Sistemas operativos: Gestión de archivos y almacenamiento

Glosario de términos

- Archivo: unidad lógica mínima que contiene información.
- Desfragmentación: proceso de unión de bloques de datos de un mismo archivo, evitando la disgregación o esparcimiento de estos entre si en el medio de almacenamiento.
- Directorio: contenedores lógicos de ficheros o directorios.
- Enlace simbólico: tipo de archivo en sistemas de archivos ext4, entre otros, que referencia a otros archivos o directorios del mismo u otro sistema de archivos.
- Estructura de directorios de un sistema operativo: Conjunto de directorios que constituyen la organización y despliegan los archivos y subdirectorios del propio sistema operativo.
- Formateo: Proceso mediante el cual se instala un sistema de archivos en una partición.
- I-nodo: estructura de datos, en sistemas de archivos ext4, entre otros, que almacenan meta-información del archivo que representan.
- Journaling: registro del sistema de archivos que evita la inconsistencia de ficheros y facilita la recuperación de datos en los medios de almacenamiento.
- Partición: división interna del dispositivo de almacenamiento que permite organizar la información y facilitar la gestión del almacenamiento.
- RAID: configuración de un grupo de discos independientes para aumentar la integridad, la capacidad de almacenamiento, la velocidad de transferencia o disminuir el riesgo a fallos.

Introducción

Al igual que la gestión de procesos y la gestión de usuarios, la gestión de los archivos es uno de los pilares fundamentales de cualquier sistema operativo.
Existen multitud de sistemas de archivos aunque nosotros nos vamos a centrar en los más usados: ext4 y NTFS.
Los sistemas operativos proveen herramientas para la gestión del almacenamiento, ya sea por línea de comandos o por interfaz gráfica, para realizar particiones de los medios de almacenamiento, formatear, montar y desmontar los sistemas de archivos, desfragmentar, chequear los sistemas de archivos, busca información y crear esquemas RAID.

Sistemas de archivos
En los sistemas de archivos el archivo o fichero el la unidad lógica mínima de almacenamiento que contiene información.

Otro elemento empleado por los sistemas de archivos son los directorios o carpetas. Estos son ficheros que actúan de contenedores lógicos de ficheros o directorios. Independientemente del sistema de archivos donde se defina el directorio, este almacena información relativa a la localización física de la información y los atributos propios de cada archivo o directorio que contenga.

Para administrar el espacio libre y el espacio ocupado se han de definir espacios de asignación. Estas son las unidades físicas mínimas de almacenamiento gestionadas por los sistemas de archivos. Los sistemas de archivos definen el tamaño del espacio de asignación (también llamado unidad de asignación o clúster) durante la instalación del propio sistema de archivos (formateo).Este espacio determina el tamaño mínimo que ocupará un archivo en el medio de almacenamiento. A nivel físico (hardware), el dispositivo administra sectores, sin embargo, el sistema de archivos gestiona clústeres.

