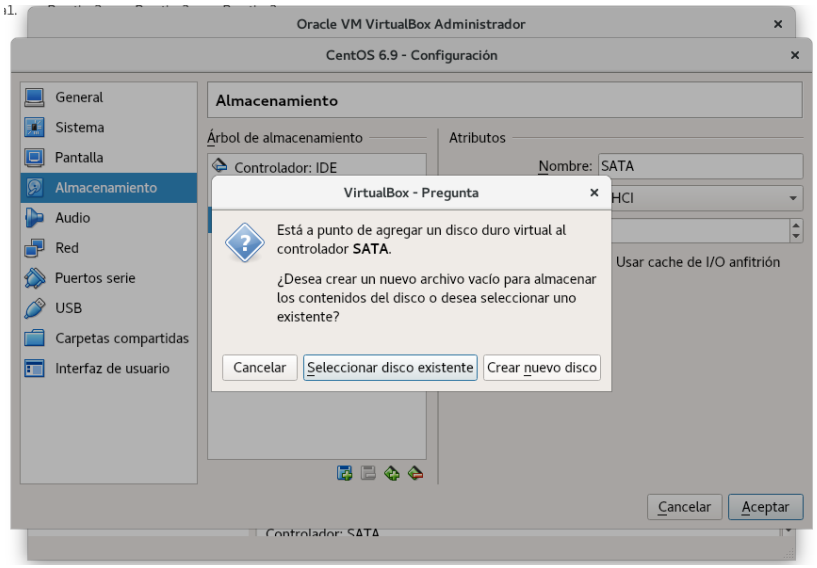
En este apartado, vamos a crear un nuevo disco duro en nuestra máquina virtual, de 384MB.

Para ello abrimos VirtualBox, y nos vamos a las propiedades de nuestra máquina virtual, a la sección de Almacenamiento.

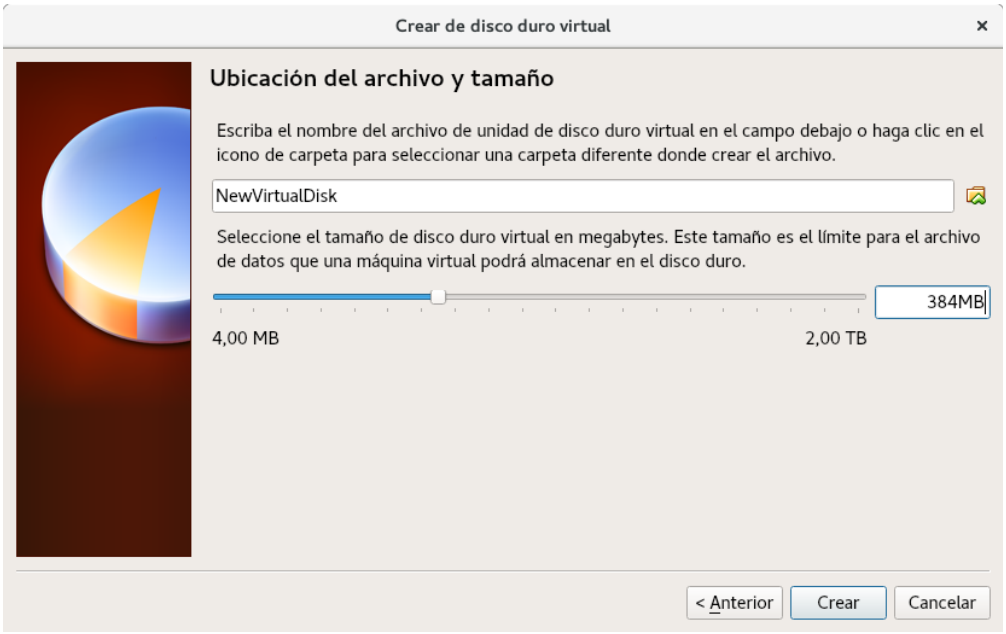
Veremos algo similar a esto:



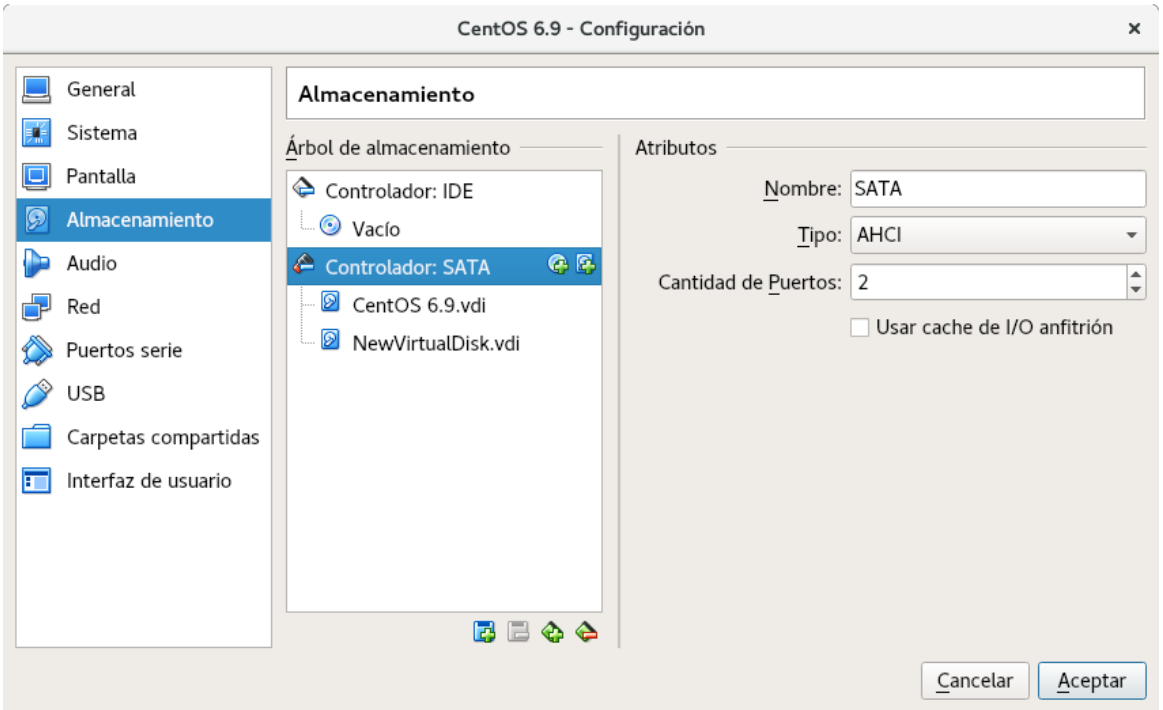
Pulsamos en el segundo botón para crear un nuevo disco
Nos aparecerá una ventana como esta. Pulsamos en “crear nuevo disco”



Este nos llevará al asistente de creación de discos, ya visto en las prácticas anteriores. Indicamos un tamaño de 384 MB, y pulsamos en crear.



