FUNCIONAMIENTO

DE LOS DISCOS

Y

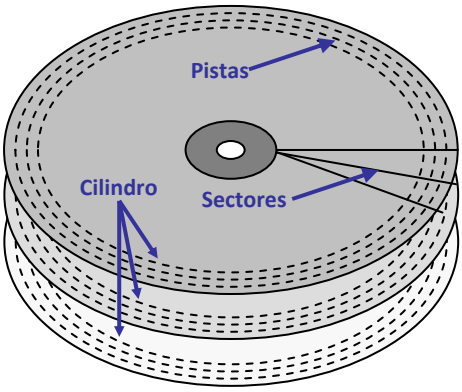
ORGANIZACIÓN DE LOS DATOS

1. ORGANIZACIÓN FÍSICA

1.1. DISCOS MAGNÉTICOS

Llamamos disco duro (HD) al dispositivo de almacenamiento masivo que nos permite grabar datos de forma aleatoria.

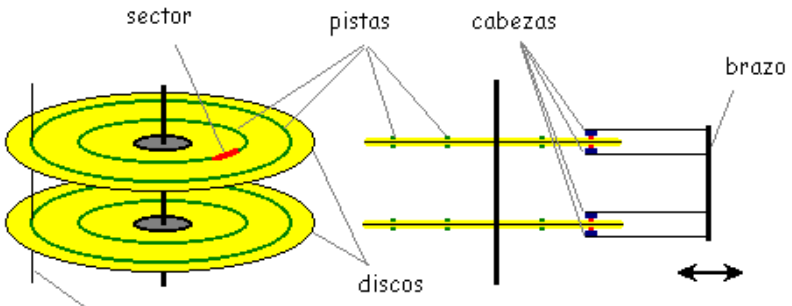
Un HD esta compuesto de la unidad de disco y el disco en si mismo. Este último es un conjunto de uno o mas **platos** de metal recubiertos de una capa de material magnetizable (excepto las **unidades de estado sólido** SSD que usan memorias no volátiles tipo memoria flash) que están que giran en tono a un eje común. Cada lado del plato de denomina **cara**.



Llamamos **pista** de un disco a cada una de las circunferencias concéntricas en que se divide cada cara. Las pistas se numeran desde la parte exterior empezando con el número 0. El número de pistas varía desde las 80 de un disquete (de 0 a 79) a varios cientos en los discos duros.

Un **cilindro** es el conjunto de pistas del mismo número de todas y cada una de las caras. Un HD tiene tantos cilindros como pistas hay en una cara. Ya que todas las **cabezas** lectograbadoras se desplazan al mismo tiempo, es más óptimo completar cilindros que caras.

Las pistas se dividen en **sectores**. Cada sector contiene la misma cantidad de información, normalmente 512 bytes. Antiguamente el número de sectores por pista era fijo, lo cual desaprovechaba el espacio significativamente, ya que en las pistas exteriores pueden almacenarse más sectores que en las interiores. Así, apareció la tecnología ZBR (**grabación de bits por zonas**) que aumenta el número de sectores en las pistas exteriores, y utiliza más eficientemente el disco duro.



Llamamos **cluster** o **bloque** a los grupos de información que vamos a utilizar (que dependerán del sistema operativo que estemos usando). Un cluster está compuesto por varios sectores. En un H.D., el cluster puede contener de 2 a más de 16 sectores, todos del mismo cilindro. Así, si tenemos sectores de 512 bytes en un HD con 4 platos, y todos los sectores de un cilindro conforman un cluster, éste será de 4096 bytes = 4 Kb (512 bytes/sector * 8 caras)

Cuando se graba un fichero en un disco, se usan cluster completos, lo que quiere decir que aunque el archivo sea mas pequeño que el cluster éste se usará entero

1.1.1 FORMATEO A BAJO NIVEL

Mediante el formateo a bajo nivel, o formateo físico, se crean las pistas y sectores en los discos magnéticos. Para ello, se crean áreas de identificación en las caras del disco que el controlador de disco utiliza para numerar los sectores e identificar el principio y fin de cada uno. Estas áreas de identificación están situadas, dentro de una pista, delante y detrás del área de datos (ID) del sector. El área de datos del sector suele ser de 512 bytes, pero el tamaño del sector aumenta un poco al añadir las áreas de identificación.

El área de identificación precedente identifica el principio del sector y recibe el nombre de cabecera o prefijo de sector, y contiene el número del sector dentro de cada pista. El área posterior, sufijo o *trailer*, marca el fin del sector y contiene el *checksum* que garantiza la integridad de los datos contenidos en el sector (el *cheksum* es una técnica para comprobar la validez de un paquete de datos. La información del paquete forma una cadena de números; de su suma se obtiene una cifra que debe coincidir con el valor del dato suministrado como *checksum*). Si un sector está defectuoso, se marca como inutilizable.

Este proceso es previo al formateo a alto nivel y, normalmente, ya viene hecho de fábrica y no debe hacerlo el usuario.

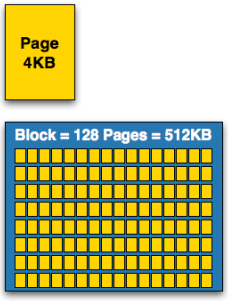
1.2. UNIDADES DE ESTADO SÓLIDO (SSD)

Una **unidad de estado sólido** o **SSD** (*solid-state drive*) es un dispositivo de almacenamiento de datos que usa una memoria no volátil, como la memoria flash, o una memoria volátil como la SDRAM, para almacenar datos, en lugar de los platos giratorios magnéticos encontrados en los discos duros convencionales. Los SSD hacen uso de la misma interfaz que los discos duros, y por tanto son fácilmente intercambiables sin tener que recurrir a adaptadores o tarjetas de expansión para compatibilizarlos con el equipo.

Aunque técnicamente no son discos a veces se traduce erróneamente en español la "D" de SSD como *disk* cuando en realidad representa la palabra *drive*, que podría traducirse como unidad o dispositivo.

Un disco SSD y un pendrive USB se parecen mucho, pero principalmente hay dos cosas que los diferencian. En primer lugar, el disco SSD va conectado por SATA (o PATA) un protocolo mucho más eficiente que el USB. En segundo lugar, un disco SSD, gracias a su avanzada controladora puede leer de muchos módulos de memoria a la vez (unos 8 o 10 es un valor normal en los SSD actuales).

En sistemas con SSD se recomienda desactivar el desfragmentador ya que su uso sobre una unidad SSD no tiene sentido, y reduciría su vida al hacer un uso continuo de los ciclos de lectura y escritura.



Un SSD se compone de páginas de 4KB y bloques normalmente formados por un total de 128 páginas, es decir 512KB. La cuestión es que se pueden llevar a cabo procesos de escritura y lectura sobre páginas, sin embargo, sólo se pueden realizar procesos de borrado sobre bloques como unidad mínima.

Cuando se borra un archivo simplemente se modifica la tabla de ocupación de bloques para indicar que ese espacio está libre. En un disco duro ese espacio no se borra realmente hasta que es sobrescrito por otro archivo y los SSDs funcionan igual.

La diferencia clave está en lo que ocurre cuando se sobrescribe un archivo (en un disco duro simplemente se sobrescribe la zona del disco). En el SSD se modifica la tabla interna y se graban los nuevos datos en una zona de espacio no asignada.

Esto provoca que si en un proceso de escritura no se cuenta con espacio (páginas libres suficientes por cada bloque), se necesita una reorganización de los datos (páginas), y por tanto la necesidad de acometer un borrado (bloques). En primer lugar hay que moverlos a una caché (puede ser la incluida en el SSD si existe en el mejor de los casos, en caso contrario deberá utilizarse memoria RAM temporal). Una vez hecho, se puede iniciar el proceso de borrado para luego volver a escribir los datos que se han tenido que mover a caché. Obviamente, todo este proceso implica una penalización de rendimiento.

Si la escritura cuenta con páginas libres todo este proceso no es necesario, la escritura se produce directamente, y mientras tenga espacio libre (páginas suficientes) el proceso será llevado a cabo de esta forma. Sin embargo, es obvio que con el uso del SSD, esta circunstancia terminará por no ocurrir y las reorganizaciones serán necesarias. En ese momento, el rendimiento del SSD decaerá a medida que aumenten estas reorganizaciones.

