

6 Almacenamiento

6.1 Trabajos prácticos

6.1.1 Práctica con Linux

6.1.1.1 Particiones y sistemas de archivos

Para ofrecer una imagen única del sistema, el sistema operativo debe ofrecer servicios que permitan asociar, y desasociar, unos sistemas de otros de forma transparente en un árbol de nombres único. Además, las utilidades de interpretación de nombres deben ser capaces de saltar de un sistema de archivos a otro sin que sea aparente en ningún momento el nombre del dispositivo físico o lógico que almacena el sistema de archivos. Las dos llamadas al sistema de UNIX que realizan estas operaciones son `mount` y `umount`. La operación de montado permite conectar un sistema de archivos, almacenado en un volumen o partición, a una entrada de directorio del árbol de directorios existente, de manera que al acceder posteriormente a ese directorio estaríamos en realidad accediendo al sistema de archivos almacenado ahí.

Por ejemplo, si en Linux colocamos el comando:

```
ubuntu@ubuntu:~$ mount
```



```
ubuntu@ubuntu:~$ mount
/cow on / type overlayfs (rw)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
/dev/sr0 on /cdrom type iso9660 (ro,noatime)
/dev/loop0 on /rofs type squashfs (ro,noatime)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
tmpfs on /tmp type tmpfs (rw,nosuid,nodev)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs (rw,noexec,nosuid,nodev,size=104857600,mode=0755)
gvfsd-fuse on /run/user/ubuntu/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,user=ubuntu)
ubuntu@ubuntu:~$
```

Figure 6.1: Salida de mount

Nos muestra (como se aprecia en la Fig.6.1) los volúmenes y particiones montadas en el sistema (la primera columna) y el directorio en donde han sido conectados, denominado «punto

de montaje» (la segunda columna), también podemos ver el tipo de sistema de archivos que dicho volumen o partición tiene, por ejemplo: `vfat` para sistemas de archivos de Windows 95 ó 98, `ntfs` para sistemas de archivos de Windows NT, `ext2` ó `ext3` (Extended Second, Extended Third) para sistemas de archivos de Linux. Y además algunas opciones del montaje. Por ejemplo: un sistema de archivos `iso9660`, propio de los CD-ROM, estará montado como «sólo lectura» o *read-only*. En cambio otros pueden estar montados para lectura/escritura o *read-write*.

Los comandos `mount` y `umount` tienen muchas opciones, puede verlas en la documentación en línea, con el comando:

```
ubuntu@ubuntu:~$ man mount
ubuntu@ubuntu:~$ man umount
```

El archivo `/etc/fstab` Estos sistemas de archivos se montan automáticamente durante el arranque de Linux. Estos comandos vistos toman las opciones por defecto del archivo `/etc/fstab` que contiene una descripción de los distintos volúmenes/particiones, con sus respectivos puntos de montaje, tipo de sistema de archivo y opciones adicionales. Véalo con el comando:

```
ubuntu@ubuntu:~$ cat /etc/fstab
```

Nos muestra mucha información que ya nos mostró el comando «`mount`».

El archivo `fstab` contiene información descriptiva acerca de distintos sistemas de archivos. Al archivo `fstab` sólo lo leen los programas, no lo escriben; es deber del administrador de sistemas de crear y mantener este archivo. Cada sistema de archivos se describe en una línea separada, y cada campo se separa por tabulaciones o espacios; las líneas que comienzan con «`#`» son comentarios. El primer campo (o columna) describe el dispositivo especial por bloques o el sistema de archivos remoto a ser montado. El segundo campo describe el punto de montaje para el sistema de archivos.

El tercer campo describe el tipo de sistema de archivos. El cuarto campo describe las opciones de montaje del sistema de archivos. El quinto campo es usado por estos sistemas de archivos por el comando `dump` para determinar qué sistemas de archivos necesitan ser volcados. El sexto campo es usado por el programa `fsck` para determinar el orden en el cual se revisan los sistemas de archivos al momento del arranque.

Particiones Este archivo menciona varias particiones de nuestro disco rígido (`/dev/sda`) ¿Cómo podemos verlas? Como superusuario coloque:

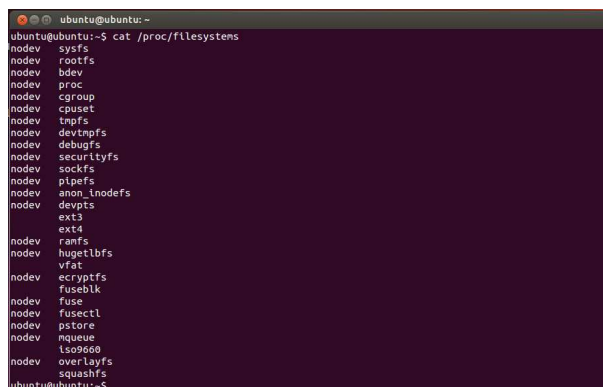
```
ubuntu@ubuntu:~$ sudo fdisk -l /dev/sda
```

El comando `fdisk` manipula la tabla de particiones del disco y, con la opción «`-l`» obtenemos un listado de cada partición, si está marcada como «activa» para que pueda arrancar desde ahí (*boot*), los cilindros que abarca (desde/hasta), y el tipo de partición de la que se trata. De acuerdo con el tipo de partición será el tipo de sistema de archivos que podrá tener (Cuidado: no confundir tipo de partición con tipo de sistema de archivos).