Una vez creado, ya veremos la nueva unidad asignada al controlador. En este caso, tendrá de nombre NewVirtualDisk.



2. Creando particiones en el nuevo disco duro

Una vez creado el nuevo disco duro, vamos a dividirlo en 3 particiones de 128MB cada una.

Antes de empezar, ejecutamos *lsblk* para comprobar que el sistema ha detectado el disco correctamente, y ver la unidad que le ha asignado al dispositivo.

```
[root@localhost ~]# lsblk
NAME                          MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sr0                            11:0    1 1024M  0 rom
sda                             8:0    0   20G  0 disk
├─sda1                          8:1    0  500M  0 part /boot
└─sda2                          8:2    0 19,5G  0 part
  ├─VolGroup-lv_root (dm-0)   253:0    0 17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1)   253:1    0    2G  0 lvm  [SWAP]
sdb                             8:16   0  384M  0 disk
[root@localhost ~]#
```

Vemos que el sistema detecta el nuevo disco duro, y su unidad es *sdb.*

Una vez con esta información, pasamos a crear las particiones. Para ello, usaremos la herramienta *fdisk,* que nos permite gestionar el espacio de nuestro disco duro desde línea de comandos.

Para iniciarlo, ejecutamos *fdisk /dev/[unidad].* Como en este caso la unidad es sdb, usamos *fdisk /dev/sdb*

```
[root@localhost ~]# fdisk /dev/sdb
El dispositivo no contiene una tabla de particiones DOS válida ni una etiqueta d
e disco Sun o SGI o OSF
Building a new DOS disklabel with disk identifier 0x3173ac49.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Atención: el indicador 0x0000 inválido de la tabla de particiones 4 se corregirá
 mediante w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Orden (m para obtener ayuda):
```

Tras iniciarlo, fdisk nos avisa de que el disco duro no tiene una tabla de particiones, así que tenemos que crear una nueva.

Escribimos “m” para mostrar las órdenes disponibles, y encontrar la que necesitamos.

```
   n   Añade una nueva partición
   o   Crea una nueva tabla de particiones DOS vacía
   p   Imprime la tabla de particiones
   q   Sale sin guardar los cambios
   s   Crea una nueva etiqueta de disco Sun
   t   Cambia el identificador de sistema de una partición
   u   Cambia las unidades de visualización/entrada
   v   Verifica la tabla de particiones
   w   Escribe la tabla en el disco y sale
   x   Funciones adicionales (sólo para usuarios avanzados)
```

Vemos que la orden para crear una nueva tabla de particiones de tipo DOS es “o”, así que lo ejecutamos.

```
Orden (m para obtener ayuda): o
Building a new DOS disklabel with disk identifier 0xa86a1b94.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Atención: el indicador 0x0000 inválido de la tabla de particiones 4 se corregirá
 mediante w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Orden (m para obtener ayuda):
```

Ya con la tabla de particiones creada, pasamos a crear nuestras particiones. Para ello, usaremos la opción “n”.

Nos preguntará si queremos crear una partición primaria o extendida. En este caso, como solo vamos a crear 3 particiones, pondremos las tres como primarias.
En las siguientes consultas, nos pedirá el número de partición, y el primer cilindro, que lo dejaremos a sus valores por defecto.
Finalmente, nos pedirá el último cilindro, que lo podemos definir mediante un tamaño expresado en KiB (K), MiB (M) y GiB (G). Como queremos que la partición sea de 128MB, pondremos +128M

```
Orden (m para obtener ayuda): n
Acción de la orden
e   Partición extendida
   p   Partición primaria (1-4)

Acción de la orden
e   Partición extendida
   p   Partición primaria (1-4)
p
Número de partición (1-4): 1
Primer cilindro (1-48, default 1):
Using default value 1
Last cilindro, +cilindros or +size{K,M,G} (1-48, default 48): +128M
```

Seguimos el proceso con el resto de particiones. En la última, por un error de cálculo en el tamaño de las particiones anteriores, nos falta espacio para los 128MB, así que dejamos todo el espacio restante del disco para ella.

```
Orden (m para obtener ayuda): n
Acción de la orden
e   Partición extendida
   p   Partición primaria (1-4)
p
Número de partición (1-4): 2
Primer cilindro (18-48, default 18):
Using default value 18
Last cilindro, +cilindros or +size{K,M,G} (18-48, default 48): +128M

Orden (m para obtener ayuda): n
Acción de la orden
e   Partición extendida
   p   Partición primaria (1-4)
p
Número de partición (1-4): 3
Primer cilindro (35-48, default 35):
Using default value 35
Last cilindro, +cilindros or +size{K,M,G} (35-48, default 48): +128M
El valor está fuera del rango.
Last cilindro, +cilindros or +size{K,M,G} (35-48, default 48):
Using default value 48
```

Finalmente, usamos la orden “w” para escribir los cambios en el disco.

```
Orden (m para obtener ayuda): w
¡Se ha modificado la tabla de particiones!

Llamando a ioctl() para volver a leer la tabla de particiones.
Se están sincronizando los discos.
[root@localhost ~]#
```

Hecho esto, volvemos a ejecutar *lsblk* para mostrar las nuevas particiones y las subunidades asignadas.

```
[root@localhost ~]# lsblk
NAME                    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0                      11:0    1  1024M  0 rom
sda                       8:0    0    20G  0 disk
├─sda1                    8:1    0   500M  0 part /boot
└─sda2                    8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0  17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1) 253:1    0     2G  0 lvm  [SWAP]
sdb                       8:16   0   384M  0 disk
├─sdb1                    8:17   0 133,3M  0 part
├─sdb2                    8:18   0 133,4M  0 part
└─sdb3                    8:19   0 109,8M  0 part
[root@localhost ~]#
```

lsblk usa MB en vez de MiB, así que vemos que las 2 primeras particiones han quedado con un espacio de 133 MB, y la última, mas pequeña, con 109 MB

3. Formateando y montando las particiones

En este paso, vamos a crear nuevos sistemas de ficheros dentro de las particiones.

- **Primera partición**
  - **Creando la partición**

La primera partición, */dev/sdb1*, tendrá un sistema de ficheros ext2, y se montará en */mnt.*

Para darle formato a la partición, usaremos el comando *mkfs*, que nos permite crear un nuevo sistema de ficheros dentro de una partición.

La sintaxis de *mkfs* es:

mkfs -t [tipo] [unidad]

donde *tipo* es el tipo de sistema de ficheros a crear (ext2, ext3… etc), y unidad, la partición a formatear (/dev/sdb1, /dev/sdb2… etc).