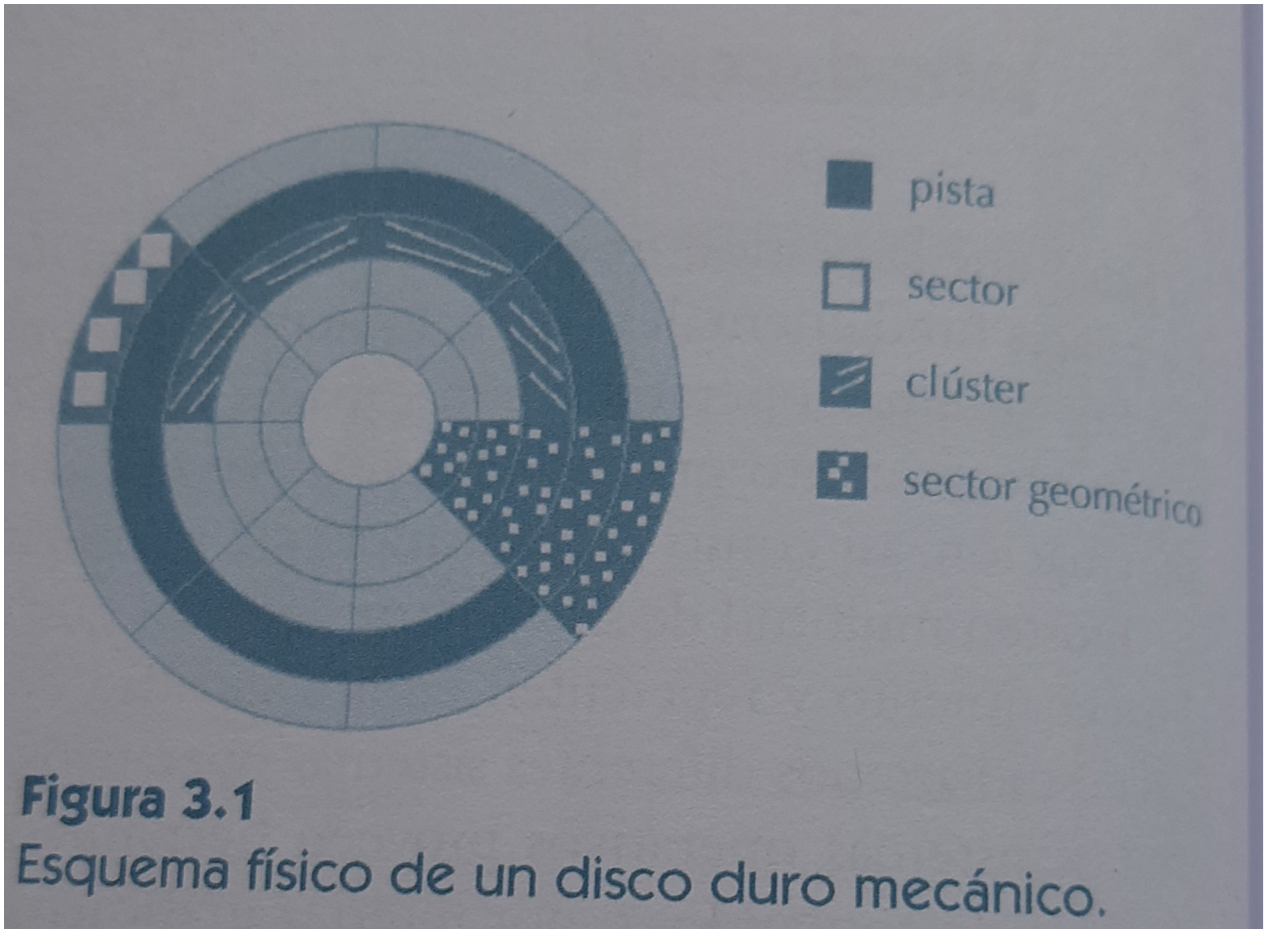

Figura 3.1
Esquema físico de un disco duro mecánico.

La planificación de este espacio es muy importante, siendo ideal un equilibrio entre el promedio del tamaño de los archivos que vaya a alojar el sistema de archivos (evitando así que se desperdicie espacio interno al clúster, también conocido como fragmentación interna) y el tamaño del volumen (facilitando la administración del sistema de archivos).

FAT32 (File Allocation Table)
Es un sistema de archivos creado para MS-DOS. La administración del espacio de almacenamiento es sencilla, por lo que se convierte en un sistema de archivos muy extendido en la mayoría de sistemas operativos. De tal manera, que se suele emplear en dispositivos utilizados para intercambiar datos en computadoras con varios sistemas operativos, videoconsolas, televisores, etc.
Sus limitaciones son:

- Imposibilidad de gestionar particiones superiores a 8 TB (32GB en Microsoft Windows) y archivos de más de 4 GB.
- Bajo rendimiento.
- Inseguro: no permite encriptación, sus atributos y permisos son limitados y no permite journaling.

JOURNALING: es un registro o diario del sistema de archivos. Los sistemas de archivos que hacen uso de este sistema se denominan transaccionales. Consiste en registrar una serie de acciones previas a la operación sobre el sistema de archivos que se vaya a realizar. De tal manera, que si se produce un error durante alguna operación (copiado, borrado, creación , etc), el sistema puede deshacer los cambios y volver a la situación original.

exFAT

Es una evolución del sistema FAT32, que elimina sus principales limitaciones. Se pueden tratar archivos de hasta 16 EB y mantiene la ligereza frente a sistemas de archivos más avanzados, como NTFS y APFS. Aunque sigue resultando inseguro, es ideal para medios de almacenamiento FLASH portables de gran capacidad y compatibles entre sistemas operativos.