2. ESTRUCTURA LÓGICA

Dependiendo si nuestra equipo esta dotado de BIOS o EFI tendremos:

- BIOS
  - El **Master Boot Record** (MBR)
  - Las **particiones**, necesarias para poder colocar los sistemas de archivos.
  - El **espacio no particionado**.
- EFI
  - El **Master Boot Record** (MBR) o **Tabla de partición GUID** (GPT)
  - Las **particiones**, necesarias para poder colocar los sistemas de archivos.
  - El **espacio no particionado**.

2.1. MASTER BOOT RECORD

Es el primer sector (sector cero) de un dispositivo de almacenamiento de datos, como un disco duro. Almacena la tabla de particiones y, en ocasiones, el **bootstrap** o arranque del sistema operativo.

Soporta hasta 4 particiones por unidad física con un límite de 2,2 TB, es decir, un disco duro u otro dispositivo de almacenamiento de 10 TB o más no se podría aprovechar su capacidad al 100%.

446 bytes	Programa del gestor de arranque		
64 bytes	16 bytes	Primera partición	
		1 byte	Marca de partición activa (01000000) 0 no (00000000).
		3 bytes	CHS de inicio
		1 byte	Tipo de partición
		3 bytes	CHS final
		4 bytes	LBA
		4 bytes	Tamaño en sectores
	16 B	Segunda partición	
		1 byte	Marca de partición activa (01000000) 0 no (00000000).
		. . . .	
	16 B	4 bytes	LBA
		4 bytes	Tamaño en sectores
	16 bytes	Cuarta partición	
		1 byte	Marca de partición activa (01000000) 0 no (00000000).
		3 bytes	CHS de inicio
		1 byte	Tipo de partición
		3 bytes	CHS final
		4 bytes	LBA
		4 bytes	Tamaño en sectores
2 bytes	Firma de unidad arrancable ("55 AA" en hexadecimal)		

2.1.1 TABLA DE PARTICIONES

La tabla de particiones está a partir del byte 446 del sector de arranque y ocupa 64 bytes, conteniendo 4 registros de 16 bytes, los cuales definen las particiones primarias. En ellos se almacena

2.1.2 BOOTSTRAP

En principio, es el programa que arranca un sistema operativo como por ejemplo GRUB, LiLo o NTLDR. También es llamado *«Bootstrap Loader»* (cargador de inicialización).

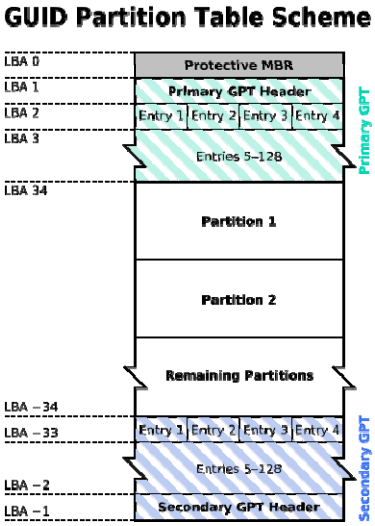
Una vez que la BIOS termina con sus tests e inicializaciones (POST) carga el primer sector en la y comprueba que contenga código válido de arranque (comprueba que esté firmado con 0x55 y 0xAA en los bytes 511 y 512 respectivamente). Entonces carga el bootstrap si lo hay.

El bootstrap de un sistema operativo puede ser sustituido por un programa gestor de arranque independiente o por el *Master boot*, que sólo buscará en la tabla de particiones cuales la partición activa.

En este caso, el bootstrap del sistema operativo deberá estar en el sector de arranque (boot record) de la partición donde éste esté instalado.

2.2. GPT

GPT usa un moderno modo de direccionamiento lógico (LBA, *logical block addressing*) en lugar del modelo cilindro-cabeza-sector (CHS) usado con el MBR. La información de MBR heredado está almacenada en el LBA 0, la cabecera GPT está en el LBA 1, y la tabla de particiones en sí en los bloques sucesivos.



GPT proporciona asimismo redundancia. La cabecera GPT y la tabla de particiones están escritas tanto al principio como al final del disco.

Soporta teóricamente hasta 9,4 ZB y no exige un sistema de archi vos concreto para funcionar

Windows soporta GPT a partir de las versiones de 64 bits.

Un gestor de arranque propio de la EFI permite también la selección y carga directa de los sistemas operativos, eliminando la necesidad de recurrir a gestores de arranque.

2.3. PARTICIONADO DEL DISCO

Consiste en definir varias unidades lógicas en un HD. En los discos magnéticos una partición se compone de un conjunto de cilindros contiguos. En un SSD las particiones tienen que estar alineadas. Esto a grandes rasgos significa que una partición debe comenzar en un punto de la memoria que sea múltiplo del tamaño de borrado de celdas que tenga el SSD (dependiendo de uno u otro podría ser 128, 256, 512KB...).

Cada disco duro constituye una unidad física distinta. Sin embargo, los sistemas operativos no trabajan con unidades físicas directamente sino con unidades lógicas. Dentro de una misma unidad física de disco duro puede haber varias unidades lógicas, esto quiere decir que podemos dividir un disco duro en, por ejemplo, dos particiones y trabajar de la misma manera que si tuviésemos dos discos duros. Cada partición tiene su propio sistema de archivos.

Como mínimo, es necesario crear una partición en cada disco duro. Esta partición puede contener la totalidad del espacio del disco duro o sólo una parte.

Las particiones pueden ser de dos tipos: **primarias** o **lógicas**. Las particiones lógicas se definen dentro de una partición primaria especial denominada partición **extendida**.

En un disco duro MBR sólo pueden existir 4 particiones primarias (incluida la partición extendida, si existe). En GPT se pueden tener hasta 128 particiones primarias. Las particiones existentes se inscribirse en la tabla de particiones. Es necesario que figure una de ellas como partición activa que es aquella a la que el programa de inicialización (*Master Boot*) cede el control al arrancar. El sistema operativo de la partición activa será el que se cargue al arrancar desde el disco duro.

2.3.1 PARTICIÓN PRIMARIA

Aquella cuya información esta contenida en uno de los registros de la tabla de particiones del MBR o GPT:

- Muchos SO sólo pueden ser instalados sobre una de ellas (Windows)
- Se pueden ocultar
- Se pueden definir como particiones activas

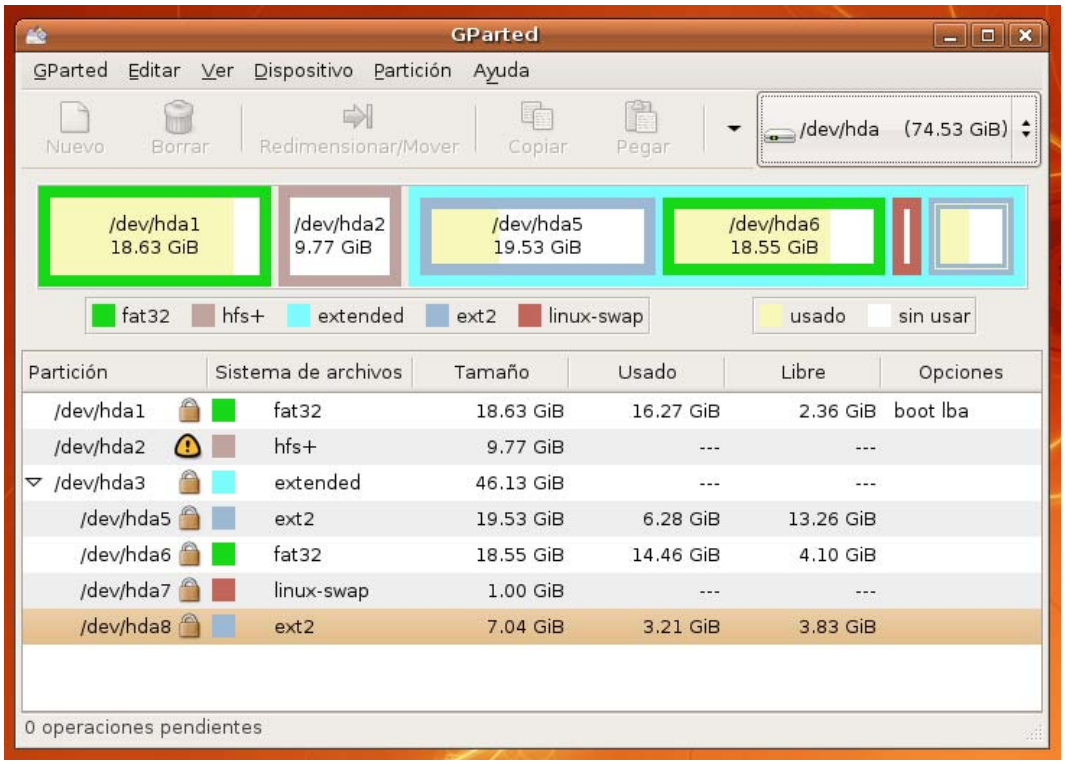
2.3.2 PARTICIÓN EXTENDIDA

Este concepto esta obsoleto en los sistemas GPT. Solo se puede tener una en la tabla de particiones MBR.