En nuestro caso, queremos crear un sistema de ficheros ext2 en la primera partición del dispositivo, correspondiente a */dev/sdb1*, así que el comando será:

mkfs -t /dev/sdb1

```
[root@localhost ~]# mkfs -t ext2 /dev/sdb1
mke2fs 1.41.12 (17-May-2010)
Etiqueta del sistema de ficheros=
Tipo de SO: Linux
Tamaño del bloque=1024 (bitácora=0)
Tamaño del fragmento=1024 (bitácora=0)
Stride=0 blocks, Stripe width=0 blocks
34136 nodos-i, 136520 bloques
6826 bloques (5.00%) reservados para el superusuario
Primer bloque de datos=1
Número máximo de bloques del sistema de ficheros=67371008
17 bloque de grupos
8192 bloques por grupo, 8192 fragmentos por grupo
2008 nodos-i por grupo
Respaldo del superbloque guardado en los bloques:
        8193, 24577, 40961, 57345, 73729

Escribiendo las tablas de nodos-i: hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hec
ho

Este sistema de ficheros se revisará automáticamente cada 29 montajes o
180 días, lo que suceda primero.  Utilice tune2fs -c o -i para cambiarlo.
[root@localhost ~]#
```

Tras ejecutar el comando, *mkfs* nos muestra un resumen con los datos del sistema de ficheros recién creado.

- **Montando la partición**

Una vez creada la partición, la montamos en el directorio */mnt.* Para ello, usaremos el comando *mount,* que nos permite montar una partición en un directorio de nuestro árbol de ficheros.

La sintaxis de *mount* es:

mount [unidad] [punto de montaje]

donde *unidad* se corresponde a nuestra partición, y *punto de montaje* al directorio donde lo queremos montar.

En nuestro caso, será:

mount /dev/sdb1 /mnt

```
[root@localhost ~]# mount /dev/sdb1 /mnt
```

- **Comprobando el espacio ocupado con *du* y *df***

Una vez montado, usaremos los comandos *du* y *df* para obtener el espacio libre y ocupado de la partición.

El comando *du* nos indica el espacio ocupado por un fichero o, en caso de un directorio, por los ficheros contenidos en este.

El comando *df* nos da información sobre el espacio libre y ocupado de un dispositivo.

```
[root@localhost ~]# du /mnt -h
2,0K    /mnt
[root@localhost ~]# df /dev/sdb1 -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdb1       130M  5,6M  117M   5% /mnt
[root@localhost ~]#
```

Al comparar la salida de ambos ficheros, vemos que *du* nos indica un espacio ocupado de 2KB, mientras que *df* nos da 5,6MB.

Esto puede ser debido a que el sistema de ficheros reserva parte del espacio para sus propios ficheros y estructuras.

Hecho esto, pasamos a desmontar la partición. Para ello usamos el comando *umount*, indicándole el punto de montaje o partición que queremos desmontar

```
[root@localhost ~]# umount /mnt
[root@localhost ~]#
```

- **Segunda partición**
    - **Creando una partición sin espacio reservado para root**

En la segunda partición crearemos otro sistema de ficheros ext2, esta vez sin espacio reservado para root

Para crear la partición, volveremos a usar el comando *mkfs,* esta vez con la unidad *sdb2*

Para indicar el porcentaje de espacio reservado, usaremos la opción *-mpercent*

```
[root@localhost ~]# mkfs -t ext2 /dev/sdb3 -mpercent 0
mkfs.ext2: el porcentaje de bloques reservados es inválido - percent
[root@localhost ~]#
```

En este caso, nos dice que el porcentaje de 0% es invalido, así que lo forzamos ejecutando directamente la herramienta a nivel bajo, con el comando *mkfs.ext2*

```
[root@localhost ~]# mkfs.ext2 /dev/sdb2 -m 0 -F
mke2fs 1.41.12 (17-May-2010)
Etiqueta del sistema de ficheros=
Tipo de SO: Linux
Tamaño del bloque=1024 (bitácora=0)
Tamaño del fragmento=1024 (bitácora=0)
Stride=0 blocks, Stripe width=0 blocks
34272 nodos-i, 136552 bloques
0 bloques (0.00%) reservados para el superusuario
Primer bloque de datos=1
Número máximo de bloques del sistema de ficheros=67371008
17 bloque de grupos
8192 bloques por grupo, 8192 fragmentos por grupo
2016 nodos-i por grupo
Respaldo del superbloque guardado en los bloques:
        8193, 24577, 40961, 57345, 73729

Escribiendo las tablas de nodos-i: hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho

Este sistema de ficheros se revisará automáticamente cada 21 montajes o
180 días, lo que suceda primero.  Utilice tune2fs -c o -i para cambiarlo.
```

El resumen nos indica que hay 0 bloques reservados para el superusuario.

- **Comprobando el espacio ocupado con *du* y *df***

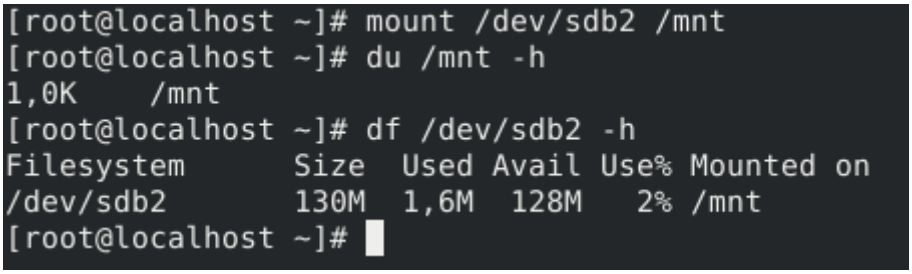
Una vez hecho esto, ejecutamos los comandos *du* y *df* para comparar los resultados.

Empezamos ejecutando *df* con la partición desmontada.

```
[root@localhost ~]# df /dev/sdb2 -h
Filesystem      Size  Used Avail Use% Mounted on
-               1,5G  216K  1,5G   1% /dev
[root@localhost ~]#
```

Vemos que, aunque indicamos que no reservara espacio para root, sigue habiendo un 1% en uso. Eso es debido a que el sistema de ficheros necesita un mínimo espacio para guardar sus datos y sus tablas de inodes.

Ahora pasamos a ejecutar el comando *du.* Para ello, montamos la partición en */mnt* y ejecutamos el comando. También volvemos a ejecutar *df* para comparar los resultados



Volvemos a ver una gran diferencia entre el espacio usado indicado por ambos comandos. Eso es debido a que *df* cuenta entre su espacio usado, el ocupado por el propio sistema de ficheros, mientras que *du* solo cuenta con el espacio ocupado por los ficheros, en este caso, el

3.1. Anexo: Cambiar el espacio reservado para root con tune2fs.

Si quisiéramos cambiar este parámetro posteriormente a la creación del sistema de ficheros, podríamos usar *tune2fs*

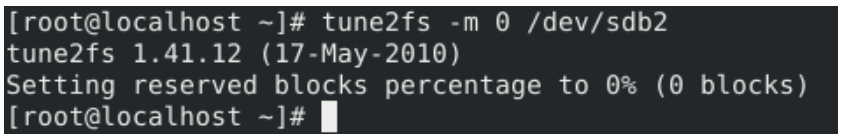
tune2fs es un comando que nos permite cambiar los parámetros de nuestro sistema de ficheros.