NTFS

- Se considera el sistema de archivos estándar de Microsoft Windows. Sus mejoras son considerables con respecto a FAT32, primando la seguridad y la confiabilidad. Sus ventajas son:
- Emplea journaling.
- Permite cifrado y compresión.
- Reduce significativamente la fragmentación y aumenta la velocidad de búsqueda de archivos con respecto a FAT32.
- Puede llegar a gestionar volúmenes de hasta 16 EB y archivos de hasta 16 TB.
- Emplea Unicode para el nombre de archivos, con hasta 256 caracteres.

APFS
Sistema de archivos empleado por Apple Inc. para sus medios de almacenamiento. Sus características son similares a NTFS y ext4 por lo que permite administrar volúmenes de hasta 8 EB. Permite encriptación.

ext4
Sistema de archivos predeterminado para sistemas operativos de tipo Linux en su cuarta versión. Incluye journaling, maneja archivos de hasta 16 TB y volúmenes de hasta 1 EB.
A diferencia de NTFS, ext4 no emplea extensiones como parte del nombre del los archivos. En Microsoft Windows un nombre de archivo se divide en <nombre>.<extensión>, donde extensión es un conjunto de caracteres (3 ó 4) que se asocian con programas para que el sistema operativo reconozca la manera de ejecutar el archivo.
No obstante, muchos nombres de archivos en Linux incorporan sufijos separados por un punto en el nombre del fichero por convención, pero no como requisito establecido por el sistema de archivos.

Sistema de archivos en Linux / Unix

Los sistemas de archivos están situados en dispositivos modo bloque, como discos o cintas.

Los sistemas Unix pueden manejar uno o varios discos físicos cada uno de los cuales puede contener uno o varios sistemas de archivos. Los sistemas de archivos son particiones logicas de archivos.

Cada disco es considerado como un dispositivo lógico. Un controlador del disco se va a encargar de transformar las direcciones lógicas (se las da el kernel) en direcciones físicas.

Un sistema de archivos se compone de una secuencia de bloques lógicos , cada uno de los cuales tiene un tamaño fijo (homogéneo). El tamaño del bloque es el mismo para todo el sistema de archivos y suele ser múltiplo de 512.

El tamaño elegido para el bloque va a influir en las prestaciones globales del sistema. Por un lado interesa que los bloques sean grandes para que la velocidad de transferencia entre disco y memoria sea grande. Pero si los bloques lógicos son demasiado grandes la capacidad de almacenamiento del

disco se puede ver desaprovechada cuando abundan los archivos pequeños que no llegan a ocupar un bloque completo. Tamaños típicos de bloque son: 512, 1024 y 2048 bytes.

Estructura de un sistema de archivos Unix:

La lista de inodos (nodos índice) se encuentra a continuación del superbloque. Esta lista tiene una entrada por cada archivo donde se guarda una descripción del mismo.

¿Qué son los inodos?

Un inodo es una estructura de datos presente en sistemas de archivos de Unix y Linx que almacena información sobre un archivo de nuestro sistema de archivos. Un inodo no tiene nombre y se identifica mediante un número entero único. El nombre al cual apunta el inodo es guardado en una tabla que guarda el kernel. Cada inodo únicamente puede contener datos de un solo archivo del sistema de archivos. Por lo tanto, si tenemos 4 archivos y 4 directorios estaremos usando 8 inodos.

Los sistemas de archivos tienen un número finito de inodos. Esto limita cantidad de archivos que puede ser creados y un riesgo al sistema cuando se tiene una aplicación que genera muchos archivos pequeños, ya que podrían usar todos los inodos de un sistema de archivos aunque exista espacio de almacenamiento.