Por ejemplo, verificamos que el pseudo sistema de archivos `proc` está montado en el directorio `«/proc»`, que `«sysfs»` están montado en `«/sys»` y `«tmpfs»` en `«/tmp»`.

Para saber cuáles son los sistemas de archivos que nuestra determinada versión de núcleo de Linux está reconociendo, deberemos observar el contenido del archivo `«/proc/filesystems»` mediante el comando `«cat»`, como vemos en la Fig.6.2, de esta manera:

```
ubuntu@ubuntu:~$ cat /proc/filesystems
```



```
ubuntu@ubuntu:~$ cat /proc/filesystems
nodev sysfs
nodev rootfs
nodev bdev
nodev proc
nodev cgroup
nodev cpuset
nodev tmpfs
nodev devtmpfs
nodev debugfs
nodev securityfs
nodev sockfs
nodev pipefs
nodev anon_inodefs
nodev devpts
nodev ext3
nodev ext4
nodev ramfs
nodev hugetlbfs
nodev vfat
nodev cryptfs
nodev fuseblk
nodev fuse
nodev fusectl
nodev pstore
nodev mqueue
nodev iso9660
nodev overlayfs
nodev squashfs
ubuntu@ubuntu:~$
```

Figure 6.2: Contenido de `/proc/filesystems`

La primera columna dice `«nodev»` para indicar que ese determinado sistema de archivos no es para dispositivos (*no device*, en inglés) mientras que los sistemas `ext3`, `ext4` y `vfat`, por ejemplo, sí lo son.

Todo esto nos permite ver a los archivos, directorios y distintos sistemas de archivos contenidos en varias particiones y dispositivos como un árbol único, todos anclados al sistema raíz indicado mediante el nombre `«/»`, como vemos en la Fig.6.3.

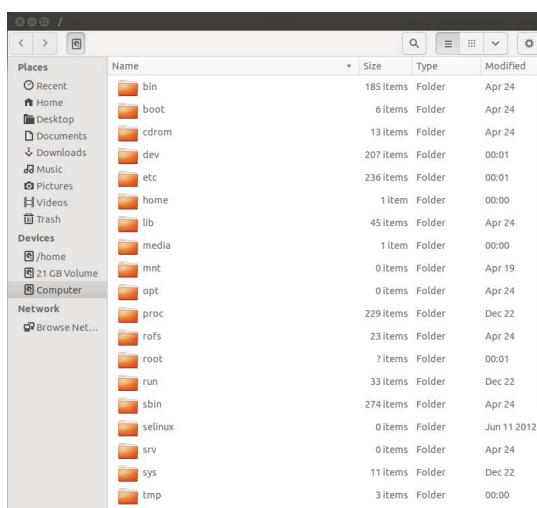


Figure 6.3: Directorios en la raíz

Creación de sistemas de archivos vfat Entonces, antes de poder crear archivos en un dispositivo, éste debe tener previamente un sistema de archivos capaz de alojarlo, por lo tanto, si no lo tuviera, deberíamos crearle uno. A esta operación de creación de sistema de archivos se la suele conocer en la jerga como «formatear», término que proviene del uso del programa `FORMAT` del sistema operativo DOS.

Si vamos a crear un sistema de archivos propio de Linux utilizaremos el comando «`mkfs`» (del inglés *make filesystem*). `mkfs` se lo utiliza para construir un sistema de archivos de Linux en un dispositivo, normalmente en una partición de disco rígido. En realidad, `mkfs` es simplemente un fachada para los verdaderos constructores de sistemas de archivos de Linux, por ejemplo el comando «`mkfs -t ext4`» invoca al comando «`mkfs.ext4`».

Comenzaremos cambiándonos al directorio `Desktop`, para que veamos lo que ocurre tanto desde la CLI como desde la GUI:

```
ubuntu@ubuntu:~$ cd Desktop
ubuntu@ubuntu:~/Desktop$
```

A continuación crearemos un archivo con el que simularemos a un dispositivo, que montaremos a través del *loop device*, pero habiéndole creado previamente un sistema de archivos:

```
ubuntu@ubuntu:~/Desktop$ dd bs=1440k count=1 if=/dev/zero of=midisquete.img
1+0 records in
1+0 records out
1474560 bytes (1.5 MB) copied, 0.00662853 s, 222 MB/s
ubuntu@ubuntu:~/Desktop$
```

El comando «`dd`» (*disk dump*) creará al archivo de salida indicado en el parámetro «`of=`» (en este caso, `midisquete.img`) a partir del archivo de entrada indicado en «`if=`» (en este caso, caracteres nulos), leyendo y escribiendo «`bs=`» bytes, tantas veces como se indique en «`count=`».