Su sintaxis básica es:

tune2fs [opciones] [argumentos] [unidad]

Para cambiar el porcentaje de espacio reservado se usa la opción *-m* seguido del porcentaje.

Tras ejecutar el comando, vemos que el porcentaje de bloques reservados ha pasado a ser 0%, con lo cual ya no debería haber espacio reservado para root.



Volvemos a ejecutar *df* sobre esta partición.

```
[root@localhost ~]# df /dev/sdb2 -h
Filesystem      Size  Used Avail Use% Mounted on
-               1,5G  216K  1,5G   1% /dev
[root@localhost ~]#
```

Vemos que, aunque hemos indicado que no use espacio reservado para root, el sistema de ficheros sigue teniendo el mismo porcentaje ocupado que al crear el sistema de ficheros.

Esto se debe a que el cambio no ha afectado al espacio que ya tenía ocupado el sistema de ficheros.

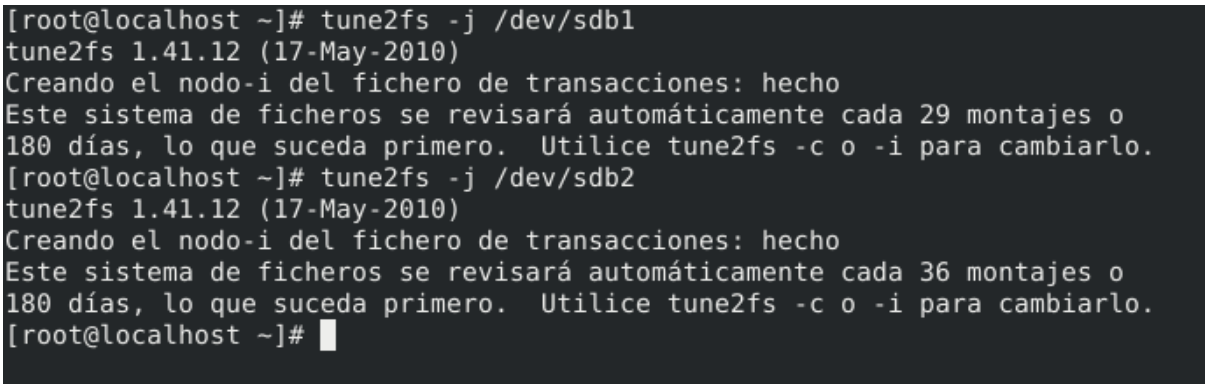
4. Convirtiendo las particiones a ext3 y estableciendo puntos de montaje

En este paso, convertiremos las dos primeras particiones, formateadas como ext2, a ext3. Posteriormente, montaremos las particiones en sus ubicaciones finales, y crearemos las configuraciones necesarias para que se monten automáticamente al inicio del sistema.

4.1. Convirtiendo las particiones a ext3

Para convertir una partición ext2 a ext3, debemos añadirle un journalling. Para esto, usaremos el comando *tune2fs* con la opción *-j*

Ejecutamos la orden con las dos particiones, *sdb1* y *sdb2*



Vemos como el fichero de journal se ha añadido correctamente en ambas particiones

4.2. Montando las particiones

Hecho esto, pasamos a montar las particiones.

La primera partición, *sdb1,* la montaremos en */tmp;* y la segunda partición, *sdb2,* en */home.*

```
[root@localhost ~]# mount /dev/sdb1 /tmp
[root@localhost ~]# mount /dev/sdb2 /home
[root@localhost ~]#
```

Ejecutamos *lsblk* para comprobar que ambas particiones están montadas en sus ubicaciones correctas

```
[root@localhost ~]# lsblk
NAME                          MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0                            11:0    1  1024M  0 rom
sda                             8:0    0    20G  0 disk
├─sda1                          8:1    0   500M  0 part /boot
└─sda2                          8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0)   253:0    0  17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1)   253:1    0     2G  0 lvm  [SWAP]
sdb                             8:16   0   384M  0 disk
├─sdb1                          8:17   0 133,3M  0 part /tmp
├─sdb2                          8:18   0 133,4M  0 part /home
└─sdb3                          8:19   0 109,8M  0 part
[root@localhost ~]#
```

4.3. Añadiendo etiquetas a las particiones

Ahora añadiremos etiquetas a ambas particiones. La primera partición tendrá la etiqueta TEMPORAL, y la segunda USUARIOS.

Para ello, usaremos el argumento -L de *tune2fs.*

Desmontamos previamente las particiones para poder aplicar dicha opción.

```
[root@localhost ~]# tune2fs -L TEMPORAL /dev/sdb1
tune2fs 1.41.12 (17-May-2010)
[root@localhost ~]# umount /dev/sdb2
[root@localhost ~]# tune2fs -L USUARIOS /dev/sdb2
tune2fs 1.41.12 (17-May-2010)
[root@localhost ~]#
```

4.4. Creando la swap en la tercera partición

Finalmente, preparamos la tercera partición para swap.

Para ello, usamos la orden *mkswap* seguida de la unidad.

```
[root@localhost ~]# mkswap /dev/sdb3
Setting up swapspace version 1, size = 112448 KiB
no label, UUID=00d61fd1-2713-4eaa-9d76-3cb18ff3f710
[root@localhost ~]#
```

Para usar esa partición como swap, usamos el comando *swapon*

```
[root@localhost ~]# swapon /dev/sdb3
[root@localhost ~]#
```

4.5. Añadiendo los cambios de forma persistente

Con estos comandos tenemos las particiones montadas en sus respectivos puntos de montaje, pero estos cambios no son persistentes, y desaparecerán al reiniciar el sistema.

Para que los cambios sean persistentes, y las particiones se monten automáticamente al iniciar el sistema, debemos editar el fichero *fstab*, ubicado en el directorio */etc.*

El fichero *fstab* esta compuesto por varias líneas, cada cual correspondiente a un punto de montaje. En cada línea se indica la partición a montar, el directorio que se usará como punto de montaje, el sistema de ficheros de la partición, y las opciones de montaje.

La partición se puede indicar mediante su unidad , mediante su etiqueta, o mediante su UUID.

La unidad es generada durante la primera conexión del dispositivo. El número de subunidad estará definido por el orden en que el sistema detectó el dispositivo.

El UUID es un identificador único que se genera durante la creación del sistema de ficheros.

La etiqueta es un nombre asignado a la partición por el usuario.

A la hora de indicar la partición en el *fstab*, podemos optar por cualquiera de las tres formas, cada cual de ellas con sus ventajas e inconvenientes.

- El usar la unidad tiene como ventaja la simplicidad a la hora de escribirlo, pero su inconveniente es que es muy dependiente del sistema, y el orden en que este detecte las particiones. En caso de que el sistema detecte las particiones en otro orden, este método ya no funcionará.