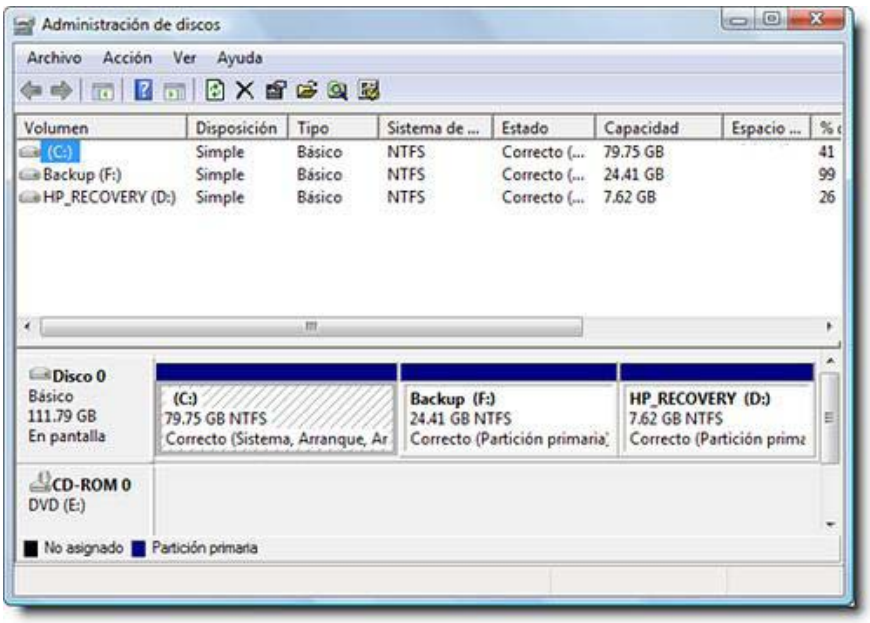
En su Boot Record, existe una entrada con la información de la primera partición lógica. Ésta a su vez contiene un Boot Record, con información de la segunda, etc.

- Sólo algunos SO pueden ser instalados sobre una de ellas (Linux)
- No se pueden ocultar
- No se pueden definir como particiones activas
- Algunos sistemas operativos no pueden acceder a particiones primarias distintas a la suya

A continuación se muestran unos ejemplos de discos duros con espacio particionado en un sistema MBR (La ventana procede de la utilidad de discos GParted de Linux):



En este caso estamos viendo un disco duro con dos particiones primarias (/dev/hda1 con sistema de archivos *FAT32* y /dev/hda2 con sistema de archivos *hfs+*) y una partición extendida con varias particiones lógicas (/dev/hda5 con *ext2*, /dev/hda6 con *FAT32*, linuxSwap con *swap* y /dev/hda8 con *ext2*).



En este otro caso (la ventana pertenece a la utilidad de discos de Windows 7) podemos apreciar tres particiones primarias con sistemas de archivos *NTFS*.

2.4. PARTICIONES VS. DIRECTORIOS.

Ambas estructuras permiten organizar datos dentro de un disco duro. Sin embargo, presentan diferencias:

- Las particiones son divisiones de tamaño fijo del disco duro; los directorios son divisiones de tamaño variable de la partición
- Las particiones ocupan un grupo de cilindros contiguos del disco duro (mayor seguridad); los directorios suelen tener su información desperdigada por toda la partición
- Cada partición del disco duro puede tener un sistema de archivos (sistema operativo) distinto; todos los directorios de la partición tienen el sistema de archivos de la partición.

2.5. SECUENCIA DE ARRANQUE DE UN ORDENADOR.

Lo primero que hace el programa de arranque es un breve chequeo de los componentes hardware (POST).

Tras realizar el POST, la BIOS transfiere el control a la interrupción INT 19H que realiza una rutina denominada *bootstrapping* que consiste, esencialmente, en la búsqueda, carga y ejecución de un sistema operativo básico, encargado de tomar el control del ordenador en el momento de encenderlo.

Primero intenta el arranque desde la primera unidad física indicada en la secuencia de arranque[1]. Si el intento es fallido, repite la operación con la segunda unidad de la lista y así hasta que encuentre una unidad arrancable. Si no existiese ninguna, el programa de arranque mostraría una advertencia.

Suponiendo que arrancamos desde el disco duro, el programa de arranque de la BIOS cederá el control a su programa de inicialización: *Bootstrap*, gestor de arranque o *Master Boot*.

Si es un *master boot*, buscará en la tabla de particiones la partición activa y le cederá el control a su sector de arranque. El programa contenido en el sector de arranque de la partición activa deberá ser el *bootstrap* del sistema operativo.

Si es un *bootstrap* o un gestor de arranque, procederá a su ejecución.

[1] Se define en la configuración de la BIOS (también llamada SETUP) y se almacena en la CMOS.

Normalmente, los gestores de arranque proceden a mostrar un menú selectivo y a marcar como activa la partición seleccionada, cediendo el control a continuación al *master boot*.

Los *bootstrap*, por el contrario, necesitan información que está almacenada en la partición del sistema operativo que lo ha instalado por lo que no arrancaria si se pierde o daña esa partición.

2.6. VOLUMEN

Los discos básicos y los discos dinámicos son dos tipos de configuraciones de disco duro en Windows. La mayoría de los equipos personales están configurados como discos básicos, que son los más sencillos de administrar. Los usuarios avanzados y profesionales informáticos pueden emplear discos dinámicos, que pueden usan varios discos duros de un equipo para administrar datos, normalmente para obtener un mayor rendimiento o confiabilidad.

Un disco básico usa particiones primarias, particiones extendidas y unidades lógicas para organizar datos. Una partición formateada también se denomina volumen (los términos volumen y partición se usan por lo general indistintamente).

Los discos dinámicos pueden contener un gran número de volúmenes dinámicos (aproximadamente 2000), que funcionan como las particiones primarias usadas en discos básicos. En algunas versiones de Windows, es posible combinar discos duros dinámicos independientes en un único volumen dinámico (lo que recibe el nombre de expansión), dividir datos entre varios discos duros (lo que recibe el nombre de sección) para obtener un mayor rendimiento o duplicar datos entre varios discos duros (lo que recibe el nombre de reflejo) para obtener una mayor confiabilidad.

DISCOS DINÁMICOS **contienen volúmenes dinámicos**	**DISCOS BÁSICOS** **contienen particiones**
SIMPLES. Es un volumen dinámico formado por espacio de *un solo disco dinámico*	PARTICIÓN
DISTRIBUIDOS. Es una forma de repartir el espacio no asignado en un sistema *con varios discos* en una única unidad lógica. No puede ser reflejado y no es tolerante a errores, aunque permite extender su tamaño a otras unidades disponibles.	CONJUNTO DE VOLÚMENES
REFLEJADOS. (RAID-1): usan dos copias llamadas espejo, aunque aparecen como una única entidad. Alta fiabilidad.	CONJUNTO DE ESPEJOS
SECCIONADOS. (RAID-0) variante del volumen distribuido. Cuando se guarda un archivo éste se distribuye en las bandas en las que se dividió cada disco duro. Ofrece más rendimiento que el volumen distribuido pero no más seguridad. No se pueden extender a otros discos dinámicos en caso de que sea necesario y tampoco se pueden reflejar.	CONJUNTO DE BANDAS SIN PARIDAD
RAID-5. Tolerante a errores (se pueden recuperar los datos si un disco falla). Tiene sus datos distribuidos en tres o más discos físicos.	CONJUNTO DE BANDAS CON PARIDAD

2.6.1 RAID

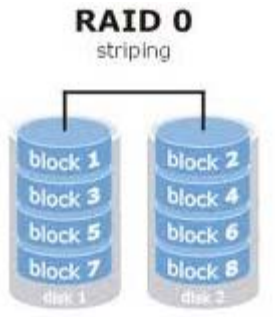
El acrónimo **RAID** (del inglés ***Redundant Array of Independent Disks***), «conjunto redundante de discos independientes», anteriormente conocido como ***Redundant Array of Inexpensive Disks***, «conjunto redundante de discos baratos») hace referencia a un sistema de almacenamiento que usa múltiples disco entre los que se distribuyen o replican los datos, presentándose al usuario como un único volumen.