Los campos de que se compone un inodo en disco son los siguientes:

- *Identificador del propietario del archivo*. La posesión se divide entre un propietario individual y un grupo de propietarios, y define el conjunto de usuarios que tienen derecho de acceso al archivo. El superusuario tiene derecho de acceso a todos los archivos del sistema de archivos.
- *Tipo de archivo*. Los archivos pueden ser ordinarios de datos (regulares), directorios, especiales de dispositivo (en modo carácter o en modo bloque) y tuberías (o pipes).
- *Tipo de acceso al archivo*. El sistema protege a los archivos estableciendo tres niveles de permisos: permisos del propietario, del grupo de usuarios al que pertenece el propietario y el resto de los usuarios. Cada clase de usuarios puede tener habilitados o deshabilitados los derechos de lectura,

escritura y ejecución. Para los directorios el derecho de ejecución significa poder acceder o no a los archivos que contiene.

- *Tiempos de acceso al archivo*. Estos campos dan información sobre la fecha de la última modificación del archivo, la última vez que se accedió a él (día y hora de la última modificación y acceso al archivo) y la última vez que se modificaron los datos de su inodo (día y hora de la última modificación de su inodo).

- *Número de enlaces (de nombres) del archivo*. Representa el total de los nombres que el archivo tiene en la jerarquía de directorios. Como veremos más adelante, un archivo puede tener asociados diferentes nombres que correspondan a diferentes rutas, pero a través de los cuales accedemos a un mismo inodo y por consiguiente a los mismos bloques de datos. Es importante destacar que en el inodo no especifica el (los) nombre(s) del archivo.
- *Entradas para los bloques de dirección de los datos de un archivo*. Si bien los usuarios tratan los datos de un archivo como si fueran una secuencia de bytes contiguos, el *kernel* puede almacenarlos en bloques que no tienen por qué ser contiguos. En los bloques de dirección es donde se especifican los bloques de disco que contienen los datos del archivo.
- *Tamaño del archivo*. Los bytes de un archivo se pueden direccionar indicando un desplazamiento a partir de la dirección de inicio del archivo (desplazamiento 0). El tamaño del archivo es igual al desplazamiento del byte más alto incrementado en una unidad. Por ejemplo, si un usuario crea un archivo y escribe en el sólo un byte en la posición 2000, el tamaño del archivo es 2001 bytes.

Hay que especificar que el nombre del archivo no queda especificado en su inodo. Como veremos mas adelante, es en los archivos de tipo directorio donde a cada nombre de archivo se le asocia su inodo correspondiente.
También hay que indicar la diferencia entre escribir el contenido de un inodo en disco y escribir el contenido del archivo. El contenido del archivo (sus datos) cambia sólo cuando se escribe en él. El contenido de un inodo cambia cuando se modifican los datos del archivo o la situación administrativa del mismo (propietario, permisos, enlaces, etc.).

Mediante el comando stat podemos consultar la información que un inodo guarda sobre un fichero.

Enlaces simbólicos

La manera más sencilla de comprender que es un enlace simbólico en Linux es compararlo con el «enlace directo» o «shortcut» en Windows. El fichero o directorio se encuentra en un único punto del disco y los enlaces son un

puntero contra él. Cada enlace simbólico tiene su propio número de inodo lo que permite hacer enlaces simbólicos entre distintos sistemas de ficheros.

Para crear enlaces (tanto simbólicos como duros) usamos el **comando ln**. En este caso vamos a crear un enlace simbólico (parámetro `-s`) del fichero test:

```
$ ln -s test enlace-a-test
```

Si listamos ambos veremos que el enlace tiene el carácter `l` que lo identifica como enlace simbólico:

```
$ ls -l

lrwxrwxrwx 1 alex alex 4 2011-04-27 18:59 enlace-a-test -> test

-rw-r--r-- 1 alex alex 0 2011-04-27 18:58 test
```