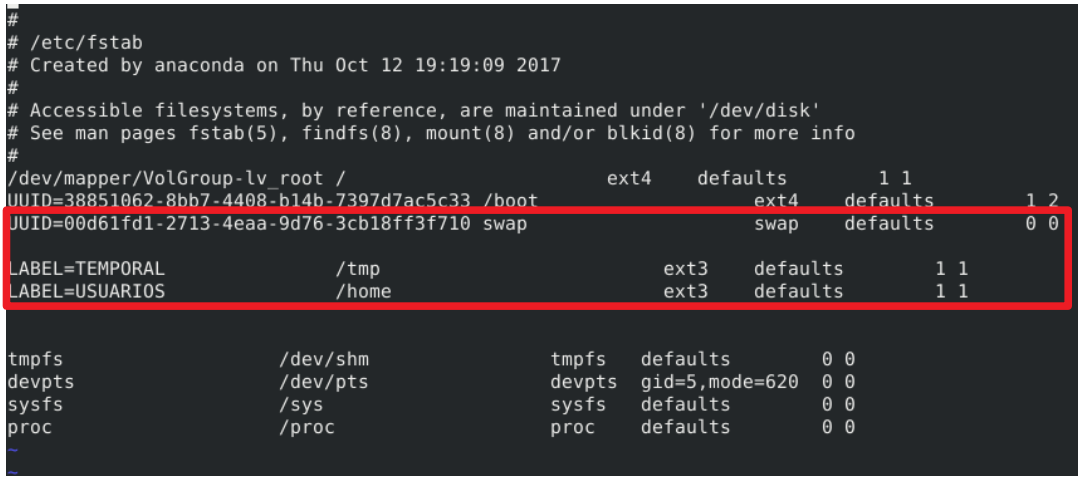
- El UUID tiene como ventaja el que la partición puede ser identificada en caso de instalar otro sistema o de mover el disco duro a otro equipo. Como inconveniente, dado que el UUID es generado por el sistema de ficheros, este cambiará cuando formateemos la partición.

- La etiqueta tiene como ventaja que es independiente a los dos casos anteriores.

En este caso, aprovecharemos las etiquetas que hemos creado como identificadores de la partición.

Para la swap, usaremos el UUID obtenido con el comando *mkswap*.

El fichero quedará así:



4.6. Comprobando los cambios

Empezamos comprobando la swap. Para ello, deshabilitamos la swap activada en *sdb3,* y veremos los cambios con *lsblk.*

Para deshabilitar una partición como swap, usaremos el comando *swapoff.*

```
[root@localhost ~]# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0    20G  0 disk
├─sda1                      8:1    0   500M  0 part /boot
└─sda2                      8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0  17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1) 253:1    0     2G  0 lvm
sdb                         8:16   0   384M  0 disk
├─sdb1                      8:17   0 133,3M  0 part /tmp
├─sdb2                      8:18   0 133,4M  0 part /home
└─sdb3                      8:19   0 109,8M  0 part [SWAP]
sr0                        11:0    1  1024M  0 rom
[root@localhost ~]# swapoff /dev/sdb3
[root@localhost ~]# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda                         8:0    0    20G  0 disk
├─sda1                      8:1    0   500M  0 part /boot
└─sda2                      8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0  17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1) 253:1    0     2G  0 lvm
sdb                         8:16   0   384M  0 disk
├─sdb1                      8:17   0 133,3M  0 part /tmp
├─sdb2                      8:18   0 133,4M  0 part /home
└─sdb3                      8:19   0 109,8M  0 part
sr0                        11:0    1  1024M  0 rom
[root@localhost ~]#
```

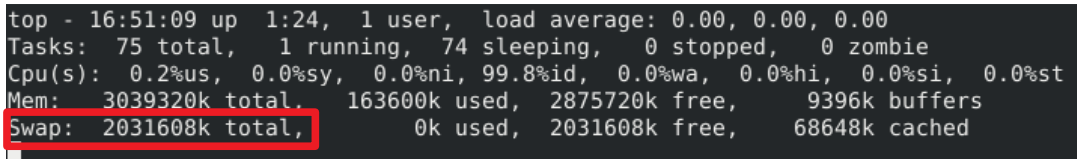
Vemos que, tras deshabilitar la swap, *lsblk* ya no muestra la etiqueta *[SWAP]* junto a la partición.

También vemos que el sistema ha detectado y montado correctamente todas las demás particiones indicadas en el *fstab.*

Con esto, ya vemos que el sistema es capaz de detectar la swap y el resto de particiones. Pero, para comprobarlo definitivamente, queremos ver el cambio de tamaño producido en la swap antes y después de activar la partición asignada a la misma.

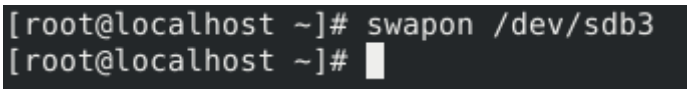
Para ello, usaremos el comando *top.*

Ejecutamos el comando con la swap de la partición *sdb3* desactivada, y miramos el campo *swap* dentro de la salida de *top.*

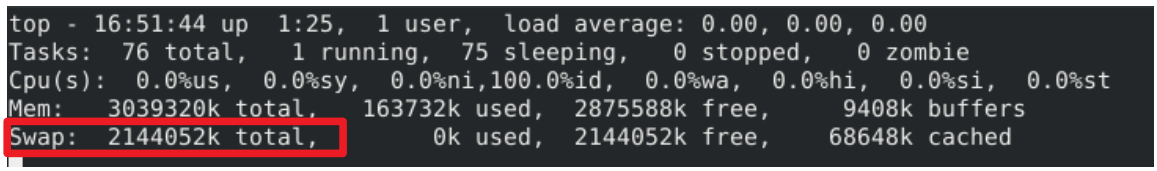


Vemos que indica 2031608 KB de swap.

Reactivamos la swap mediante *swapon*



Una vez activada la swap, ejecutamos de nuevo el comando *top* para ver los cambios.



Vemos que la swap se ha incrementado de 2031608 KB a 2144052 KB

5. Comprobando los sistemas de ficheros

Tras finalizar la creación y puesta en marcha de todas las nuevas particiones, queremos forzar la comprobación de los sistemas de ficheros de los mismos durante el arranque.

Para ello, podemos usar el comando *shutdown -rF now,* que reiniciará el sistema, y ejecutará una comprobación de sus particiones en el momento del arranque.