Dependiendo de su configuración (a la que suele llamarse «nivel»), los beneficios de un RAID respecto a un único disco son uno o varios de los siguientes: mayor integridad, mayor tolerancia a fallos, mayor *throughput* (rendimiento) y mayor capacidad. RAID 0.- Muchos discos como 1, si se estropea uno se pierde la información

Los niveles RAID más comúnmente usados son:

- RAID 0: Conjunto dividido
- RAID 1: Conjunto en espejo
- RAID 5: Conjunto dividido con paridad distribuida

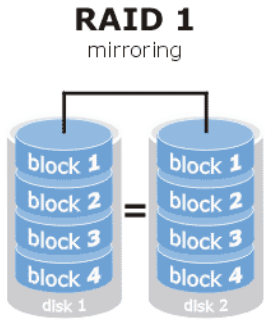
RAID 0



Un **RAID 0** (también llamado **conjunto dividido** o **volumen dividido**) distribuye los datos equitativamente entre dos o más discos sin información de paridad que proporcione redundancia. Es importante señalar que el RAID 0 no era uno de los niveles RAID originales y que no es redundante. El RAID 0 se usa normalmente para incrementar el rendimiento, aunque también puede utilizarse como forma de crear un pequeño número de grandes discos virtuales a partir de un gran número de pequeños discos físicos.

Puede ser creado con discos de diferentes tamaños, pero el espacio de almacenamiento añadido al conjunto estará limitado por el tamaño del disco más pequeño (por ejemplo, si un disco de 300 GB se divide con uno de 100 GB, el tamaño del conjunto resultante será sólo de 200 GB, ya que cada disco aporta 100GB).

RAID 1

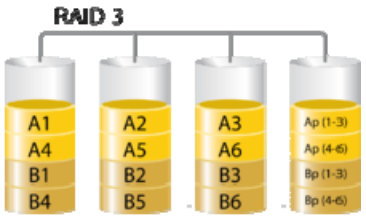


Crea una copia exacta (o **espejo**) de un conjunto de datos en dos o más discos. Esto resulta útil cuando el rendimiento en lectura es más importante que la capacidad.

Un conjunto RAID 1 sólo puede ser tan grande como el más pequeño de sus discos. Un RAID 1 clásico consiste en dos discos en espejo, lo que incrementa exponencialmente la fiabilidad respecto a un solo disco; es decir, la probabilidad de fallo del conjunto es igual al producto de las probabilidades de fallo de cada uno de los discos.

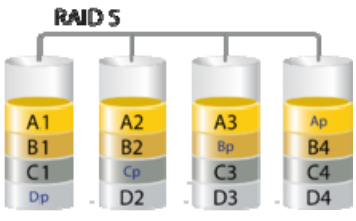
Adicionalmente, dado que todos los datos están en dos o más discos, con hardware habitualmente independiente, el rendimiento de lectura se incrementa aproximadamente como múltiplo lineal del número del copias; es decir, un RAID 1 puede estar leyendo simultáneamente dos datos diferentes en dos discos diferentes, por lo que su rendimiento se duplica

RAID 3



Un **RAID 3** usa división a nivel de bytes con un disco de paridad dedicado. El RAID 3 se usa rara vez en la práctica. Uno de sus efectos secundarios es que normalmente no puede atender varias peticiones simultáneas, debido a que por definición cualquier simple bloque de datos se dividirá por todos los miembros del conjunto, residiendo la misma dirección dentro de cada uno de ellos. Así, cualquier operación de lectura o escritura exige activar todos los discos del conjunto, suele ser un poco lento porque se producen cuellos de botella. Son discos paralelos pero no son independientes (no se puede leer y escribir al mismo tiempo).

RAID 5

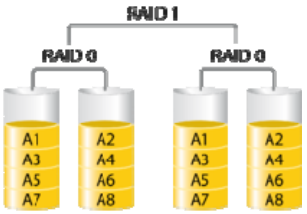


Un **RAID 5** usa división de datos a nivel de bloques distribuyendo la información de paridad entre todos los discos miembros del conjunto. El RAID 5 ha logrado popularidad gracias a su bajo coste de redundancia. Generalmente, el RAID 5 se implementa con soporte hardware para el cálculo de la paridad. RAID 5 necesitará un mínimo de 3 discos para ser implementado.

En el gráfico de ejemplo anterior, una petición de lectura del bloque «A1» sería servida por el disco 0. Una petición de lectura simultánea del bloque «B1» tendría que esperar, pero una petición de lectura de «B2» podría atenderse concurrentemente ya que seria servida por el disco 1.

Una serie de bloques (un bloque de cada uno de los discos del conjunto) recibe el nombre colectivo de división (*stripe*). Cada vez que un bloque de datos se escribe en un RAID 5, se genera un bloque de paridad dentro de la misma división (*stripe*). Un bloque se compone a menudo de muchos sectores consecutivos de disco. Si otro bloque, o alguna porción de un bloque, es escrita en esa misma división, el bloque de paridad (o una parte del mismo) es recalculada y vuelta a escribir. El disco utilizado por el bloque de paridad está escalonado de una división a la siguiente, de ahí el término «bloques de paridad distribuidos». Las escrituras en un RAID 5 son costosas en términos de operaciones de disco y tráfico entre los discos y la controladora.

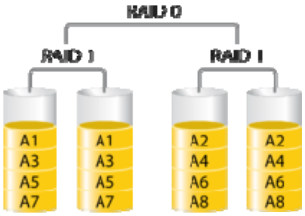
RAID 0+1



Un **RAID 0+1** (también llamado **RAID 01)** es un RAID usado para replicar y compartir datos entre varios discos. La diferencia entre un RAID 0+1 y un RAID 1+0 es la localización de cada nivel RAID dentro del conjunto final: un RAID 0+1 es un espejo de divisiones.

Como puede verse en el diagrama, primero se crean dos conjuntos RAID 0 (dividiendo los datos en discos) y luego, sobre los anteriores, se crea un conjunto RAID 1 (realizando un espejo de los anteriores). La ventaja de un RAID 0+1 es que cuando un disco duro falla, los datos perdidos pueden ser copiados del otro conjunto de nivel 0 para reconstruir el conjunto global. Sin embargo, añadir un disco duro adicional en una división, es obligatorio añadir otro al de la otra división para equilibrar el tamaño del conjunto.

RAID 1+0

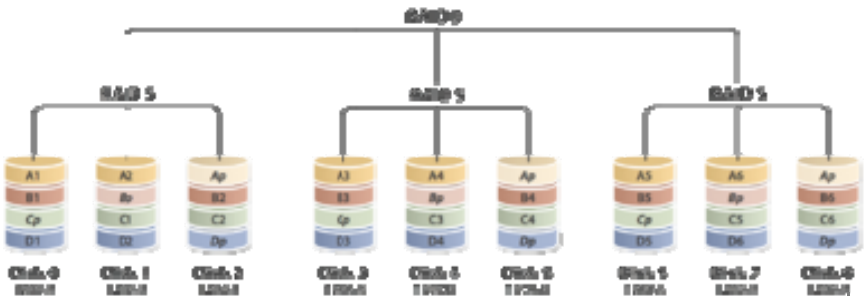


Un **RAID 1+0**, a veces llamado **RAID 10**, es parecido a un RAID 0+1 con la excepción de que los niveles RAID que lo forman se invierte: el RAID 10 es una división de espejos.

En cada división RAID 1 pueden fallar todos los discos salvo uno sin que se pierdan datos. Sin embargo, si los discos que han fallado no se reemplazan, el restante pasa a ser un punto único de fallo para todo el conjunto. Si ese disco falla entonces, se perderán todos los datos del conjunto completo. Como en el caso del RAID 0+1, si un disco que ha fallado no se reemplaza, entonces un solo error de medio irrecuperable que ocurra en el disco espejado resultaría en pérdida de datos. Es a menudo la mejor elección para bases de datos de altas prestaciones, debido a que la ausencia de cálculos de paridad proporciona mayor velocidad de escritura.