A continuación le crearemos un sistema de archivos `vfat`:

```
ubuntu@ubuntu:~/Desktop$ mkfs.vfat midisquete.img
mkfs.vfat 4.1 (2017-01-24)
ubuntu@ubuntu:~/Desktop$
```

Creemos un directorio sobre el cual podamos montar el contenido de nuestro «disquete»:

```
ubuntu@ubuntu:~/Desktop$ mkdir contenido
ubuntu@ubuntu:~/Desktop$
```

En el ambiente gráfico veremos, como en la Fig.6.4, los iconos del directorio y de la imagen recientemente creados:

Ahora procedemos a montar el contenido de «`midisquete.img`» en el directorio `contenido` mediante el dispositivo *loop*. Por tratarse de una labor administrativa, es necesario escalar privilegios a superusuario `root` mediante el comando «`sudo`»:

```
ubuntu@ubuntu:~/Desktop$ sudo mount -o loop,uid=ubuntu midisquete.img contenido
ubuntu@ubuntu:~/Desktop$
```

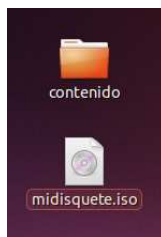


Figure 6.4: Mi disquete

Con el comando «mount» veremos si, efectivamente, la imagen ha quedado montada en el directorio y si listamos el contenido, obviamente estará vacío.

```
ubuntu@ubuntu: ~$ mount
...
/home/ubuntu/Desktop/midisquete.img on /home/ubuntu/Desktop/contenido type vfat (rw,uid=999)
ubuntu@ubuntu: ~/Desktop$ ls contenido
ubuntu@ubuntu: ~/Desktop$
```

Podemos crear un archivo dentro de ese sistema de archivos montado:

```
ubuntu@ubuntu: ~/Desktop$ touch contenido/archivo.txt
ubuntu@ubuntu: ~/Desktop$
```

Pero, por otra parte, en Ubuntu contamos con una aplicación gráfica del escritorio Gnome¹ llamada “Gnome Disk Image Mounter” que vemos en la Fig.6.5 y podemos invocar mediante el comando:

```
ubuntu@ubuntu: ~/Desktop$ gnome-disk-image-mounter
```

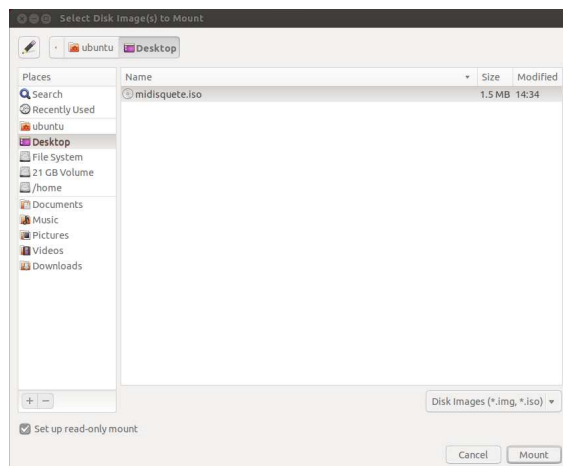


Figure 6.5: Gnome Disk Image Mounter

Y, finalmente, podemos desmontar el sistema de archivos:

¹ Suele estar en el paquete «gnome-disk-utility» y puede instalarse en entornos KDE

```
ubuntu@ubuntu:~/Desktop$ sudo umount contenido
ubuntu@ubuntu:~/Desktop$
```

El archivo «`archivo.txt`» ha quedado dentro de nuestro sistema de archivos «`midisquete.img`». Pruebe usted mismo: Si lista el directorio «`contenido`», verá que está vacío. Recupere el comando con el que montó la imagen utilizando las teclas de cursor hacia arriba ↑ hasta encontrar el comando adecuado y presione la tecla «`Enter`». Al hacerlo, puede aparecer una ventana de Nautilus como la que vemos en la Fig.6.6:

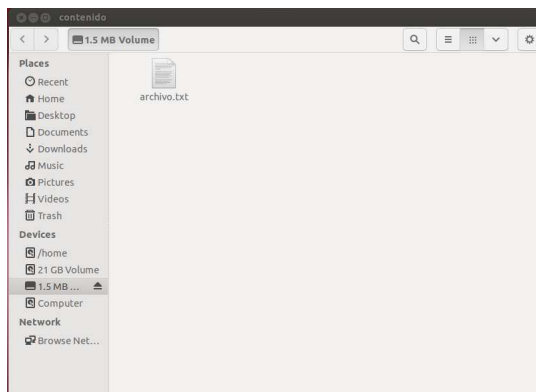


Figure 6.6: Contenido de «`midisquete.img`»

Si usted introduce un *pendrive*, el sistema hará el montaje automáticamente, y abrirá una ventana de Nautilus para mostrar el contenido. Desde la línea de comandos, si quiere ver cómo está montado, con el comando «`mount`» verá:

```
ubuntu@ubuntu:~$ mount
...
/home/ubuntu/Desktop/midisquete.img on
    /home/ubuntu/Desktop/contenido type vfat (rw,uid=999)
/home/sdb1 on /media/ubuntu/PENDRIVE type vfat (rw,uid=999, ...)
ubuntu@ubuntu:~/Desktop$
```

Si bien en esta Práctica hemos forzado, de alguna manera, a efectuar estas operaciones de montaje dentro de nuestro «`Escritorio`», lo habitual es que se utilice el directorio

«`/media/<usuario>/<rotulo>`»

como punto de montaje. Por este motivo vemos montado nuestro *pendrive* en

«`/media/ubuntu/PENDRIVE`»

(el nombre del rótulo depende del que tenga el dispositivo).

Comprobar sistemas de archivos vfat Normalmente es una buena idea el comprobar de vez en cuando los sistemas de archivos en busca de archivos dañados o corruptos.