Ejecutamos el comando, que nos reiniciará la máquina.

```
[root@localhost ~]# shutdown -rF now_
```

Al reiniciar, comprobamos que, efectivamente, los sistemas de ficheros son chequeados durante el arranque

```
Iniciando udev:                                            [  OK  ]
Configuración del nombre de la máquina localhost.localdomai[  OK  ]
Configurando gestor de volúmenes lógicos:   2 logical volume(s) in volume group
"VolGroup" now active
                                                           [  OK  ]
Verificando sistema de archivos
/dev/mapper/VolGroup-lv_root: 89500/1152816 ficheros (0.1% no contiguos), 200159
0/4605952 bloques
TEMPORAL:
No se encontró /lost+found.  CREADO.
TEMPORAL: 12/34136 ficheros (0.0% no contiguos), 9969/136520 bloques
USUARIOS:
No se encontró /lost+found.  CREADO.
USUARIOS: 11/34272 ficheros (0.0% no contiguos), 9983/136552 bloques
/dev/sda1: 55/128016 ficheros (3.6% no contiguos), 126461/512000 bloques
                                                           [PASADO]
Remontando sistema de archivos raíz en modo de lectura y es[  OK  ]
Montando sistema de archivos local:                        [  OK  ]
Activando espacio swap de /etc/fstab:                      [  OK  ]
Entrando en el inicio no interactivo
Calling the system activity data collector (sadc)...
Starting monitoring for VG VolGroup:   2 logical volume(s) in volume group "VolG
roup" monitored
                                                           [  OK  ]
_
```

En este caso, vemos que los test han sido pasados correctamente.

5.1. Anexo: Forzando la comprobación para el siguiente inicio

Otra forma de forzar la comprobación de los sistemas de ficheros es creando el fichero *forcefsck*

Este fichero obliga al sistema a chequear los sistemas de ficheros en el próximo arranque el sistema.

A diferencia de como ocurría con *shutdown -rF,* no tenemos que programar un apagado para realizar el chequeo sino que el sistema, en el siguiente arranque, lanza la comprobación.

Para crear este fichero, usamos el comando:

touch /forcefsck

Tras esto, podemos apagar o reiniciar el sistema de la forma y en el momento que queramos, pues la revisión queda programada para el próximo inicio.

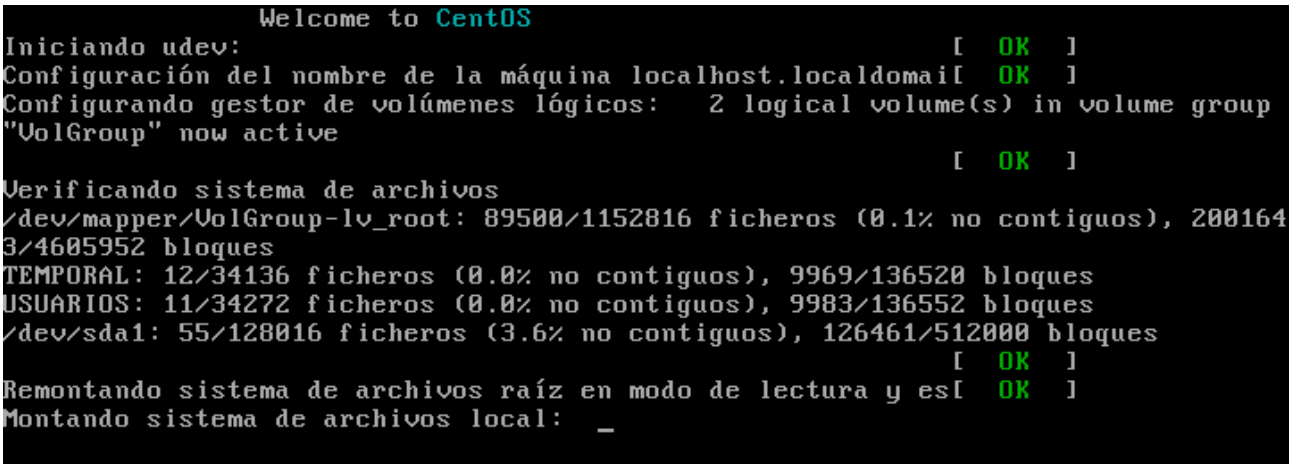
Lo comprobamos ejecutando el comando, apagando con halt, e iniciando de nuevo la máquina.

```
[root@localhost ~]# touch /forcefsck
[root@localhost ~]# halt_
```

Durante el apagado, el sistema nos avisa de que el sistema de ficheros será comprobado en el siguiente arranque

```
Apagando postfix:                                          [  OK  ]
Parando crond:                                             [  OK  ]
Stopping block device availability: Deactivating block devices:
  [SKIP]: unmount of VolGroup-lv_root (dm-0) mounted on /
                                                           [  OK  ]
Parando auditd:                                            [  OK  ]
Desactivando el generador de registros del sistema:        [  OK  ]
Interrupción de la interfaz eth0:                          [  OK  ]
Interrupción de la interfaz de loopback:                   [  OK  ]
ip6tables: Poniendo las cadenas de la política ACCEPT: filt[  OK  ]
ip6tables: Guardando las reglas del cortafuegos:           [  OK  ]
ip6tables: Descargando los módulos:                        [  OK  ]
iptables: Poniendo las cadenas de la política ACCEPT: filte[  OK  ]
iptables: Guardando las reglas del cortafuegos:            [  OK  ]
iptables: Descargando módulos:                             [  OK  ]
Sending all processes the TERM signal...                   [  OK  ]
Sending all processes the KILL signal...                   [  OK  ]
Saving random seed:                                        [  OK  ]
Syncing hardware clock to system time                      [  OK  ]
Turning off swap:                                          [  OK  ]
Unmounting file systems:                                   [  OK  ]
init: Re-executing /sbin/init
Halting system...
On the next boot fsck will be forced.
```

Tras encender la máquina, vemos que se ha iniciado la comprobación de los sistemas de fichero, y esta ha sido correcta.



6. Cifrando particiones

En este paso, vamos a crear una nueva partición, la cual montaremos en el directorio */privado* y cifraremos sus contenidos.

Para realizar el cifrado, usaremos la herramienta *dm-crypt*, que nos permite cifrar los contenidos de una partición.

- ### Añadiendo un nuevo disco

  Dado que no nos queda espacio en los discos, vamos a crear un nuevo disco duro en la máquina virtual, de 8GB de capacidad

Tras iniciar la máquina, vemos que el sistema ha detectado el nuevo disco correctamente, con el identificador *sdc*

```
[almu@debian ~]$ ssh root@localhost.localdomain -p 3500
root@localhost.localdomain's password:
Last login: Mon Dec  4 16:03:45 2017
[root@localhost ~]# lsblk
NAME                      MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0                        11:0    1  1024M  0 rom
sda                         8:0    0    20G  0 disk
├─sda1                      8:1    0   500M  0 part /boot
└─sda2                      8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0  17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1) 253:1    0     2G  0 lvm
sdb                         8:16   0   384M  0 disk
├─sdb1                      8:17   0 133,3M  0 part /tmp
├─sdb2                      8:18   0 133,4M  0 part /home
└─sdb3                      8:19   0 109,8M  0 part [SWAP]
sdc                         8:32   0     8G  0 disk
[root@localhost ~]#
```

- **Creando una partición en el disco**

Una vez creado el disco, creamos una nueva partición dentro del mismo, que ocupe todo el tamaño del disco.