RAID 50

Un **RAID 50** combina la división a nivel de bloques de un RAID 0 con la paridad distribuida de un RAID 5, siendo pues un conjunto RAID 0 dividido de elementos RAID 5.



Un disco de cada conjunto RAID 5 puede fallar sin que se pierdan datos. Sin embargo, si el disco que falla no se reemplaza, los discos restantes de dicho conjunto se convierten en un punto único de fallo para todo el conjunto. Si uno falla, todos los datos del conjunto global se pierden. El tiempo necesario para recuperar (detectar y responder al fallo de disco y reconstruir el conjunto sobre el nuevo disco) representa un periodo de vulnerabilidad del conjunto RAID.

La configuración de los conjuntos RAID repercute sobre la tolerancia a fallos general. Una configuración de tres conjuntos RAID 5 de siete discos cada uno tiene la mayor capacidad y eficiencia de almacenamiento, pero sólo puede tolerar un máximo de tres fallos potenciales de disco. Debido a que la fiabilidad del sistema depende del rápido reemplazo de los discos averiados para que el conjunto pueda reconstruirse, es común construir conjuntos RAID 5 de seis discos con un disco de reserva en línea (*hot spare*) que permite empezar de inmediato la reconstrucción en caso de fallo del conjunto. Esto no soluciona el problema de que el conjunto sufre un estrés máximo durante la reconstrucción dado que es necesario leer cada bit, justo cuando es más vulnerable. Una configuración de siete conjuntos RAID 5 de tres discos cada uno puede tolerar hasta siete fallos de disco pero tiene menor capacidad y eficiencia de almacenamiento.

El RAID 50 mejora el rendimiento del RAID 5, especialmente en escritura, y proporciona mejor tolerancia a fallos que un nivel RAID único. Este nivel se recomienda para aplicaciones que necesitan gran tolerancia a fallos, capacidad y rendimiento de búsqueda aleatoria.

A medida que el número de unidades del conjunto RAID 50 crece y la capacidad de los discos aumenta, el tiempo de recuperación lo hace también.

3. GESTIÓN DEL ALMACENAMIENTO SECUNDARIO.

Para poder almacenar los datos en un disco, estos se han de guardar respetando una serie de normas y restricciones, impuestas por el **sistema de archivos** (*file system* o FS), en el cual existen dos tipos fundamentales de elementos:

- **Archivos**: son los elementos encargados de contener los datos
- **Directorios**: conocidos como carpetas, son elementos cuya misión principal es permitir una mayor organización de los archivos dentro del disco. Son, a su vez, archivos que contienen información sobre la ubicación de otros archivos o directorios

3.1. LOS ARCHIVOS

Un fichero es un mecanismo de abstracción que sirve como unidad lógica de almacenamiento de información. El fichero agrupa una colección de informaciones relacionadas entre sí y definidas por su creador.

Los archivos son elementos que almacenan la información del usuario/sistema operativo (programas y datos) en el disco para poder volverla a leer/modificar cuantas veces sea necesario sin que el usuario tenga que preocuparse por la forma y el lugar físico de almacenamiento ni del funcionamiento real de los discos.

A todo fichero le corresponde un nombre único que lo identifique entre los demás.

Junto con el nombre del archivo, el sistema operativo almacena también una serie de datos que varían de un SO a otro, y entre ellos destacan:

- Tamaño
- Fecha y hora de creación, acceso, modificación
- Propietario
- Permisos de acceso para el propietario y otros usuarios del sistema
- Datos para la localización física (cluster/bloque donde están los datos realmente)
- Atributos: permiten modificar la forma en que el SO maneja el fichero:
  - S: Atributo de sistema
  - H: Atributo de oculto
  - R: Atributo de solo lectura
  - A: Atributo de archivo
  - D: Directorio
  - Otros atributos que se colocan para archivos especiales

3.1.1 TIPOS DE ARCHIVOS

Los archivos se pueden dividir en tres grupos: metadatos, regulares o normales y de dispositivo.

Los **metadatos** son archivos que contienen información sobre otros archivos: los directorios de los sistemas FAT, los i-nodos de Linux, la MFT de NTFS, etc.

Existen **dispositivos** cuya E/S se realiza como si fuesen ficheros, por lo tanto es razonable asociarles ficheros para simplificar y hacer más transparente el intercambio de información con dichos dispositivos. En los capítulos siguientes nos centraremos en los ficheros regulares y en los directorios.

Los archivos **regulares o normales** se clasifican según su contenido. La información que contiene un fichero la define su creador. Hay muchos tipos diferentes de información que puede almacenarse en un fichero: programas fuente, programas objeto, datos numéricos, textos, registros contables, etc., tiene una cierta estructura, definida según el uso que se vaya a hacer de él. Por ejemplo, un fichero de texto es una secuencia de caracteres organizados en líneas; un fichero fuente es una secuencia de subrutinas y funciones, etc.

Aunque no es necesario en todos los sistemas de archivo, suelen indicar su contenido mediante la extensión, siendo la estructura típica del nombre del archivo:

nombre.extension

Dependiendo del sistema de archivo, el nombre puede ser más o menos largo y la extensión tiene, habitualmente 3 caracteres. Mediante la extensión se puede determinar el tipo del fichero. Algunos sistemas de ficheros consideran a la extensión como una parte del nombre (y, de hecho, admiten que un mismo fichero posea varias extensiones anidadas), y otros la diferencian del nombre a nivel interno.

De este modo, aunque el sistema operativo no conozca internamente la estructura de los ficheros, si es capaz de manejarlos eficientemente gracias al uso de estas extensiones que son asociadas a los programas que manejan ese tipo de información.

Ésta es la aproximación de los sistemas operativos de Microsoft. Unix y sus variantes (Linux) sin embargo, optan por la no utilización de extensiones, lo que implica que el usuario es el único encargado de saber lo que se puede realizar o no con un fichero dado.

- Archivos ejecutables (.exe, .com, .bat shell scripts linux, etc)
- Documentos (.doc, .docx, .xls, .dbf, .jpg, .avi, .mp3, etc.)
- Librerías de ejecutables (.dll)
- Etc.

3.2. LOS DIRECTORIOS

Son una división lógica de almacenamiento de archivos u otros directorios. El usuario percibe que todos los archivos relacionados con un determinado tema están en un mismo lugar físico pero, en realidad, los archivos están desperdigados por cualquier parte de la unidad de almacenamiento.

En todos los sistemas de archivos existe un directorio especial llamado **raíz**, que es el directorio principal de la unidad de almacenamiento.

Todos los otros directorios (subdirectorios) son metadatos que copian la estructura del directorio raíz.

Los archivos individuales pueden colocarse en cualquier lugar de la estructura arborescente de directorios por parte del usuario. Sin embargo, solemos colocar juntos en un mismo directorio los archivos que están relacionados entre si, como si ordenásemos libros en una biblioteca.