En caso de un apagado inesperado provocado, por ejemplo, por un fallo en el suministro de energía eléctrica, los sistemas de archivos no serán desmontados como corresponde. Esto deja al sistema de archivos «marcado» de manera que, al volver el suministro de energía eléctrica o al utilizar el dispositivo en otro equipo, el sistema detectará esta situación e intentará repararlo detectando errores en el sistema de archivos.

Esto ocurre con todos los sistemas de archivo excepto los de sólo lectura.

En Linux contamos con el programa «`f sck`» (del inglés *file system check*) que se encarga de comprobar y corregir errores encontrados en distintos sistemas de archivo. En este caso que nos ocupa y de manera similar a lo que ocurre con «`mkfs`», la versión específica del programa para comprobar sistemas de archivo FAT32 es «`f sck . vfat`».

Advertencia: El dispositivo debe estar desmontado antes de efectuarle la comprobación.

Entonces, si queremos comprobar nuestro archivo de imagen de disquete creado para esta Práctica, deberíamos colocar:

```
ubuntu@ubuntu:~$ fsck.vfat -l -v midisquete.img
fsck.fat 4.1 (2017-01-24)
Checking we can access the last sector of the filesystem
Boot sector contents:
System ID "mkfs.fat"
Media byte 0xf0 (5.25" or 3.5" HD floppy)
    512 bytes per logical sector
    512 bytes per cluster
    1 reserved sector
First FAT starts at byte 512 (sector 1)
    2 FATs, 12 bit entries
    4608 bytes per FAT (= 9 sectors)
Root directory starts at byte 9728 (sector 19)
    224 root directory entries
Data area starts at byte 16896 (sector 33)
    2847 data clusters (1457664 bytes)
18 sectors/track, 2 heads
    0 hidden sectors
    2880 sectors total
Checking file /archivo.txt (ARCHIVO.TXT)
Checking for unused clusters.
midisquete.img: 1 files, 0/2847 clusters
ubuntu@ubuntu:~$
```

Como era de esperar, nuestra imagen no contiene errores. Pero la comprobación nos permite ver los detalles del sistema de archivos. También podemos valernos del comando «`file`» como vemos en la Fig.6.7.

```

ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ file midisquete.img
midisquete.img: DOS/MBR boot sector, code offset 0x3c+2, OEM-ID "mkfs.fat", root entries 224,
sectors 2880 (volumes <=32 MB), sectors/FAT 9, sectors/track 18, serial number 0x7e139696, unl
abeled, FAT (12 bit), followed by FAT
ubuntu@ubuntu:~$

```

Figure 6.7: «file» en una imagen de floppy

Creación de sistemas de archivos ext4 Con nuestro sistema de archivos en «midisquete.iso» desmontado previamente vamos a crearle un sistema de archivos distinto pero sobre un «dispositivo» mas grande. Esta operación borra su contenido. Entonces:

```

ubuntu@ubuntu:~$ dd bs=2880k count=1 if=/dev/zero of=midisquete.img
1+0 records in
1+0 records out
2949120 bytes (2,9 MB, 2,8 MiB) copied, 0,0134728 s, 219 MB/s
ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ mkfs.ext4 -v midisquete.img
mke2fs 1.44.1 (24-Mar-2018)
fs_types for mke2fs.conf resolution: 'ext4', 'floppy'
Discarding device blocks: done
Discard succeeded and will return 0s - skipping inode table wipe
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=0 blocks
360 inodes, 2880 blocks
144 blocks (5.00%) reserved for the super user
First data block=1
Maximum filesystem blocks=3014656
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done
ubuntu@ubuntu:~$

```

Note que cambiamos a «bs=2880k», es decir el doble que en el anterior «disquete» porque en un dispositivo tan pequeño no cabe bien la información de **bitácora** o *journaling*.

Obtener información del sistema de archivos ext2, ext3 o ext4 Al crear cualquiera de estos tres sistemas de archivos se nos muestra información acerca de sus parámetros pero si quisiéramos más información detallada deberíamos recurrir a «dumpe2fs»:

```

ubuntu@ubuntu:~$ dumpe2fs midisquete.img
dumpe2fs 1.44.1 (24-Mar-2018)
Filesystem volume name: <none>
Last mounted on: <not available>

```


6.1 Trabajos prácticos

```
Filesystem UUID:          c41f677c-cc87-4a4d-a385-cef38a2f267e
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype extent
                          64bit flex_bg sparse_super large_file huge_file dir_nlink extra_isize metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:      Linux
Inode count:              360
Block count:              2880
Reserved block count:    144
Free blocks:              1770
Free inodes:              349
First block:              1
Block size:               1024
Fragment size:           1024
Group descriptor size:    64
Reserved GDT blocks:     22
Blocks per group:        8192
Fragments per group:    8192
Inodes per group:        360
Inode blocks per group:  45
Flex block group size:   16
Filesystem created:      Sat Oct  6 22:51:19 2018
Last mount time:         n/a
Last write time:         Sat Oct  6 22:51:19 2018
Mount count:              0
Maximum mount count:     -1
Last checked:             Sat Oct  6 22:51:19 2018
Check interval:          0 (<none>)
Lifetime writes:         42 kB
Reserved blocks uid:     0 (user root)
Reserved blocks gid:     0 (group root)
First inode:              11
Inode size:               128
Journal inode:            8
Default directory hash:  half_md4
Directory Hash Seed:     3f50ea74-3dfb-4441-a29e-a404e6856a37
Journal backup:          inode blocks
Checksum type:           crc32c
Checksum:                 0x70981c8e
Journal features:         (none)
Journal size:             1024k
Journal length:          1024
Journal sequence:        0x00000001
Journal start:           0

Group 0: (Blocks 1-2879) csum 0xf324 [ITABLE_ZEROED]
  Primary superblock at 1, Group descriptors at 2-2
  Reserved GDT blocks at 3-24
  Block bitmap at 25 (+24), csum 0x2c64b634
  Inode bitmap at 41 (+40), csum 0x847c6bcb
  Inode table at 57-101 (+56)
  1770 free blocks, 349 free inodes, 2 directories, 349 unused inodes
  Free blocks: 1110-2879
  Free inodes: 12-360
ubuntu@ubuntu:~$
```

