TEMA 11: GESTIÓN DEL ALMACENAMIENTO EN LINUX

Índice

1. Introducción

Vistos ya cómo podemos desenvolvernos en Linux con los comandos de gestión de ficheros y directorios y de administración del sistema, vamos a estudiar una parte importante de la administración como es la gestión del almacenamiento.

Cuando estudiamos el Windows Server vimos los sistemas de ficheros que utilizaba este sistema operativo, la gestión de los discos, particiones y volúmenes, así como las copias de seguridad y los discos RAID. En este tema vamos a intentar trasladar esos contenidos al sistema operativo Linux.

2. Sistemas de ficheros en Linux

Linux soporta gran variedad de sistemas de ficheros, desde sistemas basados en discos, como pueden ser **ext4, ReiserFS, BtrFS, XFS, JFS, FAT32 o NTFS**, a sistemas de ficheros que sirven para comunicar equipos en la red de diferentes sistemas operativos, como **NFS** (utilizado para compartir recursos entre equipos Linux) o **SMB** (para compartir recursos entre máquinas Linux y Windows).

Los sistemas de ficheros indican el modo en que se gestionan los ficheros dentro de las particiones. Según su complejidad, tienen características como previsión de apagones, posibilidad de recuperar datos, indexación para búsquedas rápidas, reducción de la fragmentación para agilizar la lectura de los datos, etc. Hay varios tipos, normalmente ligados a sistemas operativos concretos

Sistema de ficheros EXT4

El sistema de ficheros más utilizado en Linux es el denominado **EXT4** (Ext4fs) o *fourth extended filesystem*. Es la última versión de la familia *ext* y significa un salto cualitativo importante en los sistemas de ficheros. Es una evolución del original *EXT2* y *EXT3*.

El sistema de ficheros *ext2* era el sistema nativo de Linux, históricamente, aunque al principio se utilizaba MinixFS. Fue creado a finales de los años noventa y tiene la reputación de ser un sistema de archivo confiable. El problema principal de este sistema

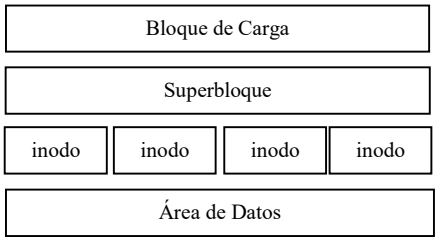
de ficheros es que las escrituras a disco se realizan de una manera asíncrona, es decir, que si se produce un fallo en el sistema, se puede perder la información que se está escribiendo, problema que se solucionó con la aparición de nuevas versiones de este sistema de ficheros como la ***ext3***, básicamente, el mismo sistema de ficheros *ext2* con registro por diario, aunque también incorporaba otras mejoras en cuanto al almacenamiento de los datos. El resultado de esta mejora es un sistema de ficheros tan confiable como *ext2*, pero capaz de recuperarse de fallos en el sistema de una manera más rápida.

La última versión, la *ext4*, mejora considerablemente las anteriores aportando:

- Soporte para ficheros de hasta 1 EB y archivos de hasta 16 TB, gracias a un cambio sustancial en la manera de almacenar la información.
- Uso mejor del procesador.
- Mejoras en la velocidad de lectura y escritura.

El funcionamiento de EXT es similar al de FAT en el modo de que funciona también como el índice de un libro, si bien su estructura lógica y modo de actuar difiere bastante.

Gráficamente, la estructura lógica sería:



El **bloque de carga** o bloque 0 de cada sistema está reservado para almacenar un programa que utiliza el sistema para gestionar el resto de partes del FS.

El **superbloque** o bloque uno contiene información sobre el sistema de archivos.

La **tabla de i-nodos** es el equivalente a las entradas de la FAT (MFT) del sistema de ficheros NTFS. Por cada archivo, Linux tiene asociado un elemento en esta tabla que contiene un número. Este número identifica la ubicación del archivo dentro del área de datos.

Por último, la **zona de datos**, que ocupa el resto del disco, es la equivalente a la zona de datos de la FAT. En esta zona se almacenan los ficheros y directorios del sistema.

Sistema de ficheros Btrfs

Btrfs (B-tree FS o normalmente pronunciado "Butter FS") es un sistema de archivos que utiliza la tecnología COW (copy-on-write). Fue desarrollado por la empresa Oracle con la participación de Red Hat, SUSEe Intel entre otras compañías, como sustitución de EXT4.

Incorpora características avanzadas de almacenamiento, mayor eficiencia en operaciones de lectura y escritura, tolerancia a fallos, reparación y fácil administración.

Es recomendable para discos especialmente grandes, que es donde más innova *btrfs* y se pueden notar sus mejoras respecto a otros sistemas de ficheros. Tiene soporte para las últimas tecnologías de discos.

En 2015 *Btrfs* fue adoptado como sistema de ficheros por defecto de la distribución SUSE Linux Enterprise Server.

Sistema de ficheros XFS

XFS es el más antiguo de los sistemas de ficheros que utilizan *journaling* (registro por diario), lo cual permite restablecer datos en caso de que la transacción de escritura en disco falle.

Es un sistema de ficheros de alto rendimiento soportado por la mayoría de las distribuciones Linux. La versión 7 de Red Hat Enterprise Linux (RHEL) utiliza XFS como el sistema de ficheros por defecto.

Al igual que *Btrfs*, está recomendado para grandes sistemas de almacenamiento donde se busca seguridad y eficiencia.

3. Dispositivos de almacenamiento

En Linux, la gestión de dispositivos de almacenamiento es totalmente distinta a como lo hace el sistema operativo Windows. Para empezar, el sistema de ficheros de Windows presentaba tantos árboles como unidades lógicas había en el sistema, donde la raíz de cada árbol venía representado por una letra que identificaba a la unidad en cuestión (C:, D:, E:, …). En Linux sólo hay un árbol en el sistema de ficheros y las distintas unidades o particiones de disco se montan en directorios, exceptuando la unidad principal que se monta en el directorio raíz (/).

En el directorio */dev* se almacenan las definiciones de los dispositivos. A cada tipo de dispositivo le corresponde un fichero en este directorio. Es importante saber la nomenclatura de estos ficheros para saber a qué dispositivo hacen referencia. En caso de dispositivos de disco con varias particiones, a esta también les corresponde un fichero distinto.

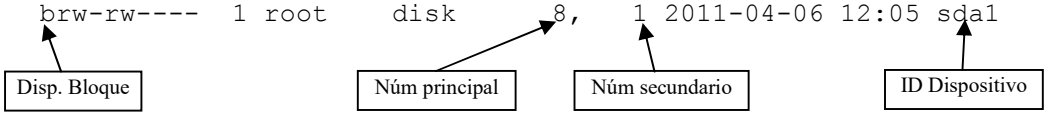
Un contenido típico (muy resumido) de este directorio sería:

```
#ls /dev -l
brw-rw----  1 root    disk      8,   0 2011-05-11 11:33 sda
brw-rw----  1 root    disk      8,   1 2011-04-06 12:05 sda1
brw-rw----  1 root    disk      8,   2 2011-05-11 11:33 sda2
brw-rw----  1 root    disk      8,   5 2011-04-06 12:05 sda5
crw-rw----  1 root    cdrom    21,   0 2011-04-06 12:05 sg0
crw-rw----  1 root    disk     21,   1 2011-04-06 12:05 sg1
drwxrwxrwt  2 root    root         180 2011-05-06 12:18 shm
crw-rw----  1 root    root     10, 231 2011-04-06 12:05 snapshot
brw-rw----+ 1 root    cdrom    11,   0 2011-04-06 12:05 sr0
crw-rw-rw-  1 root    tty       5,   0 2011-05-04 12:50 tty
crw--w----  1 root    root      4,   0 2011-04-06 12:05 tty0
crw--w----  1 root    tty       4,  11 2011-04-06 12:05 tty11
crw-------  1 root    root      4,   2 2011-04-06 12:05 tty2
```

Donde se aprecia los distintos tipos de dispositivos (bloques o caracteres) marcados por el tipo de fichero y el nombre del dispositivo que indica qué dispositivo es y qué lugar ocupa.

Ejemplo:

```
brw-rw----  1 root    disk      8,   1 2011-04-06 12:05 sda1
```



En este caso nos indica que **sda1** es un *dispositivo de bloque*, identificado por su número principal 8, y al ser su número secundario un 1 nos indica que es la primera partición primaria del disco. El nombre del dispositivo recoge todos estos datos de manera que *sda* nos indica que es el primer disco y el 1 nos indica la primera partición.

Esta nomenclatura de *sda1* se puede especificar de manera general de la siguiente forma:

- Disqueteras
    - Primera disquetera: */dev/fd0* (en Windows sería la disquetera A:)
    - Segunda disquetera: */dev/fd1*
- Discos duros (en general: /dev/sd*x*#, donde *x* es el disco y # es la partición)
    - Primer disco duro: (todo el disco) */dev/sda*
        - Particiones primarias
            - Primera partición primaria: */dev/sda1*
            - Segunda partición primaria: */dev/sda2*
            - Tercera partición primaria: */dev/sda3*
            - Cuarta partición primaria: */dev/sda4*
        - Particiones lógicas
            - Primera partición lógica: */dev/sda5*
            - Sucesivamente: */dev/sda#*
    - Segundo disco duro: (todo el disco) */dev/sdb*
        - Particiones primarias
            - Primera partición primaria: */dev/sdb1*
            - Segunda partición primaria: */dev/sdb2*
            - Tercera partición primaria: */dev/sdb3*
            - Cuarta partición primaria: */dev/sdb4*
        - Particiones lógicas
            - Primera partición lógica: */dev/sdb5*
            - Sucesivamente: */dev/sdb#*

- Discos IDE: (igual que los anteriores, pero cambiando **sd**xx por **hd**xx)
    - Disco maestro primer controlador IDE: */dev/hda*
    - Disco esclavo primer controlador IDE: */dev/hdb*
    - Disco maestro segundo controlador IDE: */dev/hdc*
    - Sucesivamente ...
- Primer CD-ROM: */dev/scd0*, también conocido como */dev/sr0*

Actualmente, todos los discos, independientemente de la tecnología utilizada (IDE, SATA, SCSI, memorias Flash, …) utilizan la nomenclatura **sd**xx, ya no utilizándose la **hd**xx. Nótese que las particiones lógicas comienzan su numeración a partir del número 5 (la primera partición lógica del primer disco sería sda5).