Los nombres de los directorios pueden tener extensión al igual que el nombre de archivo y es recomendable que sea lo más descriptivo posible de los archivos que contiene, junto con el nombre

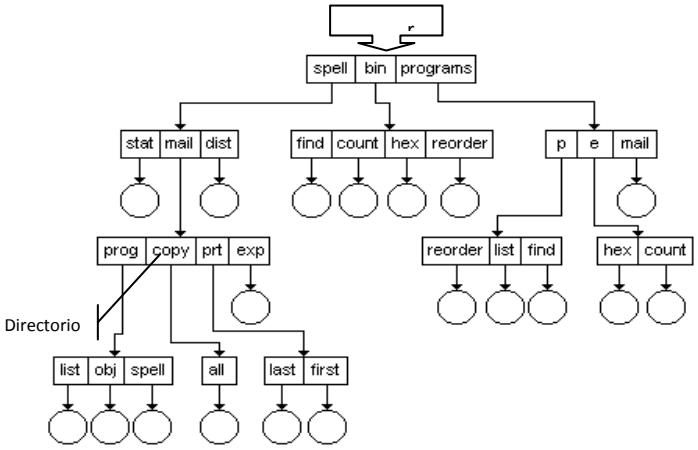
del directorio el sistema operativo almacena también unos atributos que califican al directorio, varían de un sistema a otro y entre ellos destacan:

- H: Atributo de oculto
- R: Atributo de solo lectura
- A: Atributo de archivo
- Fecha
- Hora

3.2.1 ESTRUCTURAS DE DIRECTORIOS.

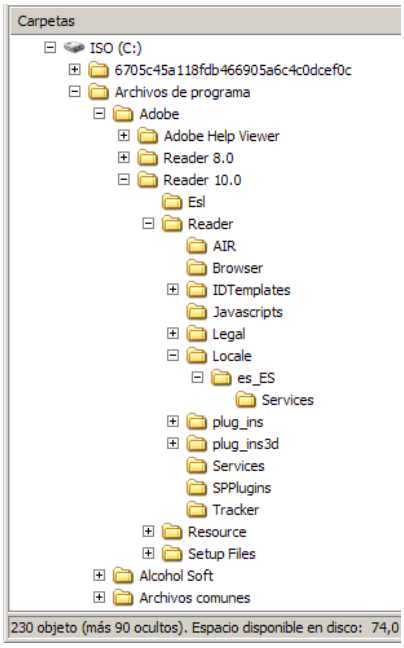
El usuario percibe la estructura de directorios como un árbol. Esta estructura arborescente prácticamente no tiene limitaciones. Un directorio puede incluir otros directorios, sin importar su número, y estos nuevos directorios pueden contener otros directorios

El árbol es de raíz única, de modo que cada fichero tiene un único nombre de ruta de acceso. Un directorio (o subdirectorio) contiene a su vez ficheros y/o subdirectorios, y todos los directorios poseen el mismo formato interno. Las entradas del directorio indican si el objeto referenciado es un fichero o un subdirectorio.



Se define directorio padre de un fichero o subdirectorio como el directorio en el que se encuentra su entrada de referencia Cada fichero o directorio (a excepción del directorio raíz) posee un único directorio padre. Así, *spell* es el padre de *mail* que, a su vez, es el padre de *prog*. Todos los directorios, salvo el raíz, tienen una entrada . . que apunta a su directorio padre.

Se define directorio hijo de un directorio como el directorio que tiene por padre al primero. Un directorio puede contener múltiples directorios hijos, y cada directorio (a excepción del raíz) es hijo de algún otro (*proa* es el hijo de *mail* que a su vez es hijo de *spell*).



Se define directorio actual o directorio activo como aquel el usuario está situado y, por defecto, actuaran las órdenes de manipulación de archivos. Todos los directorios tienen una entrada . que apunta a si mismo.

En este caso, los nombres de ruta de acceso pueden adoptar dos formas alternativas. La primera es la del nombre de ruta absoluta, por la cual se nombra a cada fichero partiendo del directorio raíz (*spell**mail**copy**all*).

La segunda opción es la del nombre de ruta relativa. En este caso, se nombra al fichero con respecto al directorio actual (siendo *copy* el directorio actual, la ruta relativa del fichero del párrafo anterior sería *all*).

3.2.2 SISTEMAS DE ARCHIVOS

Es necesario que el sistema operativo cuente con un sistema que se encargue de administrar la información almacenada en los dispositivos en forma de ficheros: el sistema de archivos.

Un sistema de ficheros es el aspecto más visible de todo sistema operativo y existe por razones tecnológicas, ya que no hay memoria principal lo suficientemente grande como para no necesitar de almacenamiento secundario. El sistema operativo ofrece una visión lógica y uniforme del almacenamiento de información realizando una abstracción de las propiedades físicas de sus dispositivos de almacenamiento

Los objetivos principales de todo sistema de archivos deben ser los siguientes:

1. Crear, borrar y modificar ficheros.
2. Permitir el acceso controlado a la información.
3. Permitir intercambiar datos entre ficheros.
4. Poder efectuar copias de seguridad recuperables.
5. Permitir el acceso a los ficheros mediante nombres simbólicos.

Hay otros objetivos secundarios, entre los que destacan:

1. Optimizar el rendimiento.
2. Tener soportes diversos para E/S (para poder seguir utilizando los mismos ficheros aunque cambie el soporte).
3. Ofrecer soporte multiusuario.
4. Minimizar las pérdidas de información.

3.3. IMPLEMENTACIÓN DEL SISTEMA DE ARCHIVOS

El aspecto clave de la implementación del almacenamiento de archivos es el registro de cada uno de los bloques de datos asociados a un fichero.

Windows utiliza el sistema FAT para gestionarlo, o MFT si se utiliza formato NTFS.

Linux utiliza un sistema de ficheros basado en i-nodos. En esta técnica, se asocia a cada archivo un metadato llamado i-nodo, que contiene la información sobre el archivo y direcciones en disco de los bloques de datos.

3.3.1 FORMATEO DE LOS DISCOS

El formateado, formateo o formateo a alto nivel consiste en definir las estructuras necesarias para el sistema de archivos que vamos a utilizar.

Así mismo, se define el tamaño de los clusters.

3.4. TIPOS DE SISTEMAS DE ARCHIVOS

3.4.1 SEGÚN SU FUNCIONAMIENTO

TRANSACCIONALES

Una transacción es un conjunto de operaciones en las que se ejecutan todas ellas o no se ejecuta ninguna. Las ordenes se envían todas una a una pero no se graban hasta el final, momento en el cual, por medio de una orden *commit*, se realiza la transacción. En caso de que una transacción no finalice bien o tenga algún fallo existe otra orden conocida como *rollback*, que deshace la transacción.

En un sistema de archivos transaccional el estado del sistema de archivos siempre debe ser coherente, por lo cual se implementa un concepto de registro de diario (), que guarda las acciones aparte para poder volverlas a reproducir si el sistema se bloquea.

En un sistema clásico la nueva versión de un archivo se sobreescribe físicamente sobre la vieja. En un sistema transaccional, las transacciones en disco se escriben de manera secuencial en un área llamada *journal* o *log*, antes de ser escritas en sus localizaciones finales dentro del sistema de archivo.

El *journaling* registra los cambios realizados a un sistema de archivos, facilitando y haciendo mucho más rápida la restauración de un sistema de archivos dañado (por ejemplo, tras un cierre repentino del sistema o un fallo del suministro eléctrico)

DISTRIBUIDOS

Permiten que los directorios localizados en cualquier parte de una red sean visualizados como un árbol de directorios único sin necesidad de que os usuarios sepan en que servidor reside cada archivo, se le denomina también sistema de archivos de red.

Ventajas:

- Un usuario puede acceder a sus archivos desde diferentes máquinas.
- Usuarios diferentes desde diferentes máquinas pueden acceder a los mismos archivos.
- Cada usuario y/o servidor puede usar un sistema operativo distinto

CIFRADOS

Un sistema de archivos cifrado es aquel que permite a los usuarios almacenar sus datos en el disco de forma cifrada.

El cifrado es el proceso de conversión de los datos a un formato que no puede ser leído por otro usuario, cuando un archivo se cifra permanece así en el disco. El cifrado de archivos se puede realizar con cualquier algoritmo de cifrado.

El descifrado es el proceso de reconversión de los datos a su formato original.

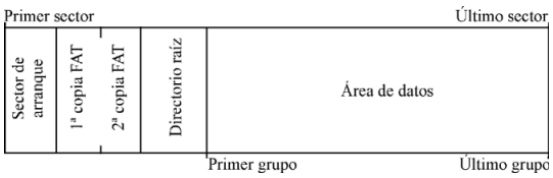
El cifrado y descifrado de archivos se realiza para cada archivo o para toda una carpeta, incluidas todas las subcarpetas. Todos los objetos creados en una carpeta que están marcados para cifrado se cifran automáticamente. Cada archivo tiene su propia clave de cifrado única, lo que permite cambiar el nombre del archivo de forma segura. Si al cambiar el nombre de un archivo, éste pasa de una carpeta cifrada a una no cifrada del mismo volumen, el archivo permanecerá cifrado. Sin embargo, si copia un archivo no cifrado a una carpeta cifrada, cambiará el estado del archivo. En ese caso, el archivo sigue estando cifrado. Se proporcionan herramientas de línea de comandos e interfaces administrativas para usuarios avanzados y agentes de recuperación.