El UUID o *Universal Unique Identifier* (Identificador Único Universal) es un número de 128 bit (16 bytes) usado para identificar información en sistemas informáticos, mostrado en cinco grupos separados por guiones. También se utiliza el término Identificador Único Global (o *Globally Unique Identifier*, GUID). Muchas veces se lo utiliza para montar el dispositivo.

Cambiar los parámetros del sistema de archivos Ya está creado el sistema de archivos ext4, entonces para ver los parámetros del sistema de archivos usamos el comando «`tune2fs -l midisque .img`», como apreciamos en la Fig.6.8:

```

ubuntu@ubuntu:~$
ubuntu@ubuntu:~$ tune2fs -l midisque.img
tune2fs 1.44.1 (24-Mar-2018)
Filesystem volume name: <none>
Last mounted on: <not available>
Filesystem UUID: c41f677c-ccb7-4a4d-a385-cef38a2f267e
Filesystem magic number: 0xEF53
Filesystem revision #: 1 (dynamic)
Filesystem features: has_journal ext_attr resize_inode dir_index filetype extent 64bit flex_bg sparse_super large_file huge_file dir_nlink
extra_isize metadata_csum
Filesystem flags: signed_directory_hash
Default mount options: user_xattr acl
Filesystem state: clean
Errors behavior: Continue
Filesystem OS type: Linux
Inode count: 360
Block count: 2880
Reserved block count: 144
Free blocks: 1770
Free inodes: 349
First block: 1
Block size: 1024
Fragment size: 1024
Group descriptor size: 64
Reserved GDT blocks: 22
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 360
Inode blocks per group: 45
Flex block group size: 16
Filesystem created: Sat Oct 6 22:51:19 2018
Last mount time: n/a
Last write time: Sat Oct 6 22:51:19 2018
Mount count: 0
Maximum mount count: -1
Last checked: Sat Oct 6 22:51:19 2018
Check interval: 0 (<none>)
Lifetime writes: 42 kB
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
First inode: 11
Inode size: 128
Journal inode: 8
Default directory hash: half_md4
Directory Hash Seed: 3f50ea74-3dfb-4441-a29e-a404e6856a37
Journal backup: inode blocks
Checksum type: crc32c
Checksum: 0x70981c8e
ubuntu@ubuntu:~$

```

Figure 6.8: `tune2fs -l midisque.img`

El comando «`tune2fs`» sirve para ajustar los parámetros del sistema de archivos. La opción «`-l`» nos lista estos parámetros.

Hasta ahora hemos visto cómo aplicar estos comandos a un archivo que nos ha hecho las veces de «dispositivo» al que le pudimos crear sistemas de archivos y conocer sus parámetros. Esto lo hacemos para no poner en riesgo ni dañar nuestros verdaderos dispositivos que seguramente estarán en uso. Ahora nos valdremos de un disco virtual que se crea en memoria RAM.

Un disco SCSI en RAM Por suerte Linux posee muchos recursos para ello. Por ejemplo, ahora vamos a crear un disco rígido virtual de tecnología SCSI al que el núcleo de Linux lo detectará como tal. Podemos tratarlo como a un verdadero disco rígido. Al crearlo, no tendrá particiones definidas, tal como un disco rígido nuevo. Le crearemos particiones y los sistemas de archivos que se nos ocurra, que podremos montar y desmontar a nuestro antojo. Eso sí, al estar creado en memoria volátil, los cambios no serán permanentes.

Se trata del manejador del adaptador «scsi_debug» el cual es capaz de simular un número variable de discos SCSI. Es similar a los discos RAM que ya hemos utilizado en cuanto a su volatilidad pero nos permite definirle muchas características «técnicas» que encontramos en los verdaderos discos SCSI. Nosotros haremos un uso sencillo de él. Se crea mediante el comando:

```
ubuntu@ubuntu:~$ sudo modprobe scsi_debug
ubuntu@ubuntu:~$
```

El comando «modprobe» carga un módulo en memoria. Un **módulo** es un trozo de software perteneciente al núcleo de Linux. Puede cargarse y descargarse del núcleo a voluntad, cada vez que se lo necesite.