A modo de ejemplo, el dispositivo de almacenamiento *sdb6* correspondería a la segunda partición lógica del segundo disco conectado.

Existe un directorio en */dev* llamado */dev/disk* que contiene a su vez una serie de directorios interesantes con enlaces a ficheros en */dev*:

```
$ ls /dev/disk -l
drwxr-xr-x 2 root root 140 mar 28 18:10 by-id
drwxr-xr-x 2 root root  60 mar 28 18:11 by-label
drwxr-xr-x 2 root root 140 mar 28 18:10 by-path
drwxr-xr-x 2 root root 100 mar 28 18:11 by-uuid

$ ls /dev/disk/by-id/ -l
lrwxrwxrwx 1 root root  9 mar 28 18:11 ata-VBOX_CD-ROM_VB2-01700376 -> ../../sr0
lrwxrwxrwx 1 root root  9 mar 28 18:11 ata-VBOX_HARDDISK_VB7eed4173-5bd66d66 -> ../../sda
lrwxrwxrwx 1 root root 10 mar 28 18:11 ata-VBOX_HARDDISK_VB7eed4173-5bd66d66-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 mar 28 18:11 ata-VBOX_HARDDISK_VB7eed4173-5bd66d66-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 mar 28 18:11 ata-VBOX_HARDDISK_VB7eed4173-5bd66d66-part5 -> ../../sda5

$ ls /dev/disk/by-label/ -l
lrwxrwxrwx 1 root root 9 mar 28 18:11 VBOXADDITIONS_5.0.26_108824 -> ../../sr0

$ ls /dev/disk/by-path/ -l
lrwxrwxrwx 1 root root  9 mar 28 18:11 pci-0000:00:01.1-ata-2 -> ../../sr0
lrwxrwxrwx 1 root root  9 mar 28 18:11 pci-0000:00:0d.0-ata-1 -> ../../sda
lrwxrwxrwx 1 root root 10 mar 28 18:11 pci-0000:00:0d.0-ata-1-part1 -> ../../sda1
lrwxrwxrwx 1 root root 10 mar 28 18:11 pci-0000:00:0d.0-ata-1-part2 -> ../../sda2
lrwxrwxrwx 1 root root 10 mar 28 18:11 pci-0000:00:0d.0-ata-1-part5 -> ../../sda5

$ ls /dev/disk/by-uuid/ -l
lrwxrwxrwx 1 root root  9 mar 28 18:11 2016-07-18-12-58-15-00 -> ../../sr0
lrwxrwxrwx 1 root root 10 mar 28 18:11 7dbbb005-92ed-4e04-a082-1028b7e2b92c -> ../../sda1
lrwxrwxrwx 1 root root 10 mar 28 18:11 d22365fa-f0b2-4516-b5a5-17353f1dfe98 -> ../../sda5
```

Podemos ver los correspondientes ficheros de */dev* que corresponden a discos o particiones con nombres distintos según su *id*, *uuid*, *path*, o nombre del dispositivo. En el caso de las particiones GPT esta opción es bastante interesante para poder identificar su *uuid* con el tipo que le corresponde.

Una vez identificados los dispositivos, hay que "montarlos" en un directorio determinado del sistema de ficheros. Este proceso de montaje puede hacerse de manera manual o automática. Actualmente, Linux, al detectar que se inserta un dispositivo USB de almacenamiento extraíble lo monta automáticamente, pero puede suceder que no lo detecte correctamente y lo debamos hacer de manera manual. Del mismo modo sucede con las unidades de CD-ROM.

El montaje manual de dispositivos lo veremos en otro apartado más adelante.

Particionado del disco

Los sistemas Windows no requieren un esquema de particionado determinado, y la única condición necesaria para instalarlos es una partición primaria, marcada como arrancable.

Linux, en cambio, necesita de una partición especial, denominada SWAP o área de intercambio que va a utilizar como espacio donde guardar los procesos que se están ejecutando y que no han de mantenerse en la memoria RAM.

Este mecanismo, también conocido como *memoria virtual* se implementa en Windows mediante un área de intercambio ubicada en un fichero (*C:\pagefile.sys*), en cambio en Linux se utiliza una partición entera.

No hay unas reglas claras para determinar el tamaño de la partición SWAP, pero la relación siguiente suele ser válida para la mayoría de los casos:

- Si la RAM es inferior a 512 MB, el tamaño de la SWAP debería ser el doble del tamaño de la RAM.
- Si se dispone de entre 512 MB y 4 GB, la SWAP debería tener el mismo tamaño que la RAM.

- Si se superan los 4 GB de memoria, con 4 GB de SWAP es suficiente.

En el caso de los sistemas Linux, se requieren un mínimo de dos particiones:

- Sistema de ficheros raíz (/): es el que contiene todo el sistema.
- Sistema de ficheros de intercambio (swap): se necesita para paginar la memoria RAM en el disco duro, cuando la RAM disponible se agota.

Sin embargo, cuando se hace la instalación de un sistema operativo Linux, puede interesar disponer de particiones independientes para otros puntos del sistema como **/home**, **/boot** o **/var**. Por otra parte, directorios como */etc*, */bin* y */sbin*, */lib* y */dev* deben encontrarse en el mismo sistema de archivos que la raíz del sistema.

Es una buena opción montar una partición independiente en */home* para que contenga los archivos de usuario. Así los datos del usuario se conservan durante una actualización del sistema al aislarlos en una partición independiente. También para cifrar de manera independientes particiones. El tamaño de esta partición dependerá del número de usuarios y de sus necesidades de almacenamiento de datos.

Es habitual asignar, durante el proceso de instalación, el directorio **/boot** a una partición exclusiva. Esta partición tiene el gestor de arranque y los distintos kernels con los que podemos arrancar. Su tamaño puede variar, aunque se recomienda entre 150MB y 500MB.

En el caso de los discos GPT, al igual que ocurría con Windows, si el disco es de arranque, es necesario crear una partición EFI para incluir el arranque del sistema. También un valor aproximado de 250MB sería suficiente.

Algunos ejemplos de particionado de discos con Linux serían:

Disco de 100 GB sin particionado extra

sda1 / 98GB	sda5 Swap 2GB

Disco de 100 GB con partición para /boot

sda1	sda2	sda5
/boot	/	Swap
250MB	98GB	2GB

Disco de 100 GB con partición para /boot y /home

sda1	sda5	sda6	sda7
/boot	/	/home	Swap
250MB	50GB	50GB	2GB

Disco de 100 GB con tabla de particiones GPT y arranque EFI sin particionado extra

sda1	sda2	sda3
EFI	/	Swap
250MB	98GB	2GB

Los tres primeros ejemplos correspondían a discos MBR en los que había al menos una partición primaria y en todos ellos la partición de *swap* era una partición lógica.

En el ejemplo del disco GPT son todas particiones primarias y la primera es la partición especial reservada para el EFI.

Manipulación de particiones en el entorno gráfico

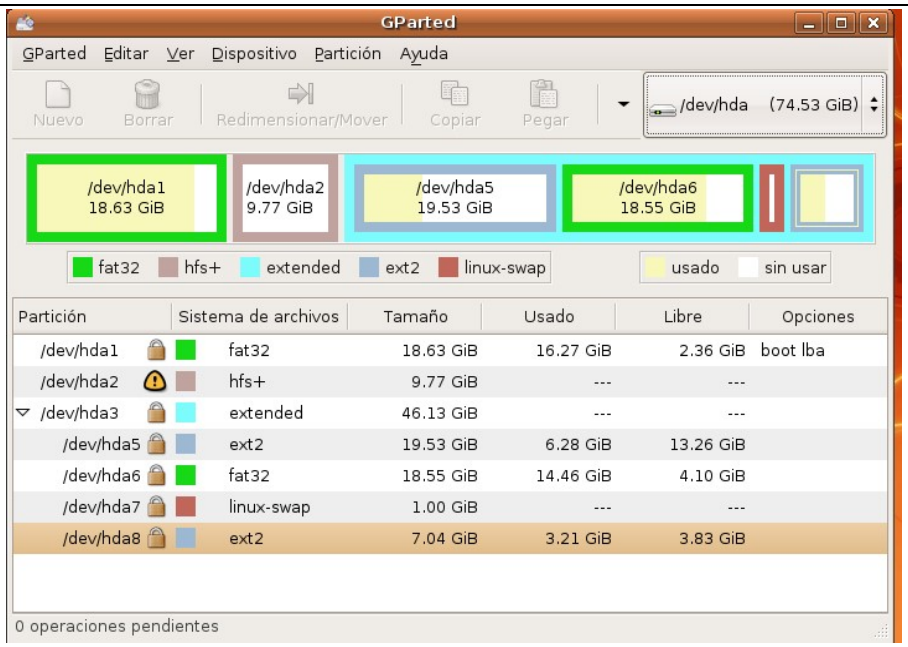
Para la manipulación de discos mediante el entorno gráfico utilizaremos la herramienta **GParted** (GNOME Partition Editor), que se puede instalar en el sistema operativo Ubuntu desde el centro de software de Ubuntu o mediante la siguiente orden:

```
# apt-get install gparted
```

GParted es utilizado en el particionado de disco, cuando se instala Ubuntu, por eso está preinstalado en su CD.

La mayoría de las operaciones de manipulación de disco se deben llevar a cabo cuando estos discos no se encuentran en uso. Por este motivo, la utilización del Live-CD es óptima en este caso. Sin embargo, si se hace uso de la versión instalada de esta utilidad, se requerirá el reinicio del sistema cuando sea necesario.

Tanto con la instalación nativa en Ubuntu como con la del Live-CD, la interfaz de trabajo es idéntica.