3.4.2 SEGÚN SU ESTRUCTURA

FAT

El disco se divide en tres áreas

1. La FAT (*File Allocation Table*) tabla de localización de ficheros.
2. Copia de la FAT (opcional aunque casi siempre presente)
3. El DIRECTORIO
4. Área de datos



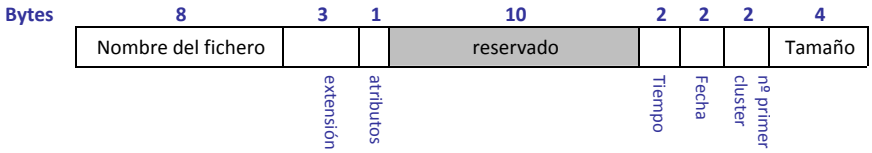
La FAT es una matriz unidimensional que contiene tantas celdas como cluster tenga la partición. El tamaño de cada celda es de 16 bits (FAT16) o 32 bits (FAT32).

Cada celda contiene un valor que indica si el cluster al que referencia esta libre, no utilizable (es aquel que contiene al menos un sector defectuoso) u ocupado. En caso de estar ocupado, contendrá la señal de fin de fichero (EOF) o, si el archivo ocupa más de un cluster, el número del cluster en donde continúan los datos del archivo.

LA FAT se carga en memoria al arrancar y es periódicamente reescrita en el disco. Si el sistema se apaga bruscamente, pueden quedar clusters con contenido no referenciados en la FAT:

El Directorio, es un fichero especial de un único cluster de tamaño en FAT16 aunque en FAT32 puede tener un tamaño variable, y ser una cadena de clusters, como cualquier otro directorio.

En FAT16 cada entrada del directorio contiene 32 Bytes repartidos así:



En FAT32, usando LFNs (Long File Names):

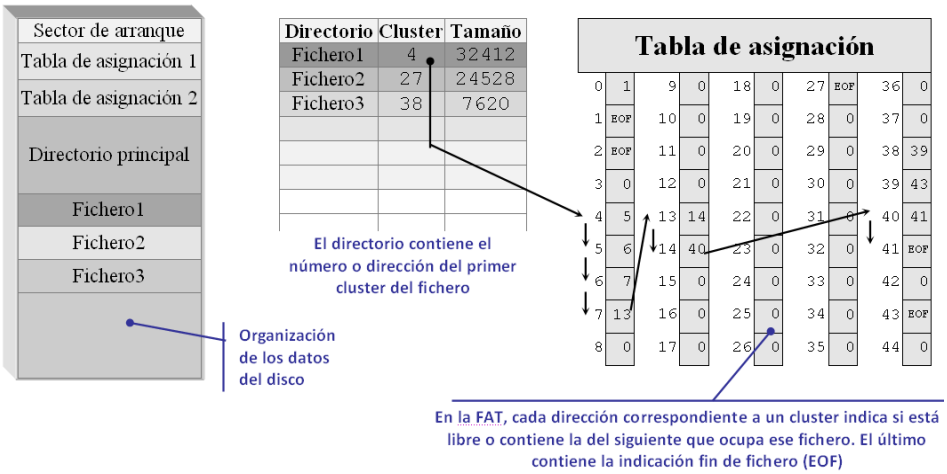
Bytes 1	10	1	1	1	12	2	4
Para nombres largos	Nombre del fichero	atributos	Continuidad nombre largo	Comprobación nombre corto	continuación nombre largo	0	Tamaño

Los nombres largos se almacenan ocupando varias entradas en el índice para el mismo archivo o carpeta.

Cuando el Directorio ocupa un único cluster, el número de entradas que puede tener es limitado. Cuando se crea un subdirectorio, el sistema crea una copia de la estructura del Directorio en un nuevo cluster. Los subdirectorios pueden tener entradas ilimitadas ya que si se completa el primer cluster asignado pueden continuar en un nuevo cluster.

Cuando grabamos un fichero, el sistema busca los primeros clusters libres, normalmente a partir de la posición actual del cabezal de escritura. Si encuentra clusters ocupados una vez comienza la escritura, los salta y busca los siguientes disponibles. Cuando esto ocurre se dice que el fichero está fragmentado. Un exceso de fragmentación ralentiza el sistema, por los que es aconsejable desfragmentar el disco periódicamente. Por último, escribe la entrada del archivo en el directorio correspondiente.

Para acceder a un archivo, el sistema busca en el directorio la entrada correspondiente y toma nota del cluster inicial. A continuación, busca en la FAT la celda correspondiente a ese cluster y confecciona una lista de los clusters que ocupa el archivo.



Como vemos en el ejemplo el `Fichero1` ocupa los clusters 4 a 7, salta al 13 y 14 y de ahí al 40, terminando en el 41.

Cuando borramos un fichero, se pone un carácter especial a su nombre en el directorio y las direcciones de los clusters que ocupaba se ponen a 0, es decir, disponibles. Por ello, tras borrar varios ficheros, los próximos que grabemos ocuparán los clusters con 0 y pueden quedar fragmentados. No tiene mayor importancia, pero el acceso a un fichero muy fragmentado puede hacerse mucho más largo.

Por ello, es conveniente pasar un defragmentador al HD periódicamente ya que la contigüidad de los clusters de un mismo fichero acelera los tiempos de acceso. Además, la defragmentación compacta todos los datos eliminando los huecos y elimina físicamente los ficheros borrados.

Límites	**FAT12**	**FAT16**	**FAT32**
Tamaño máximo del volumen	2 MB	2 GB	2 TB
Tamaño máximo de archivo	32 MB	2 GB	4 GB
Número máximo de archivos	077	65.517	268.435.437
Nombres *case sensitive*	No		
Longitud máxima del nombre de archivo	8.3 (11) o 255 caracteres cuando se usan LFNs (Long File Names)		

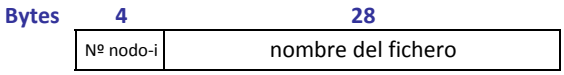
LINUX EXT3

Es uno de los mas eficientes y flexibles, se usa básicamente en Linux, el tamaño máximo de volumen es de 32TB, y el de archivo es de 2T, distingue entre mayúsculas y minúsculas y dispone de un registro diario (*journaling*) que permite almacenar la información necesaria para restablecer los datos afectados por una transacción en caso de que esta falle.

Tiene las siguientes estructuras

- Tabla mapa de cluster libres: 1 bit por cluster
- Directorio principal (Nodo-i y Estructura del directorio)

La estructura de los directorios de UNIX es extremadamente simple. Cada entrada contiene un nombre de fichero y el número de su nodo-i. Toda la información sobre el tipo, tamaño, tiempos, propietario y bloques en disco del fichero está contenida en su nodo-i. Todos los directorios en UNIX son ficheros y pueden contener un número arbitrario de estas entradas de 16 bytes.



Cada fichero UNIX tiene una pequeña tabla asociada (en disco), llamada nodo-i que contiene, entre otras informaciones, los 10 primeros números de cluster donde se encuentran los datos del archivo en el disco y 3 números de bloque indirecto. Para ficheros de longitud menor o igual a 10 cluster, todas las direcciones de los cluster se guardan en el nodo-i, lo que facilita su localización.