Para confirmar que el enlace simbólico tiene un inodo distinto usamos el comando stat:

```
$ stat test
  File: «test»

  Size: 0               Blocks: 0          IO Block: 4096   archivo regular vacío

Device: 804h/2052d      Inode: 73793       Links: 1

Access: (0644/-rw-r--r--)  Uid: ( 1000/    alex)   Gid: ( 1000/    alex)

Access: 2011-04-27 18:58:53.124142406 +0200

Modify: 2011-04-27 18:58:53.124142406 +0200

Change: 2011-04-27 18:58:53.124142406 +0200


$ stat enlace-a-test

  File: «enlace-a-test» -> «test»

  Size: 4               Blocks: 0          IO Block: 4096   vínculo simbólico

Device: 804h/2052d      Inode: 77212       Links: 1

Access: (0777/lrwxrwxrwx)  Uid: ( 1000/    alex)   Gid: ( 1000/    alex)

Access: 2011-04-27 18:59:07.812139890 +0200

Modify: 2011-04-27 18:59:06.460112888 +0200

Change: 2011-04-27 18:59:06.460112888 +0200
```

También lo podemos verificar sacando el inodo en el ls (`-i`):

```
$ ls -li

77212 lrwxrwxrwx 1 alex alex 4 2011-04-27 18:59 enlace-a-test -> test

73793 -rw-r--r-- 1 alex alex 0 2011-04-27 18:58 test
```

Hay que tener en cuenta, que en Linux / Unix (al igual que con los accesos directos de Windows), si borramos el fichero o directorio origen, el enlace simbólico permanece pero los datos desaparecen para siempre.

Enlaces duros (hard links)

Los enlaces duros lo que hacen es asociar dos o más ficheros compartiendo el mismo inodo. Esto hace que cada enlace duro es una copia exacta del resto de ficheros asociados, tanto de datos como de permisos, propietario, etc. Esto implica también que cuando se realicen cambios en uno de los enlaces o en el fichero este también se realizará en el resto de enlaces.

Los enlaces duros no pueden hacerse contra directorios.

El comando para crear enlaces en ubuntu es ln.

Vamos a crear un hard link contra el fichero «test» de antes y veremos que efectivamente comparten inodo y que los datos se sincronizan entre ambos:

```
$ ln test enlace-duro-test

$ ls -li

73793 -rw-r--r-- 2 alex alex 5 2011-04-27 19:09 enlace-duro-test

73793 -rw-r--r-- 2 alex alex 5 2011-04-27 19:09 test
```

En la primera columna verificamos que tienen el mismo número de inodo y en la tercera se especifica cuando enlaces duros tiene el fichero. Si hacéis cambios en uno de ellos veréis que también se hacen en el resto.

Diferencias entre soft y hard links

- Los enlaces simbólicos se pueden hacer con ficheros y directorios mientras que los duros solo entre ficheros.
- Los enlaces simbólicos se pueden hacer entre distintos sistemas de ficheros, los duros no.
- Los enlaces duros comparten el número de inodo, los simbólicos no.

•En los enlaces simbólicos si se borra el fichero o directorio original, la información se pierde, en los duros no.

•Los enlaces duros son copias exactas del fichero mientras que los simbólicos son meros punteros o «accesos directos».

Tipos de ficheros

Los tipos de ficheros en Linux son muy importantes. Cualquier elemento físico (discos, impresoras, etc) o lógico (directorio, enlace, etc) se representan en Linux mediante un archivo, lo que facilita y estandariza la gestión de todos ellos. Se distinguen 4 tipos elementales de ficheros:

a) Regulares. Son ficheros ordinarios que contienen información de diversa naturaleza. Dentro de ellos se incluyen los ficheros ejecutables como un tipo de fichero regular que tiene algún permiso de ejecución.

b) Directorios. Almacenan en su bloque de datos el número de i-nodo y el nombre de los archivos que contiene.

c) Enlaces.

- Enlaces duros: se trata de reutilizar un i-nodo asignándole nombres distintos y localizándose en diferentes directorios o en el mismo. Sólo se pueden establecer sobre ficheros regulares, no sobre directorios. Para crear enlaces duros se utiliza el comando *ln fichero fichero_enlace*
- Enlaces simbólicos o enlaces blandos. El directorio que contiene un enlace simbólico almacena un i-nodo (diferente al i-nodo con el archivo que enlaza) y un nombre de fichero. Para crear enlaces simbólicos se debe incluir la opción -s con ln: ln -s fichero fichero_enlace.