Entre otras opciones, GParted permite:

- Crear nuevas tablas de particiones.
- Crear y eliminar particiones.
- Redimensionar y mover particiones.
- Comprobar el estado de las particiones.
- Etiquetar particiones.
- Copiar y pegar particiones.
- Manipular los sistemas de ficheros:
  - btrfs
  - ext2, ext3, ext4
  - fat16, fat32
  - hfs, hfs +
  - Linux-swap
  - ntfs
  - Reiserfs / Reiser4
  - ufs
  - xfs

Manipulación de particiones mediante la línea de mandatos

Los sistemas operativos Linux disponen de herramientas por medio de la línea de comandos que permiten llevar a cabo una manipulación muy completa de los discos. En concreto, las tres herramientas básicas para la manipulación de discos desde la línea de comandos son (entre otras):

- **fdisk**: herramienta que permite manipular la tabla de particiones.
- **gdisk**: herramienta equivalente a la anterior pero para tabla de particiones GPT.
- **mkfs**: herramienta que permite asignar sistemas de ficheros en las particiones (formatear).

Los sistemas Windows también disponen de una utilidad, *diskpart*, que permite hacer la manipulación de particiones, pero tiene un mecanismo de trabajo bastante complejo comparado con las dos herramientas de Linux mencionadas.

La herramienta *fdisk* es proporcionada por el paquete *util-linux*. Esta orden debe ser ejecutada como administrador, aunque se utilice en modo de consulta. No funciona con discos cuya tabla de particiones sea GPT.

Para ejecutar *fdisk* en modo consulta tecleamos:

```
# fdisk –l
```

Y nos muestra por pantalla algo similar a esto:

```
Disc /dev/sda: 8589 MB, 8589934592 bytes
255 heads, 63 sectors/track, 1044 cylinders
Units = cilindres of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x00042d5d
Dispositiv arrenc. Inicio     Final Blocs       Id Sistema
/dev/sda1              1     255   2048256     83 Linux
/dev/sda2            511   1044  4289355     5  Extendida
/dev/sda3            256   510   2048287+    7  HPFS/NTFS
/dev/sda5            511   574   514048+     82Intercambio Linux/Sol
/dev/sda6            575   701   1020096     83 Linux
/dev/sda7            702   828   1020096     b W95 FAT32
/dev/sda8            829   1044  1734988+    83 Linux
```

Si se ejecuta el comando *fdisk* sobre un disco concreto, se entra en el modo interactivo:

```
# fdisk /dev/sdb
```

Donde aparece un menú con todas las opciones que se pueden realizar con el disco (crear particiones, eliminar, redimensionar, …).

```
root@Turing:~# fdisk /dev/sdb
El dispositivo no contiene una tabla de particiones DOS válida ni una etiqueta d
e disco Sun o SGI o OSF
Se está creando una nueva etiqueta de disco DOS con el identificador 0x40eb5587.
Los cambios sólo permanecerán en la memoria, hasta que decida escribirlos.
Tras esa operación, el contenido anterior no se podrá recuperar.

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

Orden (m para obtener ayuda): m
Orden  Acción
   a   Conmuta el indicador de iniciable
   b   Modifica la etiqueta de disco bsd
   c   Conmuta el indicador de compatibilidad con DOS
   d   Suprime una partición
   l   Lista los tipos de particiones conocidos
   m   Imprime este menú
   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)

Orden (m para obtener ayuda):
```

Comenzamos por crear una partición tecleando el comando *n*. Debemos indicarle el tipo de partición (primaria, extendida o lógica), la posición de inicio y el tamaño (o posición final). Pulsando Intro sin más, indicamos el valor predeterminado:

```
Orden (m para obtener ayuda): n
Tipo de partición:
   p primaria (0 primaria, 0 extendida, 4 libre)
   e extendido
Seleccione (predeterminado p): p
Número de partición (1-4, valor predeterminado 1):
Se está utilizando el valor predeterminado 1
Primer sector (2048-104857599, valor predeterminado 2048):
Se está utilizando el valor predeterminado 2048
Último sector, +sectores o +tamaño{K,M,G} (2048-104857599, valor predeterminado
104857599): +30G
```

Repetiríamos el proceso para cada una de las particiones a crear y cuando hemos acabado debemos pulsar el mandato *w* para que se escriban en disco todas las órdenes tecleadas.

Si deseamos particionar un nuevo disco con una tabla de particiones GPT en lugar de *fdisk* debemos utilizar el comando ***gdisk.***

El funcionamiento es similar al *fdisk*. Si lo ejecutamos seguido de un nombre de dispositivo válido entramos en modo interactivo con un menú donde teclear las opciones de particionado de la unidad.

```
# gdisk /dev/sdc
```

```
root@Turing:~# gdisk /dev/sdc
GPT fdisk (gdisk) version 0.8.8

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries.

Command (? for help): ?
b       back up GPT data to a file
c       change a partition's name
d       delete a partition
i       show detailed information on a partition
l       list known partition types
n       add a new partition
o       create a new empty GUID partition table (GPT)
p       print the partition table
q       quit without saving changes
r       recovery and transformation options (experts only)
s       sort partitions
t       change a partition's type code
v       verify disk
w       write table to disk and exit
x       extra functionality (experts only)
?       print this menu
```

Con el mandato *n* creamos una nueva partición. Si en el disco no hay ninguna tabla de particiones, la creará como GPT. Podemos crear la tabla de particiones con el mandato *o*. Al crear la partición con el mandato *n* hay que indicar el sector de comienzo y/o final, o bien podemos optar por indicar directamente el tamaño de la partición. Es importante indicar el tipo de partición que estamos haciendo (recordemos que las particiones GPT necesitan de un GUID[1] que identifica el tipo de partición). Para ver el código de los distintos tipos de partición pulsamos *L*.

[1] Ver tema de instalación de sistemas operativos, apartado de Particionado de Discos, discos GPT.

```
Command (? for help): n
Partition number (1-128, default 1):
First sector (34-104857566, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-104857566, default = 104857566) or {+-}size{KMGTP}: +30G
Current type is 'Linux filesystem'
Hex code or GUID (L to show codes, Enter = 8300): L
0700 Microsoft basic data  0c01 Microsoft reserved    2700 Windows RE
4100 PowerPC PReP boot     4200 Windows LDM data      4201 Windows LDM metadata
7501 IBM GPFS              7f00 ChromeOS kernel       7f01 ChromeOS root
7f02 ChromeOS reserved     8200 Linux swap            8300 Linux filesystem
8301 Linux reserved        8302 Linux /home           8400 Intel Rapid Start
8e00 Linux LVM             a500 FreeBSD disklabel     a501 FreeBSD boot
a502 FreeBSD swap          a503 FreeBSD UFS           a504 FreeBSD ZFS
a505 FreeBSD Vinum/RAID    a580 Midnight BSD data     a581 Midnight BSD boot
a582 Midnight BSD swap     a583 Midnight BSD UFS      a584 Midnight BSD ZFS
a585 Midnight BSD Vinum    a800 Apple UFS             a901 NetBSD swap
a902 NetBSD FFS            a903 NetBSD LFS            a904 NetBSD concatenated
a905 NetBSD encrypted      a906 NetBSD RAID           ab00 Apple boot
af00 Apple HFS/HFS+        af01 Apple RAID            af02 Apple RAID offline
af03 Apple label           af04 AppleTV recovery      af05 Apple Core Storage
be00 Solaris boot          bf00 Solaris root          bf01 Solaris /usr & Mac Z
bf02 Solaris swap          bf03 Solaris backup        bf04 Solaris /var
bf05 Solaris /home         bf06 Solaris alternate se  bf07 Solaris Reserved 1
bf08 Solaris Reserved 2    bf09 Solaris Reserved 3    bf0a Solaris Reserved 4
bf0b Solaris Reserved 5    c001 HP-UX data            c002 HP-UX service
ea00 Freedesktop $BOOT     eb00 Haiku BFS             ed00 Sony system partitio
ef00 EFI System            ef01 MBR partition scheme  ef02 BIOS boot partition
fb00 VMWare VMFS           fb01 VMWare reserved       fc00 VMWare kcore crash p
fd00 Linux RAID
Hex code or GUID (L to show codes, Enter = 8300): 8300
Changed type of partition to 'Linux filesystem'
```

Al igual que con *fdisk*, una vez creadas la partición o particiones debemos pulsar el comando *w* para que los cambios se escriban en disco y surtan efecto.

```
Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/sdc.
The operation has completed successfully.
```

Formateo de particiones

Una vez se dispone de la distribución del disco que se quiere, la herramienta **mkfs** permite asignar sistemas de ficheros en estas particiones, es decir, **formatear la partición** y prepararla para un determinado sistema de ficheros. Tiene la siguiente sintaxis:

```
# mkfs.tipo [opciones] sistema_de_ficheros
```

Donde *sistema_de_ficheros* es el nombre del dispositivo (*/dev/sdb1*), *tipo* es el tipo de sistema de archivos que se desea crear. Las opciones específicas para cada sistema de archivos se pueden encontrar en la página de manual de cada uno de estos.

Los tipos de sistemas de ficheros soportados por este programa son, entre otros, los siguientes:

- ext2, ext3, ext4.
- NTFS.
- FAT32 (vfat, con la opción-F 32, para indicar que se quiere FAT 32).
- btrfs
- BFS, cramfs, minix.

Por ejemplo, la siguiente instrucción:

```
# mkfs.ext4 -L etiqueta /dev/sdb1
```

Formatearía la primera partición del segundo disco (sdb1) con un formato para el tipo de fichero ext4 y le asignaría la etiqueta *etiqueta*.

No hay una opción de *mkfs* para crear particiones de intercambio. Este tipo de particiones se formatea con la herramienta **mkswap**:

```
# mkswap /dev/sdb5
```

4. Montaje de dispositivos

Una vez identificados y formateados los dispositivos y sistemas de archivos, hay que "montarlos" en un directorio determinado del sistema de ficheros. Este proceso de montaje puede hacerse de manera manual o automática. Actualmente, Linux, al detectar que se inserta un dispositivo USB de almacenamiento extraíble lo monta automáticamente, pero puede suceder que no lo detecte correctamente y lo debamos hacer de manera manual. Del mismo modo sucede con las unidades de CD-ROM.