Para ello, usaremos *fdisk*

```
[root@localhost ~]# fdisk /dev/sdc
El dispositivo no contiene una tabla de particiones DOS válida ni una etiqueta de disco Sun o SGI o OSF
Building a new DOS disklabel with disk identifier 0xc17e2236.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Atención: el indicador 0x0000 inválido de la tabla de particiones 4 se corregirá mediante w(rite)

WARNING: DOS-compatible mode is deprecated. It's strongly recommended to
         switch off the mode (command 'c') and change display units to
         sectors (command 'u').

Orden (m para obtener ayuda): n
Acción de la orden
e   Partición extendida
   p   Partición primaria (1-4)
p
Número de partición (1-4): 1
Primer cilindro (1-1044, default 1): 1
Last cilindro, +cilindros or +size{K,M,G} (1-1044, default 1044):
Using default value 1044

Orden (m para obtener ayuda): w
¡Se ha modificado la tabla de particiones!

Llamando a ioctl() para volver a leer la tabla de particiones.
Se están sincronizando los discos.
[root@localhost ~]#
```

Comprobamos que la partición se ha creado correctamente, usando *lsblk*

```
[root@localhost ~]# lsblk
NAME                        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0                          11:0    1  1024M  0 rom
sda                           8:0    0    20G  0 disk
├─sda1                        8:1    0   500M  0 part /boot
└─sda2                        8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0  17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1) 253:1    0     2G  0 lvm
sdb                           8:16   0   384M  0 disk
├─sdb1                        8:17   0 133,3M  0 part /tmp
├─sdb2                        8:18   0 133,4M  0 part /home
└─sdb3                        8:19   0 109,8M  0 part [SWAP]
sdc                           8:32   0     8G  0 disk
└─sdc1                        8:33   0     8G  0 part
[root@localhost ~]#
```

- **Preparando el volumen cifrado**

Creamos un nuevo volumen con formato LUKS, usando la herramienta *cryptsetup* y la opción *luksformat,* sobre la partición *sdc1* recién creada.

```
[root@localhost ~]# cryptsetup -y luksFormat /dev/sdc1

WARNING!
========
This will overwrite data on /dev/sdc1 irrevocably.

Are you sure? (Type uppercase yes): YES
Enter LUKS passphrase:
Verify passphrase:
[root@localhost ~]#
```

Hecho esto, abrimos el nuevo volumen cifrado con *cryptsetup luksopen,* poniéndole por nombre *crypt.* Nos pedirá la clave del mismo, así que la introducimos.

```
[root@localhost ~]# cryptsetup luksOpen /dev/sdc1 crypt
Enter passphrase for /dev/sdc1:
[root@localhost ~]# lsblk
NAME                        MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sr0                          11:0    1  1024M  0 rom
sda                           8:0    0    20G  0 disk
├─sda1                        8:1    0   500M  0 part  /boot
└─sda2                        8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0  17,6G  0 lvm   /
  └─VolGroup-lv_swap (dm-1) 253:1    0     2G  0 lvm
sdb                           8:16   0   384M  0 disk
├─sdb1                        8:17   0 133,3M  0 part  /tmp
├─sdb2                        8:18   0 133,4M  0 part  /home
└─sdb3                        8:19   0 109,8M  0 part  [SWAP]
sdc                           8:32   0     8G  0 disk
└─sdc1                        8:33   0     8G  0 part
  └─crypt (dm-2)            253:2    0     8G  0 crypt
[root@localhost ~]#
```

Tras abrirlo, vemos que *lsblk* ya nos detecta el nuevo volumen cifrado, conectado a la partición *sdc1*

El nuevo volumen estará ubicado en */dev/mapper/crypt*

- **Dando formato al volumen**

Una vez creado el volumen, creamos un nuevo sistema de ficheros ext4 sobre él.

```
[root@localhost ~]# mkfs.ext4 -j /dev/mapper/crypt
mke2fs 1.41.12 (17-May-2010)
Etiqueta del sistema de ficheros=
Tipo de SO: Linux
Tamaño del bloque=4096 (bitácora=2)
Tamaño del fragmento=4096 (bitácora=2)
Stride=0 blocks, Stripe width=0 blocks
524288 nodos-i, 2095962 bloques
104798 bloques (5.00%) reservados para el superusuario
Primer bloque de datos=0
Número máximo de bloques del sistema de ficheros=2147483648
64 bloque de grupos
32768 bloques por grupo, 32768 fragmentos por grupo
8192 nodos-i por grupo
Respaldo del superbloque guardado en los bloques:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Escribiendo las tablas de nodos-i: hecho
Creating journal (32768 blocks): hecho
Escribiendo superbloques y la información contable del sistema de ficheros: hecho

Este sistema de ficheros se revisará automáticamente cada 38 montajes o
180 días, lo que suceda primero.  Utilice tune2fs -c o -i para cambiarlo.
```

- **Montando el volumen cifrado**

Con el volumen creado y formateado, usamos el comando *mount* para realizar el montaje en el directorio /privado

```
[root@localhost ~]# mount /dev/mapper/crypt /privado/
[root@localhost ~]# ls /privado
lost+found
[root@localhost ~]# lsblk
NAME                          MAJ:MIN RM   SIZE RO TYPE  MOUNTPOINT
sr0                            11:0    1  1024M  0 rom
sda                             8:0    0    20G  0 disk
├─sda1                          8:1    0   500M  0 part  /boot
└─sda2                          8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0)   253:0    0  17,6G  0 lvm   /
  └─VolGroup-lv_swap (dm-1)   253:1    0     2G  0 lvm
sdb                             8:16   0   384M  0 disk
├─sdb1                          8:17   0 133,3M  0 part  /tmp
├─sdb2                          8:18   0 133,4M  0 part  /home
└─sdb3                          8:19   0 109,8M  0 part  [SWAP]
sdc                             8:32   0     8G  0 disk
└─sdc1                          8:33   0     8G  0 part
  └─crypt (dm-2)              253:2    0     8G  0 crypt /privado
[root@localhost ~]#
```

6.1. Comprobando los cambios

- ### Creando ficheros en la partición

Creamos un nuevo fichero, de nombre “holamundo” en el punto de montaje correspondiente a nuesto volumen

```
[root@localhost ~]# echo "holamundo" >> /privado/holamundo
[root@localhost ~]# ls /privado/
holamundo  lost+found
```

Vemos que el fichero se ha creado correctamente.