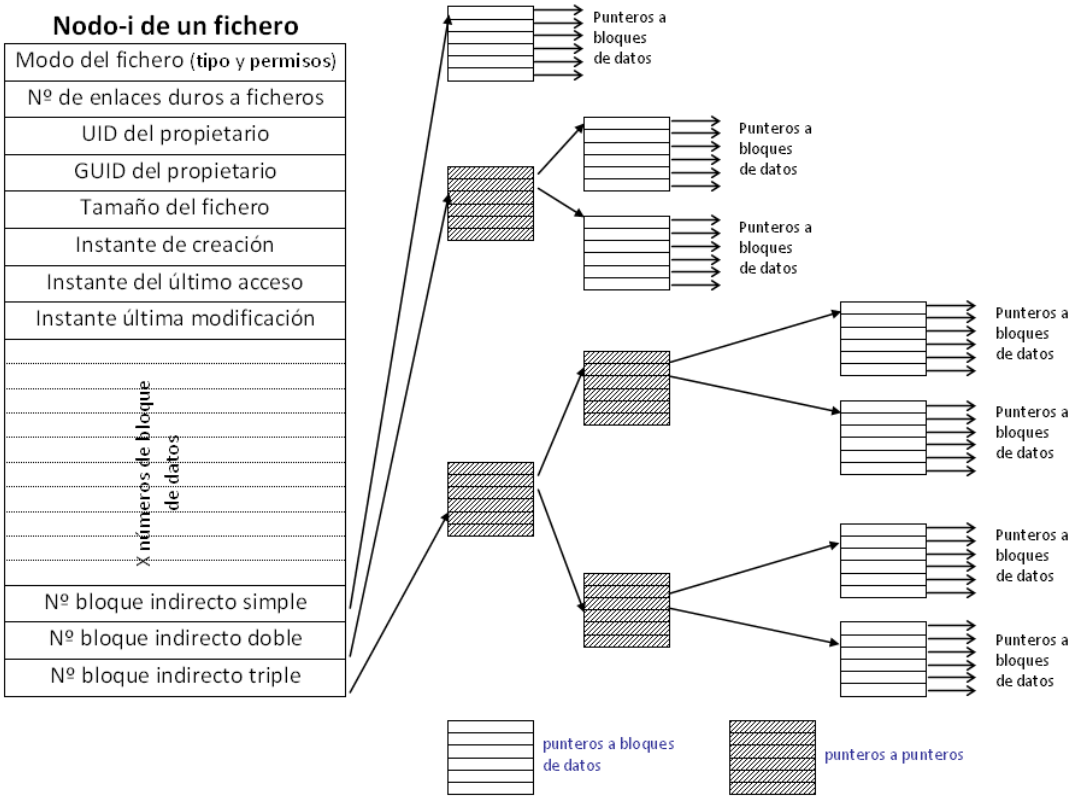
Cuando un fichero ocupa más de 10 cluster de disco, se adquiere un cluster libre y se pone el puntero indirecto simple apuntándole. Este cluster se usa para contener punteros a cluster de disco. (n punteros)

Si el tamaño del archivo es superior a los n clusters que permite direccionar puntero indirecto simple, se emplea el puntero indirecto doble, que apunta a un cluster de disco de punteros que apuntan a punteros indirectos simples en vez de hacerlo a clusters de datos. El puntero indirecto doble cubre ficheros con hasta $10+n+n^2$ clusters.

Para ficheros de mayor tamaño se emplea el puntero indirecto triple, que apunta a un cluster que contiene punteros doble a punteros indirectos dobles, lo que supone un máximo de $10+n+n^2+n^3$ clusters.

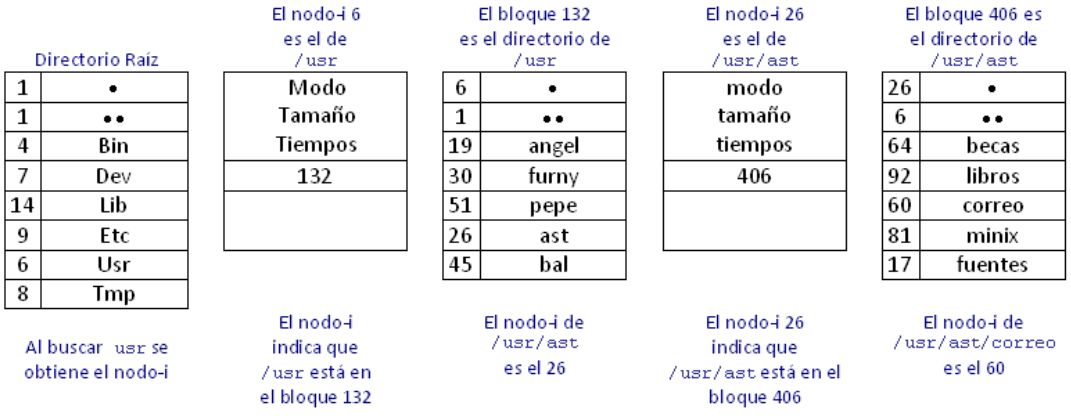
La potencia del sistema UNIX reside en que los punteros indirectos sólo se utilizan cuando hacen falta. Incluso para los ficheros más grandes son necesarias sólo tres referencias a disco para localizar la dirección de cualquier byte del fichero (se excluye la referencia de lectura del nodo-i, que se lee al abrir el fichero y se deja en memoria hasta que se cierra).

Para controlar los clusters libres, posee una tabla-mapa de clusters, con 1 bit por cluster, que indica si está libre u ocupado. Como en FAT, dispone de archivos estructurados especiales para los directorios, con la salvedad de que el directorio principal no está necesariamente ubicado en una posición predeterminada.



Estructura de un nodo-i en UNIX.

Al abrir un fichero, el sistema de ficheros debe analizar el nombre proporcionado por el usuario y localizar sus bloques en disco. Consideremos la búsqueda del fichero con nombre de ruta */usr/ast/correo*. Primero, el sistema de ficheros localiza el directorio raíz. En Linux, el nodo-i de la raíz se encuentra en un lugar fijo en el disco.



A continuación se busca en el directorio raíz (que está en un lugar fijo del disco) el primer elemento de la ruta, *usr*, para encontrar el nodo-i del fichero */usr*. A partir de este nodo-i, el sistema localiza el directorio */usr* y busca en él el siguiente elemento, *ast*. Una vez encontrada la entrada de

ast, se tiene el nodo-i del directorio */usr/ast*, a partir del cual se encuentra el directorio y se busca *correo*. Por último, se lee en memoria el nodo-i de este fichero, donde permanecerá hasta que se cierre dicho fichero.

Los nombres de ruta relativos se recorren como los absolutos, pero empezando en el directorio de trabajo en lugar de en el raíz.

Al crear el direcctorio se crean la entrada `.` que contiene el número del nodo-i del directorio actual y la `..` el número del nodo-i del directorio padre.

NTFS

NTFS (de *NT File System, New Technology File System*) es un sistema adecuado para las particiones de gran tamaño. Su principal inconveniente es que necesita para sí mismo una buena cantidad de espacio en HD, por lo que no es recomendable su uso en discos con menos de 400 MB.

Implementa cuotas de disco, compresión de archivos, cifrado, puntos de montaje de volúmenes, etc. Sus volúmenes son de hasta 16TB, y el tamaño máximo de archivo está limitado por el de volumen. Distingue entre mayúsculas y minúsculas (*case sensitive*) en los nombres de archivo

Los nombres de archivo son almacenados en Unicode (UTF-16), y la estructura de ficheros en árboles-B, una estructura que acelera el acceso a los ficheros y reduce la fragmentación.

Se emplea un registro transaccional (*journaling*) para garantizar la integridad del sistema de ficheros (pero no la de cada archivo). NTFS ha demostrado tener una estabilidad mejorada.

Este sistema de archivos posee un funcionamiento prácticamente secreto, ya que Microsoft no ha liberado su código como hizo con FAT,

Cada archivo en un volumen NTFS está representado por un registro en un archivo especial llamado tabla de fichero maestro (MFT: *Master File Table*) que es una base de datos donde se almacena la información de los archivos. NTFS reserva los 16 primeros registros de la tabla para información especial. El primer registro describe la propia MFT, seguido por un registro espejo (otras copia de la MFT). Si el primer registro MFT es erróneo, NTFS lee el segundo registro para encontrar el archivo espejo, cuyo contenido es idéntico al del primer registro. El tercer registro contiene el archivo de registro. Este es un archivo que contiene todas las acciones llevadas a cabo en la partición.

EL contenido en la MFT sobre un archivo es similar al contenido de un nodo- del sistema ext3.

Los archivos y directorios pequeños (normalmente menos de 1500 bytes) pueden ser ubicados directamente dentro de la MFT.

OTROS SISTEMAS DE ARCHIVO

MFS (para Mac OS)
ext2 (Para Kernel Linux)
ext4 (Para Kernel Linux)
UMSDOS (Linux sobre FAT)
HFS (para Mac OS)
HFS+ (para Mac OS X)
HPFS
MINIX FS
JFS (Journaling File System)
ISO 9660 (de solo lectura, para CD-ROM)
OFS (Object File System)
ReiserFS (Soportado por Linux)
Reiser4 (Disponible en Kernel Linux)
UDF (usado en DVD y en algunos CD-ROM)
ZFS (sistema de archivos de Sun Microsystems)
Btrfs (De Oracle Corporation para GNU/Linux)
WinFS (Windows File System, se planeaba su incorporación en Windows Vista y más tarde en Windows 7, pero nunca se concretó)