Para ver en qué directorio nos ha montado los distintos dispositivos y sistemas de archivos, tecleamos el mandato **mount**.

```
# mount
```

```
/dev/sda1 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
none on /sys type sysfs (rw,noexec,nosuid,nodev)
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)
none on /dev type devtmpfs (rw,mode=0755)
none on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
none on /dev/shm type tmpfs (rw,nosuid,nodev)
none on /var/run type tmpfs (rw,nosuid,mode=0755)
none on /var/lock type tmpfs (rw,noexec,nosuid,nodev)
none on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc
      (rw,noexec,nosuid,nodev)
gvfs-fuse-daemon on /home/pepe/.gvfs type fuse.gvfs-fuse-daemon
      (rw,nosuid,nodev,user=pepe)
/dev/sr0 on /media/CDROM type
      iso9660(ro,nosuid,nodev,uhelper=udisks,uid=1000,gid=1000,
      iocharset=utf8,mode=0400,dmode=0500)
```

La primera línea nos indica que la primera partición primaria del disco duro SATA principal está montada en la raíz (/) del sistema de ficheros y es de tipo ext4. Tenemos permisos de lectura/escritura (rw).

Del mismo modo la última nos está indicando que la unidad de CD-ROM (*/dev/sr0*) está montada en el directorio */media/CDROM*.

Estas unidades están montadas de manera automática. Si quisiéramos hacerlo de manera manual utilizaríamos el mandato *mount* con la siguiente sintaxis:

```
$ sudo mount -t <sistema_archivos> [-o <opciones>] /dev/<particion>
<carpeta_montaje>
```

Cabe aclarar que si ocurre algún error durante el montaje, no se pondrán en peligro los datos de la partición, solo que esta no será montada.

El significado de los parámetros usados en el comando *mount* son los siguientes:

- <sistema_archivos>: es el sistema de archivos de la partición; puede ser ext3, ext4, vfat (FAT16 y FAT32), ntfs (NTFS), ufs (UFS y UFS2), o iso9660 entre muchos otros.
- <opciones>: son las opciones de montaje, puede tomar más de un valor, en ese caso los valores se separan con comas (,). Algunos posibles valores son *defaults* (valores por defecto), *ro* (Read Only, es decir, Solo Lectura) y *ufstype* (para especificar el tipo de sistema de archivos UFS, en caso de que se use este); si no se especifican opciones especiales, podemos escribir *defaults*, u obviar este parámetro por completo (quitando también el -o de adelante).
- <particion>: es el identificador de la partición que vamos a montar; puede ser *hdXY* en caso de ser un disco IDE o ATA, o *sdX,Y* en caso de ser SATA; Para saber el nombre de la partición tecleamos el comando **fdisk -l**
- <carpeta_montaje>: es la carpeta donde se montará la partición, es decir, donde aparecerán los datos (archivos y carpetas) de la partición; en la mayoría de los casos se encuentra dentro de */media/*, aunque puede estar en cualquier otro lugar.

Ejemplo:
Para montar un dispositivo USB de almacenamiento extraíble.

1. Insertamos el dispositivo USB.
2. Si no lo monta de manera automática:
   - Creamos la carpeta donde lo vamos a montar:
     ```
     $ sudo mkdir /media/USB
     ```
   - Visualizamos el nombre del dispositivo que le ha dado Linux:
     ```
     $ sudo fdisk -l
     ```
   - Montamos el dispositivo utilizando el nombre que le ha dado y hemos visualizado en el punto anterior.
     ```
     $ sudo mount -t vfat /dev/sdb1 /media/USB
     ```

Adicionalmente, una vez montado el dispositivo, podríamos crear un enlace simbólico en nuestra carpeta $HOME o bien un acceso directo en el escritorio.

Para desmontar el dispositivo teclearíamos:

```
$ sudo umount /media/USB
```

Otro ejemplo interesante sería montar una carpeta compartida de Virtualbox en un equipo invitado con sistema operativo Linux. En primer lugar habría que configurar Virtualbox en el equipo anfitrión para compartir carpetas, en este punto es importante el nombre que le demos (por ejemplo Compartida). Una vez hecho, en el sistema invitado ejecutaríamos:

```
$ sudo mkdir /media/compartida
$ sudo mount –t vboxsf Compartida /media/compartida
```

Donde el nombre Compartida debe coincidir (con mayúsculas incluidas) con el nombre que le hemos dado en Virtualbox. El nombre de la carpeta para el punto de montaje es indiferente.

Para desmontar:

```
$ sudo umount Compartida
```

Una vez que hayamos conseguido montar una partición, esta quedará montada mientras el sistema esté en marcha. Cuando reiniciemos o apaguemos el equipo, tendremos que volver a montar la partición ya que no se montará al iniciar Ubuntu. Si queremos que se monte cada vez que iniciamos el sistema, necesitaremos modificar el archivo que contiene la tabla de sistemas de archivos (**/etc/fstab**).

```
$ sudo nano /etc/fstab
```

Una vez abierto el archivo, tenemos que cambiar la línea que comience con el identificador de la partición que hemos montado *(/dev/hdXY* o */dev/sdXY*) por la siguiente:

/dev/<particion> /media/<carpeta_montaje> <sistema_archivos> <opciones> 0 0

Si no existe esa línea, la añadimos al final del archivo.

Los argumentos son los mismos que cuando usamos el comando *mount*. Aquí, si en opciones no usamos ningún valor, tendremos que escribir *defaults*, y nos quedaría algo así:

/dev/<particion> /media/<carpeta_montaje> <sistema_archivos> defaults 0 0

Si es una partición FAT16 o FAT32 y no nos funciona con defaults, podemos probar las siguientes opciones:

```
auto,users,exec,umask=000
defaults,rw,user,auto,umask=0
```

De esta última manera se están dando permisos de lectura, escritura y ejecución a todos los usuarios. Si queremos restringir estos permisos solo a un grupo particular de usuarios (por ejemplo: users), las opciones deben quedar así:

```
defaults,rw,user,auto,umask=007,gid=<grupo> 0 0
```

Donde <grupo> debe sustituirse por el grupo de usuarios, por ejemplo, *users*.

Finalmente, si por cualquier motivo no se detectan bien algunos caracteres (como la letra ñ), debemos añadir la siguiente opción junto con las otras utilizadas, para cambiar el mapa de caracteres:

```
iocharset=utf8
```

Por ejemplo, una línea podría quedar así:

```
/dev/hda0 /media/hda0 vfat defaults,rw,user,auto,iocharset=utf8
      ,umask=000 0 0
```

Para montar todos los dispositivos listados en el archivo */etc/fstab* sin tener que reinicia, tenemos que ejecutar el siguiente comando en una terminal:

```
$ sudo mount -a
```

Con esto ya tendemos montada nuestra partición cada vez que se inicie Ubuntu.

Normalmente sólo *root* puede ejecutar el comando *mount* (de ahí el uso de *sudo* en los ejemplos anteriormente escritos); no obstante si */etc/fstab* especifica las opciones *user*, *users* u *owner*, un usuario normal podría montar un sistema de archivos especificando una sintaxis simplificada en la que sólo se indique el dispositivo o el punto de montaje, pero no ambos.

5. RAID

Conocemos de temas anteriores el concepto de RAID y su implementación mediante software en el sistema operativo Windows Server. Vamos a ver ahora cómo implementarlo en Linux de una manera sencilla sin entrar en mucho detalle.

Vamos a utilizar el paquete MDADM (***M**ultiple **D**evice **Adm**inistrator*), que es un conjunto de herramientas que son utilizadas en GNU/Linux para la gestión del RAID. MDADM viene instalado en algunas distribuciones Linux por defecto (en las Servers normalmente) pero no en la distibución Ubuntu Desktop.

Para instalar el paquete ejecutamos:

```
$ sudo apt-get install mdadm
```

Una vez instalado podemos crear los siguientes tipos de RAID: RAID 0, RAID 1, RAID 4, RAID 5, RAID 6 y RAID 10.

Las unidades que formen parte de un mismo RAID deben ser idénticas en capacidad. Supongamos que vamos a crear un RAID-1 con dos discos definidos como dispositivos */dev/sdb* y */dev/sdc*.

Con el siguiente comando creamos un RAID-1 (*/dev/md0*), indicando el número de dispositivos que lo integrarán (*raid-devices*=2) y las unidades de almacenamiento que lo integrarán (*/dev/sdb* y */dev/sdc*).

```
# mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sdb
      /dev/sdc
```

Nos aparecerá una advertencia informando de la no conveniencia de utilizar RAID en la partición que alberga */boot* porque muchos gestores de arranque carecen de soporte para metadatos versión 1.x y que se recomienda usar *--metadata=0.90*. Al usar nuestro RAID para datos vamos a pasar por alto esta advertencia pulsando *y*.

El RAID comienza a construirse. Se puede ver el estado de progreso ejecutando lo siguiente:

```
# mdadm --detail /dev/md0
```

Cuando haya finalizado, en el estado (*state*) aparecerá *active*.

Puede verificarse un resumen del estado a través del fichero */proc/mdstat*.

```
# cat /proc/mdstat
```

Si todo es correcto, ya se puede actualizar el fichero de configuración de *mdadm* que se encuentra en */etc/mdam/mdadm.conf* y armar el RAID.

```
# mdadm --detail --scan >> /etc/mdadm/mdadm.conf
# mdadm --assemble --scan
```

A partir de este momento podemos utilizar el dispositivo */dev/md0* como cualquier otra unidad de almacenamiento (crear particiones, formatearlas, montarlas y guardar datos). Finalmente, para sustituir un disco del RAID que ha fallado y ha sido eliminado, conectaríamos un nuevo disco al sistema y lo añadiríamos al RAID mediante el comando:

```
# mdadm /dev/md0 --add /dev/sdd
```

Indicando el nuevo dispositivo (en el ejemplo */dev/sdd*).