Muchas de las cosas que ocurren en el núcleo y dignas de mencionar quedan registradas en archivos. Con el comando «dmesg» podemos ver notificaciones del núcleo, desde que arrancó hasta la actualidad. Entonces, si colocamos el comando «dmesg» veremos cómo el núcleo de Linux detectó al disco nuevo:

```
ubuntu@ubuntu:~$ dmesg
...
[ 2663.773686] scsi_debug: host protection
[ 2663.773697] scsi2 : scsi_debug, version 1.81 [20070104], dev_size_mb=8, opts=0x0
[ 2663.780341] scsi 2:0:0:0: Direct-Access Linux scsi_debug 0004 PQ: 0 ANSI: 5
[ 2663.783401] sd 2:0:0:0: Attached scsi generic sg2 type 0
[ 2663.785033] sd 2:0:0:0: [sdb] 16384 512-byte logical blocks: (8.38 MB/8.00 MiB)
[ 2663.786071] sd 2:0:0:0: [sdb] Write Protect is off
[ 2663.786075] sd 2:0:0:0: [sdb] Mode Sense: 73 00 10 08
[ 2663.788075] sd 2:0:0:0: [sdb] Write cache: enabled, read cache: enabled, supports DPO and FUA
[ 2663.792109] sdb: unknown partition table
[ 2663.798024] sd 2:0:0:0: [sdb] Attached SCSI disk
ubuntu@ubuntu:~$
```

Tal como era de esperar, lo detecta como un disco «sd». Como en nuestra computadora ya tenemos un disco rígido, de tecnología SATA, también detectado como disco «sd» y, por ser el primero, llamado «sda»; éste nuevo disco lo denomina «sdb». La línea que dice «[sdb] Attached SCSI disk» así lo comprueba. También hay una línea que dice «unknown partition table» (tabla de partición desconocida) porque se trata de un disco nuevo que no tiene particiones definidas.

Por tratarse de un disco de tecnología SCSI podemos usar el comando «lsscsi»²:

```
ubuntu@ubuntu:~$ lsscsi -s
[0:0:0:0] disk ATA ST9120822AS D /dev/sda 120GB
[1:0:0:0] cd/dvd TSSTcorp DVD+-RW TS-L632D DE04 /dev/sr0 -
[2:0:0:0] disk Linux scsi_debug 0186 /dev/sdb 8.0MB
ubuntu@ubuntu:~$
```

Mediante el programa «fdisk» crearemos una única partición en «/dev/sdb» que abarque a toda la capacidad del disco y sobre la cual ejercitaremos el crear sistemas de archivos, de una manera más cercana a la realidad que enfrentaremos más adelante.

```
ubuntu@ubuntu:~$ sudo fdisk /dev/sdb
Device contains neither a valid DOS partition table, nor Sun, SGI or OSF disklabel
```

²Se instala con «apt install lsscsi».

```

Building a new DOS disklabel with disk identifier 0x419b4dd6.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.
Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)
Command (m for help):

```

El mensaje nos advierte que el dispositivo no tiene tablas de particiones válidas, que creará un rótulo tipo DOS y que todos los cambios permanecerán pendientes hasta que decidamos escribirlos en el disco rígido (sdb).

El prompt «Command (m for help):» nos indica que si necesitamos una ayuda acerca de cuales son los comandos que podemos introducir dentro del programa «fdisk», tenemos que teclear la «m». Al hacerlo vemos la siguiente salida:

```

Command action
  a  toggle a bootable flag
  b  edit bsd disklabel
  c  toggle the dos compatibility flag
  d  delete a partition
  l  list known partition types
  m  print this menu
  n  add a new partition
  o  create a new empty DOS partition table
  p  print the partition table
  q  quit without saving changes
  s  create a new empty Sun disklabel
  t  change a partition's system id
  u  change display/entry units
  v  verify the partition table
  w  write table to disk and exit
  x  extra functionality (experts only)
Command (m for help):

```

Entonces, el comando «p» nos muestra por pantalla la tabla de particiones, que en este momento está vacía:

```

Disk /dev/sdb: 8 MB, 8388608 bytes
8 heads, 32 sectors/track, 64 cylinders
Units = cylinders of 256 * 512 = 131072 bytes
Disk identifier: 0x419b4dd6
Device Boot Start End Blocks Id System
Command (m for help):

```

Nuestro disco SCSI «/dev/sdb» tiene una capacidad de 8 MB, 8 cabezas, 32 sectores y 64 cilindros. A continuación, con el comando «n», crearemos una partición nueva, será primaria, tendrá el número 1 y abarcará todo el disco, desde el primer hasta el último cilindro.

```

Command (m for help): n
Command action
  e  extended
  p  primary partition (1-4)
p
Partition number (1-4): 1
First cylinder (1-64, default 1): 1

```

```
Last cylinder, +cylinders or +size{K,M,G} (1-64, default 64): 64
Command (m for help):
```

En este caso hemos particionado por cilindros pero también podemos particionar por sectores. Nuevamente, el comando «p» nos mostrará la partición que acabamos de crear:

```
Command (m for help): p

Disk /dev/sdb: 8 MB, 8388608 bytes
8 heads, 32 sectors/track, 64 cylinders
Units = cylinders of 256 * 512 = 131072 bytes
Disk identifier: 0x419b4dd6

   Device   Boot    Start    End    Blocks    Id    System
/dev/sdb1             1     64     8176    83    Linux

Command (m for help):
```

Efectivamente, la última línea nos muestra la partición «/dev/sdb1» que comienza en el cilindro 1 y finaliza en el 64. Tiene una capacidad de 8 MB. Si hubiéramos particionado por sectores a este disco tendríamos que la partición va desde el sector 2048 al 14336, por ejemplo. La columna «Id» nos muestra el **tipo de partición**, que no es lo mismo que el sistema de archivos que contiene. La columna «Boot» nos muestra si la partición es arrancable por la BIOS; si está marcada como «activa», según la terminología de Windows. Al crear una partición no se marca como «activa» a menos que lo hagamos mediante el comando «a» dentro de «fdisk». Además las crea para Linux, como vemos en la columna «Id» y «System». Para cambiar el tipo de partición debemos usar el comando «t» de «fdisk».