Nota: podemos utilizar el comando touch sin argumentos para crear archivos vacíos:

touch archivo1

también podemos usar ese comando para modificar la fecha de acceso y modificación de un fichero sin necesidad de modificarlo acceder a su contenido.

d) Dispositivos. Son archivos que representan a dispositivos físicos. En el directorio /dev se localizan la mayoría de ellos. Se distinguen 2 tipos de dispositivos:

- dispositivos por caracteres: suelen ser aquellos que no disponen de sistema de archivos como impresoras, teclados, terminales de texto, etc. Transfieren los datos carácter a carácter.
- Dispositivos por bloques: almacenan la información en bloques de datos físicamente, como por ejemplo, discos duros, cintas magnéticas, unidades de flash, etc.

Además, existen dispositivos virtuales, los cuales tienen un tratamiento especial, como *dev*/null (o cubo de basura) que se utiliza para descartar toda la información que se le envía.

```
cat pruebas 2>/dev/null
```

En el ejemplo anterior si pruebas no existe, el mensaje de error se envía a la basura.

El comando ls tiene activada por defecto la opción de distinguir el tipo de archivo empleando colores:

- blanco: archivo regular
- verde: archivo ejecutable
- azul: directorio
- cian: enlace simbólico
- rojo: enlace roto

Gestión de almacenamiento por línea de comandos en Linux

Linux puede tratar con una partición de disco cuando esta contiene un sistema de archivos y se anexa a su árbol de directorios mediante un directorio común que se denomina punto de montaje. Este directorio es uno más entre el conjunto de directorios, al que se le otorga la capacidad de acceder a través de él a un subsistema de archivos.

Nota: ya sabemos que la partición es una división física del disco duro, que permite organizar la información, incrementar la eficiciencia de acceso a los datos en el disco, aumentar la seguridad e instalar diferentes sistemas de archivos, así como sistemas operativos.

Los dispositivos de almacenamiento se administran a través del directorio del sistema /dev (al igual que el resto de dispositivos físicos del equipo), el cual contiene archivos que se agrupan por tipos, atendiendo al comienzo del nombre.

Nota: *dev/sd* es la interfaz para discos SCSI, SATA y unidades con conexión USB (unidades FLASH o discos duros externos).*

La sintaxis para identificar cada dispositivo en /dev es la siguiente:

dev<id_dispositivo><letra_orden><número_partición>

Cuando veamos hda (y hdb, hdc, etc.) son unidades IDE / ATA-1, mientras que sda (y sdb, etc.) son unidades SCSI o SATA.

Ejemplos:

- */dev/hda: primer disco duro IDE*
- */dev/sdb: segundo disco duro SCSI, SATA o con conexión USB*
- */dev/sdc2: segunda partición del tercer disco duro SCSI, SATA, o con conexión USB*
- */dev/ttyS0: primer puerto serie*

Montaje y desmontaje

La orden que realiza el montaje de un sistema de archivos es *mount*. Gracias a ella, se monta cualquier sistema de archivos reconocible por el núcleo de Linux en un punto de montaje del sistema. Todos los sistemas de archivos se montan directamente por nosotros o indirectamente durante el arranque del sistema, a excepción del sistema de archivos raíz "/". La sintaxis de la orden *mount* es la siguiente:

mount [-avwrt] [tipo] [dispositivo] [punto_de_montaje]

La orden *mount* sin modificadores ni argumentos lista los sistemas de archivos montados actualmente en el sistema.

Por defecto Ubuntu monta automáticamente un sistema de archivos cuando detecta que un pendrive ha sido conectado en un puerto USB o cuando se inserta un nuevo disco DVD en el lector.

Para dejar de usar completamente un dispositivo con sistema de archivos, este se debe desmontar. Para el desmontaje se emplea el comando *umount* , el cual puede completarse si no existen directorios en uso del sistema de archivos objeto a desmontar o procesos lanzados desde él. Para ejecutar el desmontaje se puede indicar la partición o el punto de montaje:

umount <dispositivo> o umount <punto de montaje>

Ejemplos:

umount */dev*sdb1
umount */home/Luis/pendrive*

Las particiones que no son de swap se pueden listar con el comando *df*. Usando el modificador *-k* el resultado se muestra más entendible para los humanos.