6. Volúmenes Lógicos

Volviendo a la analogía entre Windows y Linux, cuando estudiamos en Windows Server los volúmenes dinámicos vimos la potencia y la flexibilidad de éstos respecto a los volúmenes básicos tradicionales (las particiones de disco).

En Linux también disponemos de mecanismos de abstracción para crear volúmenes lógicos mucho más potentes y flexibles que las particiones tradicionales. Vamos a estudiar el LVM (*Gestor de volúmenes lógicos, Logical Volume Manager*), una potente herramienta para gestionar los discos físicos mediante volúmenes lógicos.

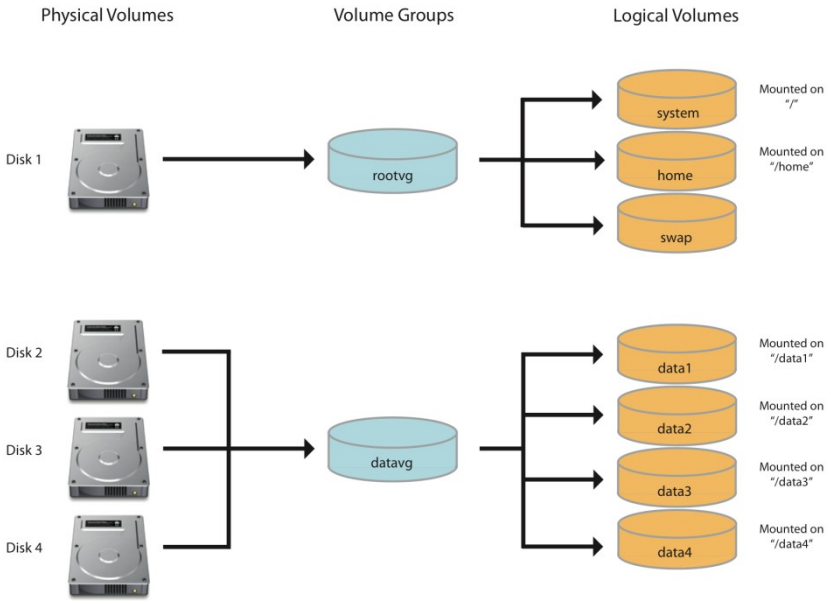
Las ventajas que tienen son múltiples, pero la inicial y más evidente es la flexibilidad frente al particionado tradicional. Si tuviéramos, por ejemplo, 4 particiones (sin LVM) contiguas en un disco y en el futuro quisiéramos aumentar alguna de las 3 primeras, no

podríamos hacerlo sin borrar las siguientes, lo que es complejo, peligroso y requiere de parada del sistema. Si quisiéramos ampliar la última partición, siempre tendríamos el límite del tamaño del disco. O si compráramos un disco nuevo, y quisiéramos ampliar el espacio de un sistema de ficheros existente en el disco anterior con el espacio nuevo, sería imposible salvo con apaños de nuevos sistemas de ficheros y puntos de montaje. Con *LVM* todas esas limitaciones desaparecen.

Con LVM podemos aumentar sus "particiones" (volúmenes lógicos) independientemente de que no haya espacio libre contiguo a éstas, podemos aumentar sus volúmenes lógicos con espacio libre de diferentes discos físicos, e incluso podemos mover volúmenes lógicos entre dispositivos físicos. Y se puede hacer en caliente, sin desmontar el sistema de ficheros y sin parar el sistema.

En primer lugar vamos a definir los conceptos necesarios para trabajar con LVM:

- **Volumen físico (**Physical Volume, PV). Un volumen físico (PV en adelante) es un dispositivo de almacenamiento. Puede ser un disco duro (/dev/sda), una partición (/dev/sdb2), una tarjeta SD, o incluso un dispositivo RAID (/dev/md0). Para simplificar diremos que un PV es una fuente de almacenamiento, es decir un dispositivo que nos proporciona espacio.
- **Grupo de volúmenes** (Volume Group, VG). Para poder usar el almacenamiento de un PV, éste debe pertenecer a un Grupo de volúmenes (en adelante VG). Un VG es un "disco" compuesto de uno o más PVs y que crece simplemente añadiendo más PVs. A diferencia de un disco real, un VG puede crecer con el tiempo, sólo hay que "darle" un PV más. En una máquina con un sólo disco podemos crear un VG que esté compuesto por un sólo PV (el disco físico o una de sus particiones). Si con el tiempo nos quedamos sin espacio en el VG, podemos comprar otro disco (PV), lo añadimos al VG y el resto es transparente para sistemas de ficheros, procesos o usuarios.
- **Volumen Lógico** (Logical Volume, LV). Los volúmenes lógicos son "el producto final" del LVM. Son estos dispositivos los que usaremos para crear sistemas de ficheros, swap, discos para máquinas virtuales, etc… Por seguir con la analogía del "disco" que es el VG, los LVs serían las particiones del VG. A diferencia de las particiones tradicionales, los LVs pueden crecer (mientras haya espacio en el VG) independientemente de la posición en la que estén.



Para instalar LVM en el sistema tecleamos:

```
# apt-get install lvm2
```

Creación de volúmenes lógicos

El primer paso para comenzar a trabajar es crear los PVs. Un PV puede ser una unidad entera, una partición un RAID o cualquier otra unidad de almacenamiento. Para crear un volumen físico (PV) basta con poner el mandato *pvcreate* y el identificador del dispositivo de almacenamiento:

```
# pvcreate /dev/sdd
# pvcreate /dev/sde
```

Con *pvs* visualizamos los volúmenes físicos creados.

```
root@Kepler:~# pvcreate /dev/sdd
  Physical volume "/dev/sdd" successfully created
root@Kepler:~# pvcreate /dev/sde
  Physical volume "/dev/sde" successfully created
root@Kepler:~# pvs
  PV         VG   Fmt  Attr PSize  PFree
  /dev/sdd        lvm2 a--  50,00g 50,00g
  /dev/sde        lvm2 a--  50,00g 50,00g
```

Una vez creados los PVs hay que agruparlos en uno o varios grupos de volúmenes (GV). Para ello utilizamos el mandato *vgcreate*, asignándole un nombre y enumerando los PVs que lo componen.

```
# vgcreate produccion /dev/sdd /dev/sde
```

Con *vgs* visualizamos los grupos de almacenamiento creados.

```
root@Kepler:~# vgcreate produccion /dev/sdd /dev/sde
  Volume group "produccion" successfully created
root@Kepler:~# vgs
  VG         #PV #LV #SN Attr   VSize  VFree
  produccion   2   0   0 wz--n- 99,99g 99,99g
```

Con *vgdisplay* visualizamos la información de los grupos de almacenamiento.

```
root@Kepler:~# vgdisplay
  --- Volume group ---
  VG Name               produccion
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  1
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                0
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               99,99 GiB
  PE Size               4,00 MiB
  Total PE              25598
  Alloc PE / Size       0 / 0
  Free  PE / Size       25598 / 99,99 GiB
  VG UUID               ybjr18-1Fd4-1wxq-6QVg-Qqtr-iqiC-XEQ4zO
```

De esta información es interesante observar que nos muestra el tamaño del VG en GiB (en nuestro caso 100GiB) y también en una unidad llamada PE (Physical Extension) que son unidades mínimas de asignación del LVM, cada una de ella de 4 MiB. En el ejemplo tenemos 25.598 PE libres que equivalen a aproximadamente 100 GiB.

Una vez tenemos un VG ya podemos crear los dispositivos que realmente usaremos. Los volúmenes lógicos (LV) pertenecen a un VG, del que toman su espacio. Pueden crearse, borrarse y crecer sin necesidad de reiniciar la máquina o parar servicios.

Un LV puede usar espacio de un solo PV, o de varios. En este último caso puede deberse a que vaya creciendo con el tiempo, y ya no quede espacio en el PV original, o que al crearlo hemos decidido que use varios PVs (por motivos de rendimiento, por ejemplo).

Para crear un LV, con el comando *lvcreate*, debemos indicarle el VG al que pertenece (como argumento del comando), el tamaño (en PEs con la opción -l o en

megas/gigas/teras con -L) y opcionalmente, pero muy recomendable, el nombre que queremos darle (-n).

```
# Un LV de 400MB en el VG produccion para guardar documentos

lvcreate -L 400M -n documentos produccion


# O especificando el número de PEs (de 4MB por defecto)

lvcreate -l 100 -n documentos produccion
```

Una vez creado, podemos visualizar la información, tal y como hacíamos con los PV y VG, con los comandos *lvs* y *lvdisplay*.

```
root@Kepler:~# lvs
  LV         VG         Attr      LSize   Pool Origin Data%  Move Log Copy%  Con
vert
  documentos produccion -wi-a---- 400,00m
```

```
root@Kepler:~# lvdisplay
  --- Logical volume ---
  LV Path                /dev/produccion/documentos
  LV Name                documentos
  VG Name                produccion
  LV UUID                FS4hEf-Cwzv-joOd-ytHN-Wav9-4jcN-hZk8iB
  LV Write Access        read/write
  LV Creation host, time Kepler, 2016-04-28 09:09:06 +0200
  LV Status              available
  # open                 0
  LV Size                400,00 MiB
  Current LE             100
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0
```

Nótese que se nos ha creado una descripción de dispositivo en */dev* denominada */dev/produccion/documentos* aunque realmente este fichero es un enlace a */dev/dm-0*

Si ejecutamos el mandato *fdisk –l* veremos cómo nos proporciona otro nombre:

```
# fdisk -l
Disco /dev/mapper/produccion-documentos: 419 MB, 419430400 bytes
255 cabezas, 63 sectores/pista, 50 cilindros, 819200 sectores en total
Unidades = sectores de 1 * 512 = 512 bytes
Tamaño de sector (lógico / físico): 512 bytes / 512 bytes
Tamaño E/S (mínimo/óptimo): 512 bytes / 512 bytes
Identificador del disco: 0x00000000

El disco /dev/mapper/produccion-documentos no contiene una tabla de
particiones válida
```

El dispoditivo */dev/mapper/produccion-documentos* también es un enlace al fichero */dev/dm-0*

Como norma general, para hacer referencia a un dispositivo VL siempre utilizaremos la nomenclatura */dev/VG/LV* (*/dev/produccion/documentos* en el ejemplo).