Con el comando «w» confirmamos y escribimos definitivamente los datos al disco.

```
Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
ubuntu@ubuntu:~$
```

El comando «dmesg» nos mostrará en su última línea que detectó la creación de la partición «sdb1». Ya podemos crearle un sistema de archivos, por ejemplo, un «ext4»:

```
ubuntu@ubuntu:~$ sudo mkfs.ext4 /dev/sdb1
mke2fs 1.42.5 (29-Jul-2012)
Filesystem label=
OS type: Linux
Block size=1024 (log=0)
Fragment size=1024 (log=0)
Stride=0 blocks, Stripe width=32 blocks
1792 inodes, 7168 blocks
358 blocks (4.99%) reserved for the super user
First data block=1
Maximum filesystem blocks=7340032
1 block group
8192 blocks per group, 8192 fragments per group
```

```

1792 inodes per group
Allocating group tables: done
Writing inode tables: done
Creating journal (1024 blocks): done
Writing superblocks and filesystem accounting information: done
ubuntu@ubuntu:~$

```

A las particiones se les puede colocar una «Etiqueta» (o «*Label*», en inglés) y que se las puede utilizar para montar la partición haciendo referencia a ella. En este caso, vemos en la línea «`Filesystem label=`» que esta partición no tiene etiqueta. La llamaremos «`mi-scsi`» y se la podemos colocar mediante el comando:

```
ubuntu@ubuntu:~$ sudo e2label /dev/sdb1 "mi-scsi"
```

Al crear el sistema de archivos aparece un icono nuevo de disco rígido en el Lanzador de Aplicaciones, el que, como ya dijimos, al picarlo con el mouse se monta y se muestra su contenido con Nautilus.

Puede desmontar la unidad mediante el comando «`umount`» o desde Nautilus, picando sobre el icono para desmontar.



o bien picando con el botón derecho del mouse y eligiendo la opción para desmontar.

Como este disco ha sido creado en memoria RAM no persistirá luego de reiniciado el sistema. Tampoco si descargamos el módulo «`scsi_debug`» de memoria, de esta manera:

```
ubuntu@ubuntu:~$ sudo modprobe -r scsi_debug
```

Vea cómo detectó la remoción del disco el núcleo de Linux con el comando «`dmesg`».

Desde la GUI También puede probar el particionado y creación de sistema de archivos con el programa `Gparted`³. Primero cargue nuevamente el módulo «`scsi_debug`» si lo descargó o reinició su equipo

```
ubuntu@ubuntu:~$ sudo modprobe scsi_debug dev_size_mb=16
```

así el disco tendrá 16 Mb en vez de 8, y a continuación invoque al programa desde la línea de comandos de esta manera:

```
ubuntu@ubuntu:~$ sudo gparted /dev/sdb
```

Estará editando su disco «`/dev/sdb`» creado gracias a «`scsi_debug`» y no su verdadero disco rígido, como vemos en la Fig.6.9:

³Se instala fácilmente con «`sudo apt install gparted`».