- ### Cerrando el volumen

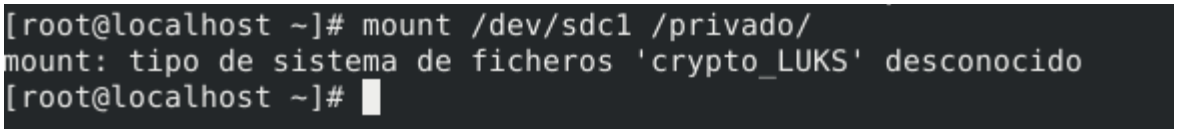
Para comprobar si el cifrado funciona, cerramos el volumen usando la opción *luksClose* y el nombre del volumen, en este caso *crypt,* y ejecutamos un *lsblk*

```
[root@localhost ~]# umount /privado/
[root@localhost ~]# cryptsetup luksClose crypt
[root@localhost ~]# lsblk
NAME                        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0                          11:0    1  1024M  0 rom
sda                           8:0    0    20G  0 disk
├─sda1                        8:1    0   500M  0 part /boot
└─sda2                        8:2    0  19,5G  0 part
  ├─VolGroup-lv_root (dm-0) 253:0    0  17,6G  0 lvm  /
  └─VolGroup-lv_swap (dm-1) 253:1    0     2G  0 lvm
sdb                           8:16   0   384M  0 disk
├─sdb1                        8:17   0 133,3M  0 part /tmp
├─sdb2                        8:18   0 133,4M  0 part /home
└─sdb3                        8:19   0 109,8M  0 part [SWAP]
sdc                           8:32   0     8G  0 disk
└─sdc1                        8:33   0     8G  0 part
```

Vemos que *lsblk* ya no muestra ningún volumen asociado a *sdc1*.

- **Intentando montar la partición con el volumen cerrado**

Con el volumen cerrado, intentamos montar la partición en el directorio */privado* , usando *mount*, de la misma forma que haríamos con cualquier sistema de ficheros

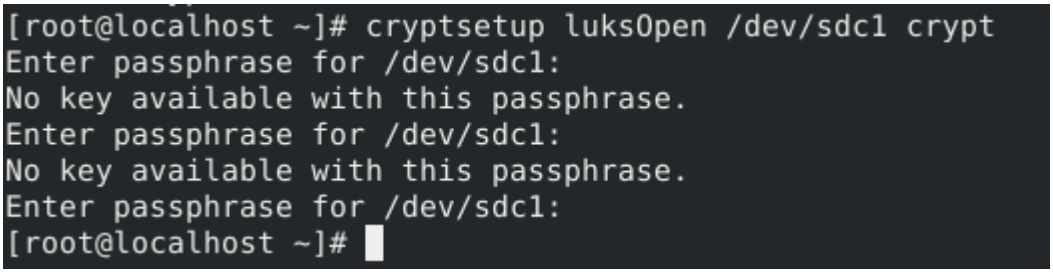


Vemos que, en este caso, *mount* no es capaz de reconocer el sistema de ficheros cifrado de LUKS.

- **Reabriendo el volumen**

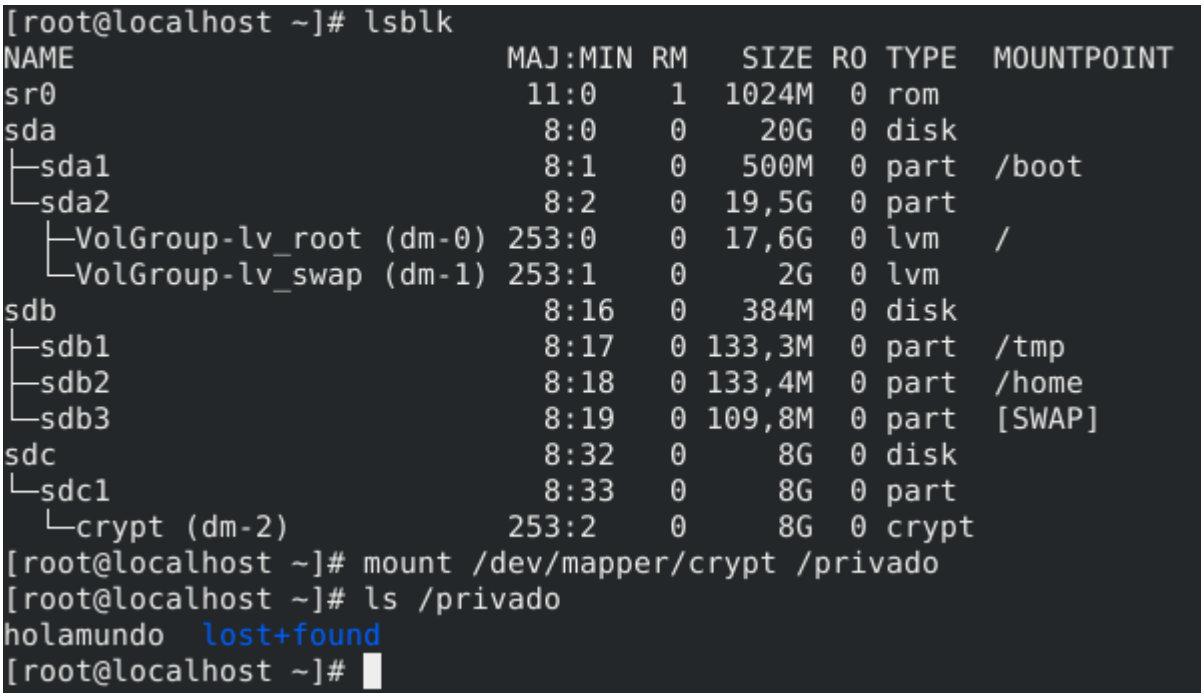
Para poder volver a montar la partición, debemos abrir previamente el volumen asociado a la misma, usando *cryptsetup luksOpen.*

Este comando nos pedirá la clave del volumen. En este caso, intentaremos introducir una clave incorrecta.



Vemos que al poner una clave incorrecta nos suelta un error, y no nos deja abrir el volumen.
En el tercer intento, tras poner la clave correcta, nos permite abrirlo.

Tras abrirlo, ya con la clave correcta, vemos que *lsblk* nos lo vuelve a detectar, y ya podemos montarlo y acceder a sus ficheros sin problema.



7. Conclusión

La gestión de los sistemas de ficheros es una labor bastante importante en nuestro sistema. La buena gestión de estos puede suponer una mayor capacidad de almacenamiento, o la resistencia a una mayor carga de trabajo.

Las herramientas *fdisk* y *mkfs* son muy útiles a la hora de crear y asignar sistemas de ficheros a nuestras particiones.

El uso tanto de estas como de otras herramientas nos ayudará a prevenir futuros errores, o a solucionarlos en caso de que estos sucedan.

Además, el cifrado del sistema de ficheros puede evitar el acceso indeseado a datos delicados. LUKS es una herramienta muy potente para conseguir esto.

En conjunto, todas estas tareas nos permiten tanto aprovechar al máximo los recursos de nuestro disco duro, como añadir capas de seguridad ante errores.