Una vez creado el LV sólo faltaría formatear el sistema de ficheros y montarlo con los métodos ya estudiados.

```
# mkfs.ext4 /dev/produccion/documentos
# mount /dev/produccion/documentos /media/
```

Ampliación de volúmenes lógicos

Una de las principales características de trabajar con volúmenes lógicos es la flexibilidad. Supongamos que una vez creado y puesto en funcionamiento un volumen lógico nos quedamos sin espacio en él. A diferencia del método tradicional de particiones, podemos ampliar el LV simplemente cogiendo espacio disponible del Grupo de Almacenamiento (VG) al que pertenece o añadiendo un nuevo disco o partición disponible como PV al Grupo de Almacenamiento y ampliando el tamaño del Volumen Lógico final.

Veamos dos ejemplos.

En primer lugar vamos a ampliar el LV cogiendo espacio libre del VG creado.
Nos aseguramos de que hay espacio libre en el VG del que queremos coger mediante el mandato *vgdisplay*.

Una vez cerciorados de la disponibilidad, ampliamos el LV con el manadto *lvextend* especificando el nombre del LV (*/dev/NOMBRE_VG/NOMBRE_LV*) como argumento y usaremos la opción *-L* para indicar el tamaño nuevo. Como argumento de la opción *-L* podemos usar el tamaño final (por ejemplo **-L2G**, para un tamaño final de 2G) o la cantidad que deseamos ampliar, poniendo un "+" delante de ésta (por ejemplo **-L+1G**, para ampliar un giga).

```
# lvextend -L+1G /dev/produccion/documentos
```

Comprobamos el nuevo tamaño del LV (los 400 MB anteriores y 1 GB de ahora).

```
root@Kepler:~# lvdisplay
  --- Logical volume ---
  LV Path                /dev/produccion/documentos
  LV Name                documentos
  VG Name                produccion
  LV UUID                FS4hEf-Cwzv-joOd-ytHN-Wav9-4jcN-hZk8iB
  LV Write Access        read/write
  LV Creation host, time Kepler, 2016-04-28 09:09:06 +0200
  LV Status              available
  # open                 0
  LV Size                1,39 GiB
  Current LE             356
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           252:0
```

Podríamos haber utilizado también el mandato *resize* para ampliar el LV. Este mandato nos permite también reducir el tamaño del volumen lógico.

OJO. Aquí no acaba la tarea. Una vez ampliado el LV debemos ampliar el sistema de ficheros (si es que estamos usando el LV para un sistema de ficheros). Si nos fijamos, aunque hemos ampliado el LV, el espacio disponible en el sistema de ficheros siguen siendo los 400MB originales. Lo comprobamos con el mandato *df* sobre el dispositivo:

```
root@Kepler:~# df -h /dev/produccion/documentos
S.ficheros                      Tamaño Usados  Disp Uso% Montado en
/dev/mapper/produccion-documentos   380M   2,3M  354M   1% /media
```

Para ello utilizamos el mandato *resize2fs* especificando el dispositivo.

```
# resize2fs /dev/produccion/documentos
```

Ahora ya tenemos el tamaño del sistema de ficheros ampliado.

Veamos un segundo ejemplo. Supongamos que ya no queda espacio en el VG desde el cuál ampliar nuestro volumen lógico, por tanto es necesario incluir un nuevo PV. Vamos a insertar un nuevo disco en nuestra máquina. Seguidamente lo convertimos en PV, lo añadimos al VG y por último extendemos el LV.

El identificador de dispositivo de nuestro nuevo disco es */dev/sdf*. Lo convertimos en volumen físico:

```
# pvcreate /dev/sdf
```

Y seguidamente lo añadimos al grupo de almacenamiento con el mandato *vgextend* indicando el grupo de almacenamiento y el nuevo PV:

```
# vgextend produccion /dev/sdf
```

Una vez ampliado el grupo de almacenamiento, sólo nos queda ampliar el volumen lógico del mismo modo que hemos visto antes y el sistema de ficheros:

```
# lvextend -L+30G /dev/produccion/documentos
# resize2fs /dev/produccion/documentos
```

7. Iniciador iSCSI

Al igual que los puntos anteriores, cuando estudiamos la gestión del almacenamiento en Windows Server vimos cómo conectar un dispositivo SAN mediante tecnología iSCSI. Vimos que necesitábamos un Target iSCSI (en el lado del servidor de almacenamiento) y un Iniciador iSCSI en el lado del cliente.

Sin entrar en detalle de cómo se instala y configura un Target iSCSI, vamos a ver cómo conectar una máquina Linux a ese dispositivo de almacenamiento mediante un Iniciador iSCSI.

Lo primero que vamos a hacer es instalar el software iniciador. En Linux contamos con varios iniciadores iSCSI, sin embargo el más usado es **open-iscsi**, que además está dentro de la paquetería de Debian. Lo instalamos:

```
# apt-get install open-iscsi
```

Esto nos instalará un script de inicio que carga los módulos y lanza un programa que controla el iniciador (*iscsid*), y un programa para configurar nuestro iniciador (*iscsiadm*). Los módulos para iSCSI ya están presente en los kernels actules (desde el 2.6.18 en adelante) por eso no se incluyen en el paquete de *open-iscsi*.

Lo primero que debemos hacer es iniciar el servicio si no lo estuviera:

```
# /etc/init.d/open-iscsi start
```

La configuración de open-iscsi se realiza a través del comando *iscsiadm* y la misma se guarda en una base de datos. Si queremos cambiar algún parámetro de la configuración tenemos que hacerlo a través de *iscsiadm*.

En primer lugar tenemos que indicarle a *iscsiadm* que detecte nuestro target iSCSI y lo agregue a su base de datos. Hay que aclarar que *iscsiadm* tiene tres formas de operación

- **discovery**: En este modo se pueden descubrir targets y agregarlos a la base de datos.
- **node**: En este modo se administran los targets ya descubiertos y se pueden visualizar datos acerca de estos nodos, así como conectarse a ellos.
- **session**: En este modo se administran los targets a los que se está conectados (en los que se ha hecho login).

Para descubrir nuestro target usamos el modo *discovery* indicando la dirección IP de nuestro Target:

```
# iscsiadm -m discovery -t sendtargets -p 192.168.1.200
```

Con lo que le indicamos a *iscsiadm* que descubra los targets que le ofrece el portal ubicado en 192.168.1.200. El método que usa para descubrirlo es "sendtargets", que es el método soportado más estable a la fecha (existen otros métodos, como por ejemplo uno que resuelve nombre de targets).

Como vemos, descubrió el target que tendríamos instalado anteriormente.

```
root@Turing:~# iscsiadm -m discovery -t sendtargets -p 192.168.1.200
192.168.1.200:3260,-1 iqn.2005-10.org.freenas.ctl:iscsi01
```

Nos ha devuelto un *targetname* encontrado:

```
iqn.2005-10.org.freenas.ctl:iscsi01
```

Por lo cual, si hacemos solo:

```
# iscsiadm -m discovery
192.168.1.200:3260 via sendtargets
```

Vemos que la información del servidor iSCSI target descubierto quedó almacenada en la base de datos de *iscsiadm*.

Una vez descubierto y conocido el nombre del target iSCSI hay que conectarse mediante un *login* especificando en el mandato *iscsiadm* el modo *node* (para configurar), el nombre del target, la dirección IP y el parámetro –l para indicar que queremos loguearnos:

```
# iscsiadm -m node --targetname iqn.2005-10.org.freenas.ctl:iscsi01 -p
192.168.1.200 -l
```

```
root@Turing:~# iscsiadm -m node --targetname iqn.2005-10.org.freenas.ctl:iscsi01
 -p 192.168.1.200 -l
Logging in to [iface: default, target: iqn.2005-10.org.freenas.ctl:iscsi01, port
al: 192.168.1.200,3260] (multiple)
Login to [iface: default, target: iqn.2005-10.org.freenas.ctl:iscsi01, portal: 1
92.168.1.200,3260] successful.
```

Ya está el disco listo para ser utilizado. Podemos comprobarlo ejecutando el mandato *fdisk –l*:

```
# fdisk -l
```

En nuestro caso nos ha creado un dispositivo en */dev/sdb* con el que ya podemos trabajar con él creando su tabla de particiones y formateando la partición en caso de que fuera necesario, o directamente montándolo y trabajando con sus datos si ya estuviera listo.

Como último punto relacionado con la configuración del iniciador, deberíamos definir que se haga *login* al target y se conecte el disco automáticamente cada vez que se inicie el servicio *open-iscsi*, de manera que no tengamos que hacerlo a mano luego de que nuestro sistema inicie. Volvemos a utilizar *iscsiadm* para modificar el parámetro que maneja esta opción:

```
# iscsiadm -m node --targetname iqn.2005-10.org.freenas.ctl:iscsi01 -p
192.168.1.200 -o update -n node.conn[0].startup -v automatic
```

Luego de esto, si hacemos

```
# /etc/init.d/open-iscsi restart
```

El disco iSCSI debería ser detectado automáticamente y conectado al equipo, incluso después de reiniciar el equipo.

8. NFS

NFS son las siglas en inglés de Network File System que podríamos traducir como Sistema de Archivos en Red. Es el sistema que utiliza Linux para compartir carpetas en una red. Un servidor puede compartir sus carpetas y desde los PCs de los usuarios se puede acceder a dichas carpetas compartidas y el resultado es el mismo que si estuvieran en su propio disco duro.

Básicamente **NFS** permite, a ordenadores que utilizan Linux, compartir y conectarse a carpetas compartidas entre sí. Es el sistema nativo que utiliza Linux para compartir y acceder a carpetas compartidas en la red.

Existen otras alternativas para compartir carpetas en una red como **samba, ssh o ftp,** pero el sistema recomendado para compartir carpetas entre sistemas Linux es NFS.