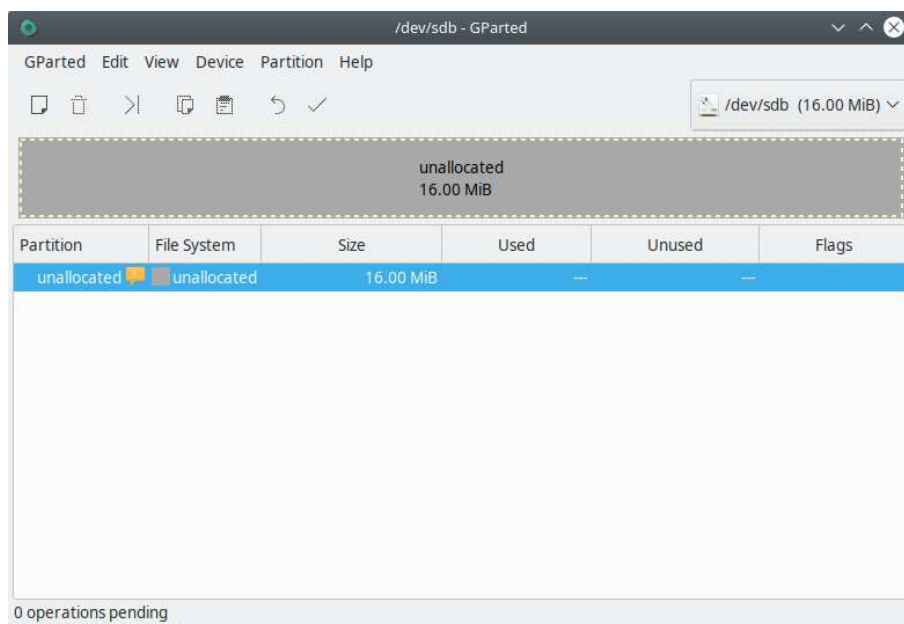


Figure 6.9: gparted sobre /dev/sdb

Con esta aplicación podemos crear, eliminar, redimensionar, inspeccionar y copiar particiones, como así también los sistemas de archivos que se encuentren en ellas. Es muy útil para crear espacio para nuevas particiones y sistemas operativos, reorganizando el uso del disco. Una ventaja fundamental, frente a «fdisk», además de lo intuitivo de uso gracias a la interfaz gráfica, es que al redimensionar no se pierde el sistema de archivos. Es decir, podemos encoger una partición con sistema de archivos NTFS para hacer lugar a otra que contendrá a Linux, pero sin perder el sistema de archivos y, por consiguiente, todos los archivos que tenemos en él.

Nuestro disco tendrá ahora dos particiones: la «/dev/sdb1» de 10 MB con sistema ext4 y la restante «/dev/sdb2» será nuestra partición de intercambio o *swap*.

Vaya a «Dispositivo» en el menú descolgable y de ahí elija «Crear tabla de particiones». Si bien -correctamente- nos advierte que se eliminarán todos los datos en el disco entero, no hay de qué preocuparse porque se trata de un disco creado en memoria RAM, como ya dijimos. El crear una tabla de particiones de tipo MS-DOS es lo habitual si usted tiene una computadora personal (PC) o una notebook. Todo el disco aparecerá «sin asignar». Al picar con el mouse sobre esa línea seleccionamos el espacio que queremos particionar. Luego de picar en «Partición» podemos seleccionar la opción «Nueva». Note cómo aparece en la nueva ventana de «Crear una partición nueva» mucha de la información que ya vimos con «fdisk». También podemos picar con la tecla derecha del mouse para operar tanto sobre espacios sin asignar como sobre particiones.

Todas las operaciones de su diseño quedarán pendientes hasta que las confirme con el icono de verificación.



Puede alterar el tamaño con el icono para alterar las dimensiones de la partición.



Ahora que ya tenemos creado a nuestro disco, con sus particiones y sus sistemas de archivos, podemos ver al dispositivo perfectamente creado, como en la Fig.6.10:

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ ls -l /dev/sdb*
brw-rw---- 1 root disk 8, 16 ene  4 22:54 /dev/sdb
brw-rw---- 1 root disk 8, 17 ene  4 22:53 /dev/sdb1
brw-rw---- 1 root disk 8, 18 ene  4 22:52 /dev/sdb2
ubuntu@ubuntu:~$
```

Figure 6.10: `ls -l /dev/sdb*`

Acá simplemente creamos dos particiones, la «`/dev/sdb1`» que contendría a todo nuestro sistema de archivos Linux, y a «`/dev/sdb2`» que es la partición de intercambio.

«`blkid`» es un comando que nos muestra los atributos de los dispositivos por bloques. En nuestro ejemplo lo veríamos como en la Fig.6.11:

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ blkid
/dev/loop0: TYPE="squashfs"
/dev/sda1: SEC_TYPE="msdos" LABEL="DellUtility" UUID="07D7-0714" TYPE="vfat"
/dev/sda5: TYPE="swap"
/dev/sda6: UUID="bc24e289-0c67-4e23-bfee-4aa10cd67370" TYPE="ext4"
/dev/sda7: LABEL="/home" UUID="649e0bed-4737-4a3e-b94f-8586d78711e8" SEC_TYPE="ext2" TYPE="ext3"
/dev/sr0: LABEL="Ubuntu 13.04 i386" TYPE="iso9660"
/dev/sdb1: LABEL="/" UUID="19302617-b35d-4de4-b869-b6f159e27178" TYPE="ext4"
/dev/sdb2: UUID="e0051031-2b32-49af-9bda-6f5f0c09b004" TYPE="swap"
ubuntu@ubuntu:~$
```

Figure 6.11: `blkid`

Vemos claramente la etiqueta, el UUID y el tipo de sistema de archivos. De manera similar, como vemos en la Fig.6.12, en el directorio «`/dev`» podemos ver a los discos listados por su UUID:

Son enlaces que apuntan a los nombres de las particiones.


```
ubuntu@ubuntu:~$ ls -l /dev/disk/by-uuid/
total 0
lrwxrwxrwx 1 root root 10 ene  5 00:12 07D7-0714 -> ../../sda1
lrwxrwxrwx 1 root root 10 ene  5 00:20 19302617-b35d-4de4-b869-b6f159e27178 -> ../../sdb1
lrwxrwxrwx 1 root root 10 ene  5 00:12 649e0bed-4737-4a3e-b94f-8586d78711e8 -> ../../sda7
lrwxrwxrwx 1 root root 10 ene  5 00:12 bc24e289-0c67-4e23-bfee-4aa10cd67370 -> ../../sda6
lrwxrwxrwx 1 root root 10 ene  5 00:20 e0051031-2b32-49af-9bda-6f5f0c09b004 -> ../../sdb2
ubuntu@ubuntu:~$
```

Figure 6.12: `ls -l /dev/disk/by-uuid/`