NFS utiliza un modelo cliente/servidor, de manera que el ordenador que va a compartir las carpetas y directorios será el servidor, y el/los ordenadores que van acceder a esas carpetas compartidas serán los clientes. No hay problema en que un mismo ordenador haga las veces de cliente de otro equipo y servidor de sus propias carpetas.

Es por ello que si el ordenador va a compartir carpetas, es necesario instalar el paquete de NFS que actúa de servidor. Para instalar el servidor NFS ejecutamos:

```
# sudo apt-get install nfs-kernel-server
```

Una vez instalado el servidor NFS y antes de arrancar el servicio NFS, es necesario indicar qué carpetas deseamos compartir y si queremos que los usuarios accedan con permisos de sólo lectura o de lectura y escritura. También existe la posibilidad de establecer desde qué equipos es posible conectarse. Estas opciones se configuran en el archivo */etc/exports*

En cada línea del archivo de configuración del servidor NFS */etc/exports*, se puede especificar:

- La carpeta que se quiere compartir
- Desde qué equipos se permite el acceso (nombre o IP del equipo o rango de IPs)

- Las opciones de acceso: diferentes opciones que asignamos a este directorio para ese equipo en concreto y que determinarán los privilegios de acceso a él. De todas la opciones disponibles, las más significativas son:
  - ro|rw: el directorio será compartido en solo lectura (*ro*) y es la opción por defecto. El directorio será compartido en lectura y escritura (*rw*).
  - sync|async: *sync* comunica al usuario los cambios realizados sobre los archivos cuando realmente se han ejecutado y es la opción recomendada. La opción *async* mejora el rendimiento y agiliza el funcionamiento del servicio, pero puede generar archivos corruptos si se produce algún tipo de fallo en el servidor.
  - no_subtree_check*:* permite que no se compruebe el camino hasta el directorio que se exporta, en el caso de que el usuario no tenga permisos sobre el directorio exportado.
  - root_squash | no_root_squash | all_squash
    - root_squash indica que un usuario identificado como *root* tendrá acceso al directorio compartido sólo con privilegios de usuario anónimo. De esta forma se ha degradado al root al usuario local de privilegios más bajos protegiendo así los archivos en el servidor NFS. Esta opción se conoce también con el nombre de 'aplastamiento del root'. Para el resto de usuarios se intenta conservar su UID y GID en el servidor.
    - no_root_squash desactiva la opción anterior, es decir, los accesos realizados como root desde el cliente serán también de root en el servidor NFS.
    - all_squash indica que todos los clientes, incluido root, tendrán acceso al directorio con privilegios de un usuario anónimo. No se mantienen los UID y GID de ningún usuario.
    - Si se utiliza alguna de las opciones squash podemos indicar cuál es el UID y GID del usuario con el que se quiere que se acceda, en lugar del anónimo. En este caso hemos de indicar a continuación de la opción squash lo siguiente:

      (rw,all_squash,anonuid=1002,anongid=1002)

Y significa que la conexión del cliente NFS se hará con los UID y GID 1002.

```
// Ejemplo de archivo /etc/exports de configuración del servidor NFS:
# Compartir la carpeta home del servidor
#  en modo lectura y escritura y accesible desde la red 192.168.1.0/24
/home       192.168.1.0/255.255.255.0(rw)
# Compartir carpeta tmp a todos como 'solo-lectura'
/tmp        *(ro)
# Compartir carpeta /var/log a un PC como 'solo-lectura'
/var/log    192.168.1.211(ro)
# Compartir carpeta /documentos a todos como 'lectura-escritura'
/documentos *(rw,all_squash,anonuid=1002,anongid=1002)
```

Nota: Los permisos de compartición por NFS no excluyen a los permisos del sistema Linux sino que **prevalecen los más restrictivos**. Si una carpeta está compartida con permiso NFS de lectura y escritura pero en los permisos del sistema solo disponemos de permiso de lectura, no podremos escribir. Si una carpeta está compartida con permisos NFS de lectura y disponemos de permisos de lectura y escritura en el sistema, tampoco podremos escribir. Para poder escribir necesitaremos disponer permiso de lectura y escritura tanto en los permisos del sistema como en los permisos de compartición NFS.

De igual forma, si compartimos la carpeta */home* con permisos de lectura y escritura pero el usuario *alumno* sólo tiene acceso a la carpeta */home/alumno*, no podrá acceder a ninguna otra carpeta dentro de */home* ya que los permisos del sistema se lo impedirán.

Cuando se comparte por NFS, se recomienda restringir al máximo los permisos. Si los usuarios no tienen la necesidad de escribir, debemos compartir con permiso de 'solo lectura'. Si los usuarios solo se conectan desde nuestra red 192.168.0.0/24, debemos permitir el acceso sólo desde dicha red.

Una vez configurado NFS, arrancamos el servicio:

```
# sudo service nfs-kernel-server start
```

Cada vez que se modifique el fichero */etc/exports* es necesario reiniciar el servicio con *restart*.

Para poder acceder desde un equipo cliente a una carpeta compartida por NFS en un servidor, lo primero que tenemos que hacer es instalar los paquetes *portmap* y *nfs-common* que nos permitirán acceder como clientes. Lo normal es que estén instalados, si no lo estuvieran los instalamos:

```
# apt-get install portmap nfs-common
```

Ahora ya estaremos en condiciones de **montar** la carpeta compartida en nuestro sistema de archivos. De ésta manera, el acceso a la carpeta compartida es exactamente igual que el acceso a cualquier otra carpeta de nuestro disco duro.

Para montar la carpeta debemos especificar como tipo de sistema de ficheros el *nfs*, la IP del equipo servidor y la carpeta, y por último la carpeta ya existente del equipo cliente donde se montará.

```
# mount -t nfs ip-del-servidor:/carpeta_del_servidor  carpeta_local
```

Ejemplo, supongamos que un servidor comparte por NFS una carpeta llamada */documentos*. En el PC cliente podemos crear una carpeta llamada */docs-servidor* y montar sobre ella la carpeta compartida en el servidor. Para ello, en el cliente y como *root* ejecutaríamos el siguiente comando:

```
# mount -t nfs ip-del-servidor:/documentos /docs-servidor
```

Podemos especificar la IP del servidor o bien el nombre DNS de la máquina.

A partir de éste momento, podemos comprobar que nuestra carpeta */docs-servidor* contiene la información de la carpeta */documentos* del servidor. Si disponemos de permisos de lectura y escritura, podemos incluso crear o modificar los archivos dentro de nuestra carpeta */docs-servidor* y los cambios se estarán guardando realmente en la carpeta */docs* del servidor.

Para realizar el montaje, debemos hacerlo sobre una carpeta existente en nuestro sistema. Si dicha carpeta de nuestro sistema contiene archivos, estos no estarán accesibles ya que la carpeta nos mostrará los archivos remotos.

Se puede automatizar el montaje de la carpeta en el equipo cliente insertando una línea en el fichero */etc/fstab*:

```
ip-del-servidor:/documentos /docs nfs
```

9. Herramientas de disco

Desfragmentación

Cuando estudiamos las herramientas de disco para el sistema operativo Windows, dedicamos una parte a la herramienta de desfragmentación de disco y a las causas que provocaban dicha fragmentación.

En los sistemas de ficheros típicos de Linux (ext2, ext3, ext4) la fragmentación es mínima gracias a la tecnología que utilizan para asignar espacio a los archivos, por lo que los sistemas Linux no incluyen, típicamente, programas de desfragmentación.

Chequeo

Con el tiempo, los discos duros pueden experimentar errores. Los sistemas operativos suelen incluir herramientas que permiten una revisión de la estructura física y lógica del disco, de modo que, si hay errores, intentan arreglar. La ejecución de estos programas de revisión suele ser una tarea programada con una cierta regularidad.

La herramienta principal de revisión de discos en Linux es **fsck**. Esta utilidad permite revisar sistemas de ficheros, localizar e intentar reparar los problemas que se encuentran. Tiene la sintaxis básica de uso siguiente:

```
# fsck [opciones] dispositivo
```

Esta orden es sólo válida para sistemas ext y hay que utilizarla con los sistemas de archivos desmontados (ya veremos cómo montar y demontar discos). Para comprobar discos FAT es necesario utilizar la orden **dosfsck**. Para la comprobación de discos NTFS es necesario utilizar la herramienta **ntfsck** del paquete **ntfsprogs**.

Se ejecuta el comando fsck con la opción-v (verbose) para que muestre el progreso de lo ejecutado. Ejemplo:

```
# fsck -v /dev/sdb2
```

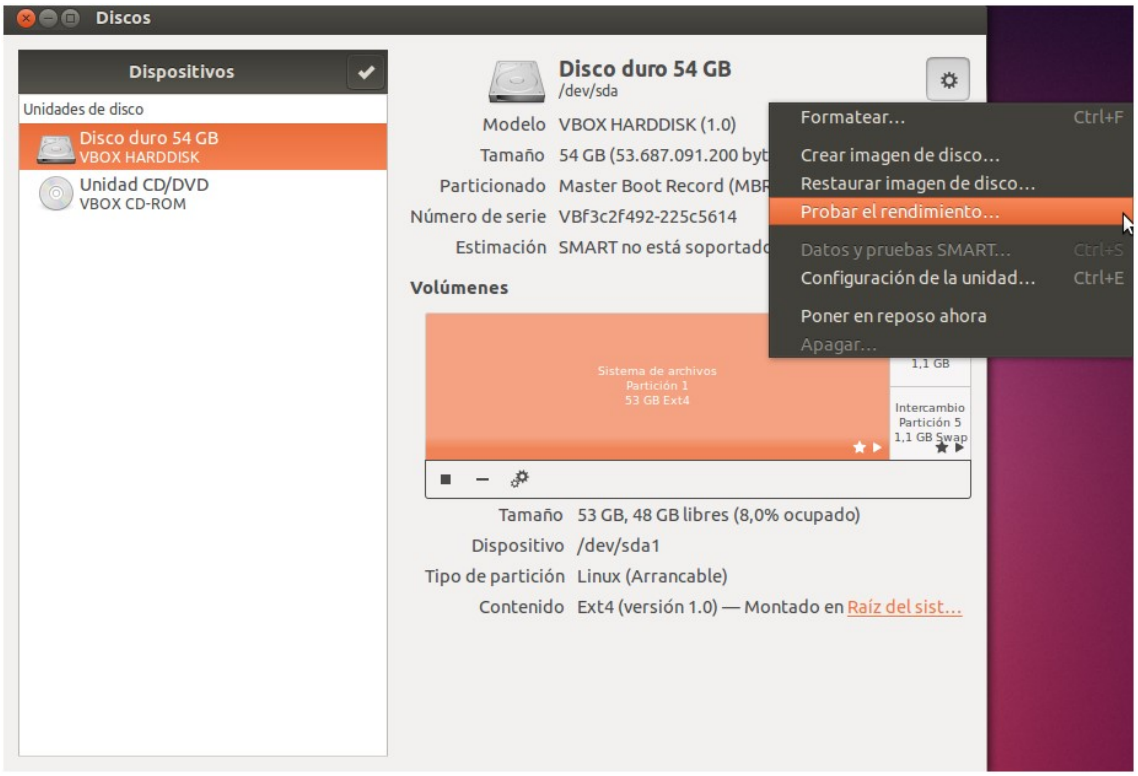
Rendimiento

El rendimiento de un disco duro no sólo es determinado por la velocidad del disco duro, que suele ser el parámetro que se proporciona cuando se vende este, sino que otros factores como el chipset, la velocidad de la CPU, la memoria RAM, la calidad de los controladores o el sistema de archivos que se utiliza también influyen en el rendimiento.

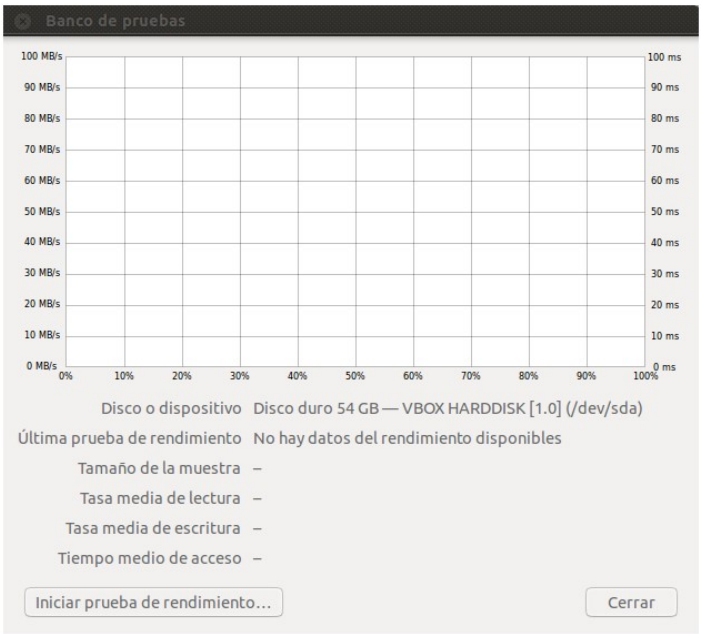
Hay varias herramientas para hacer estudios del rendimiento de los discos, tanto para Windows como para Linux.

La "**Utilidad de discos**" que incorpora Ubuntu permite la realización de tests (de sólo lectura o de lectura y escritura) sobre los discos. La herramienta calcula el rendimiento del disco en función de los ratios de escritura y lectura, junto con el tiempo de acceso medio.

Para acceder a esta herramienta, lanzamos la Utilidad de discos desde el Tablero. En la pantalla principal del programa, seleccionamos el disco del que deseamos estudiar el rendimiento y pulsamos sobre el botón de las *Más Acciones* y seleccionamos *Probar el Rendimiento.*



Nos aparece la pantalla del Banco de Pruebas y ya podemos *Iniciar la prueba de Rendimiento*.



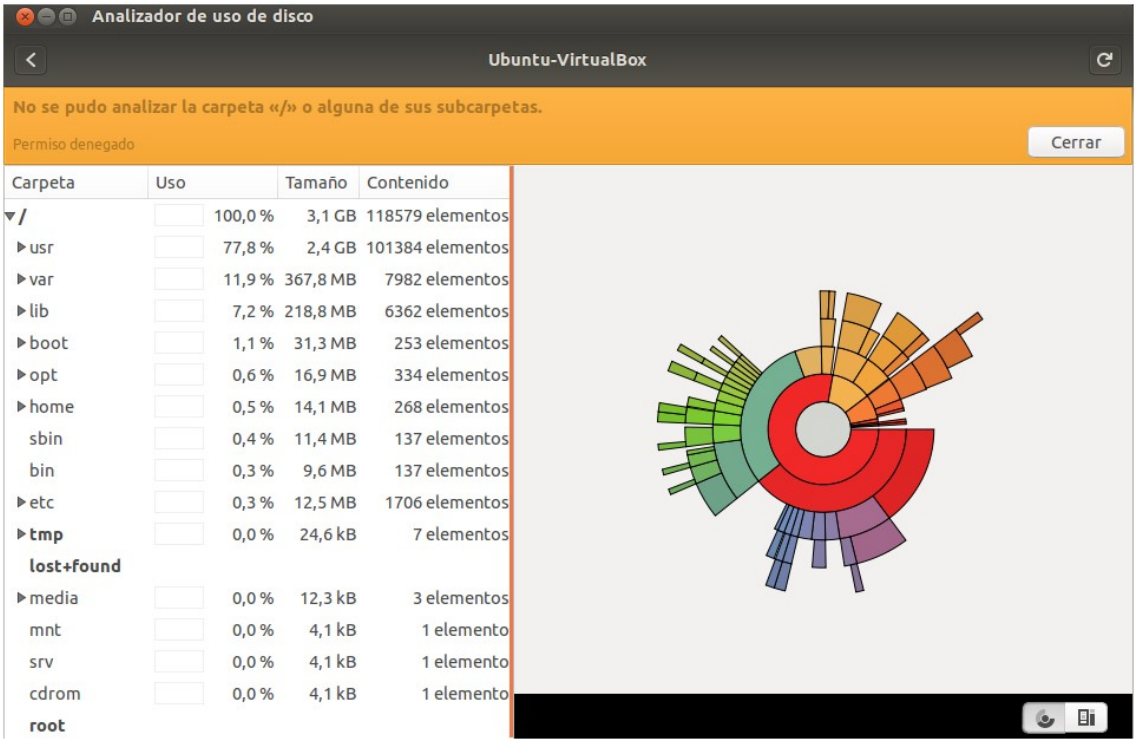
Estadísticas

Cuando se trabaja con discos es habitual tener que hacer estudios sobre el uso de las unidades y, si es posible, visualizar gráficamente la distribución de la información para averiguar qué ocupa más espacio en el disco y hacer una distribución óptima de los recursos.

Todos los sistemas operativos actuales disponen de herramientas para la elaboración de estadísticas de uso de disco.

Ubuntu incorpora una herramienta gráfica muy interesante para la realización de estudios de disco llamada **baobab** (Analizador de uso de disco). Esta herramienta está incluida en el paquete gnome-systemutils, así que se puede ejecutar directamente desde cualquier terminal y desde el Tablero.

Baobab nos permite analizar el árbol de directorios completo o un directorio específico indicado por el usuario, independientemente de que sea local o remoto. El programa devuelve el resultado del estudio mediante un árbol y un diagrama de anillos sobre cómo está distribuido y ocupado el disco. Cuando se pulsa sobre cualquiera de los anillos que simboliza la ocupación de un directorio en concreto se puede visualizar el estudio de disco de esta zona del sistema en particular.



También existen herramientas por línea de comandos. El comando **df** permite obtener estadísticas de ocupación de cada uno de los sistemas de ficheros montados. Sin ningún parámetro, busca información sobre todos los sistemas de ficheros, pero se le puede pasar como parámetro tanto un periférico como un punto de montaje.

Ejemplo de obtención de estadísticas de empleo:
Se realiza un estudio completo de los sistemas de ficheros montados actualmente:

```
$ df
```

También se puede estudiar la ocupación de disco de una estructura concreta (un directorio y todo el contenido de éste). Para ello, hay que utilizar el comando **du** (disk usage). Esta instrucción, si no se indica ninguna ruta como parámetro, realiza el estudio sobre el directorio actual.

Ejemplo de estudio de empleo de una estructura:

Se hace un estudio de disco del directorio del usuario actual:

```
$ du ~

316 /home/estudiante/.thumbnails/normal
320 /home/estudiante/.thumbnails
...
4 /home/estudiante/.gnome2_private
58393 /home /estudiante
```

El resultado se da en KB, pero se puede modificar para que utilice el formato legible por humanos (- h):

```
$ du -h ~

316K /home/estudiante/.thumbnails/normal
320K /home/estudiante/.thumbnails
...
4,0K /home/estudiante/.gnome2_private
58M /home /estudiante
```

Se puede mostrar únicamente el total :

```
$ du -s ~

58M /home/estudiante
```

Por defecto, **du** no se limita a un solo sistema de archivos, y continúa examinando si encuentra un punto de montaje. Para evitar la exploración de puntos de montaje situados en particiones o dispositivos diferentes, hay que utilizar la opción -x.

Material elaborado a partir de:

- Guía de Ubuntu Server https://help.ubuntu.com
- LPIC-1 Guía de estudio. Christine Bresnahan, Richard Blum. Anaya Multimedia.
- LPIC-2 Guía de estudio. Roderick W. Smith. Anaya Multimedia.
- Apuntes de Sistemas Informáticos. DAW. José Ramón Pellicer. IES Camp de Morvedre.
- Curso Redes de Área Local: Aplicaciones y Servicios en Linux. ITE. Ministerio de Educación.
- Administració de la Informació. Implantació de Sistemes Operatius. José Luis Antúnez Reales, Institut Obert Catalunya.
- Apuntes Implantación de Sistemas Operativos. José Antonio Carrasco Díaz. IES Romero Vargas.