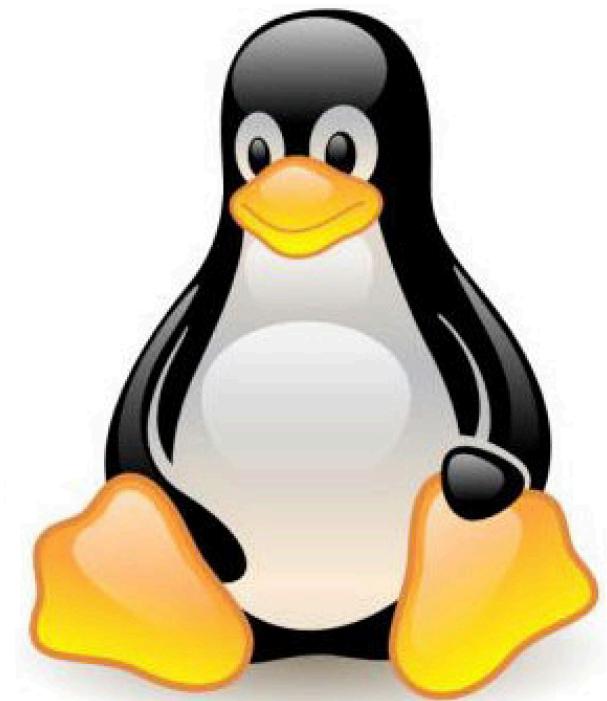
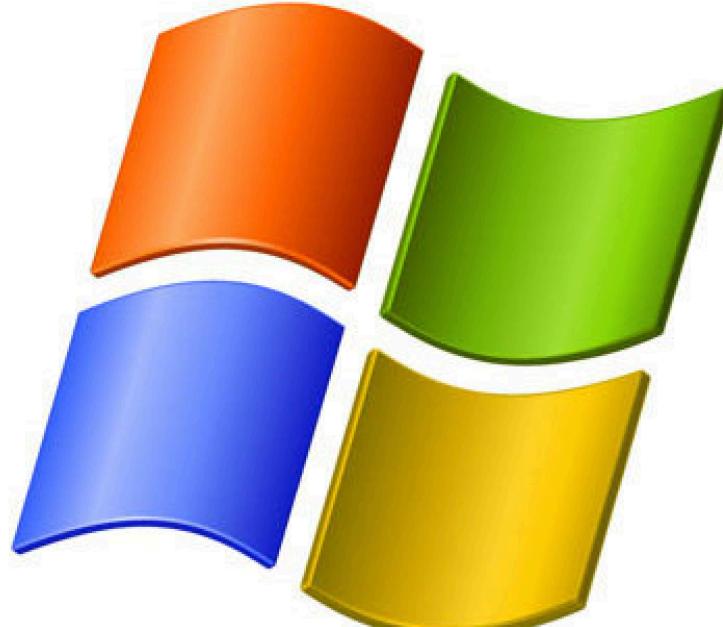


Tema 4: Sistemas de ficheros

Sistemas Operativos

ETSI Informática. Universidad de Málaga



Manuel Ujaldón

Catedrático de Arquitectura de Computadores
Departamento de Arquitectura de Computadores
Universidad de Málaga

Índice [30 diapositivas]

I. Introducción. [3]

1. Concepto, tipos y atributos.

II. Tecnologías de almacenamiento masivo. [7]

1. Discos magnéticos: Estructura y rendimiento.
2. Discos de estado sólido (SSD).

III. Implementaciones del sistema de ficheros. [5]

1. Volúmenes, particiones, ficheros y directorios.
2. Estructuras de datos en disco y memoria.

IV. Métodos de alojamiento. [9]

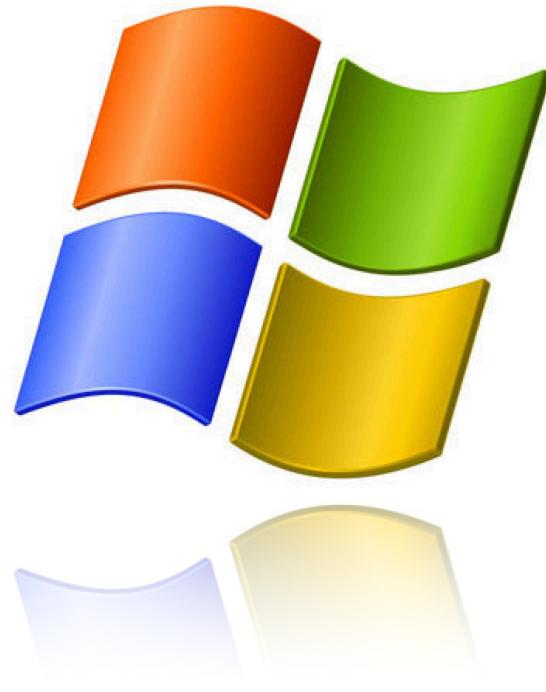
1. Contiguos, enlazados (FAT) e indexados (i-nodos).

V. Directorios. [3]

1. Concepto, implementación y estructuras de datos.

VI. Otras mejoras. [3]

1. Gestión del espacio libre.
2. Elección del tamaño del cluster.
3. Mejoras de rendimiento y consistencia.



I. Introducción

Fundamentos

- Un sistema de ficheros es la capa del Sistema Operativo que gestiona el espacio de dispositivos de bloques como los discos, organizándolos en volúmenes, particiones, directorios, ficheros, ...
- ¿Qué aporta un sistema de ficheros?
 - **Almacenamiento** de cantidades ingentes de datos.
 - **Persistencia** de la información cuando finaliza el proceso que la creó.
 - **Intercambio** de información entre usuarios.
 - **Acceso concurrente** a la información desde múltiples procesos.
 - **Fiabilidad** de la información almacenada y tolerancia a fallos.
 - Niveles de **seguridad** para proteger la información.

Fichero

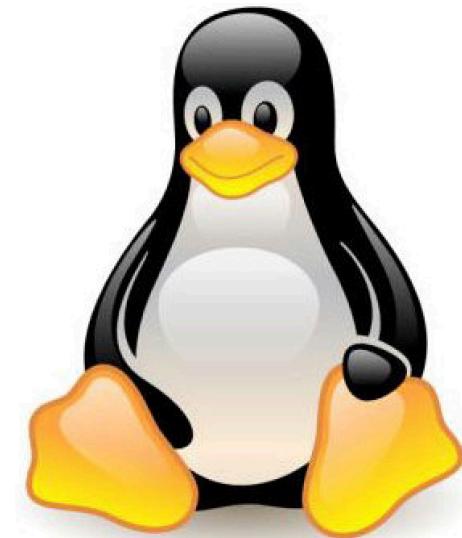
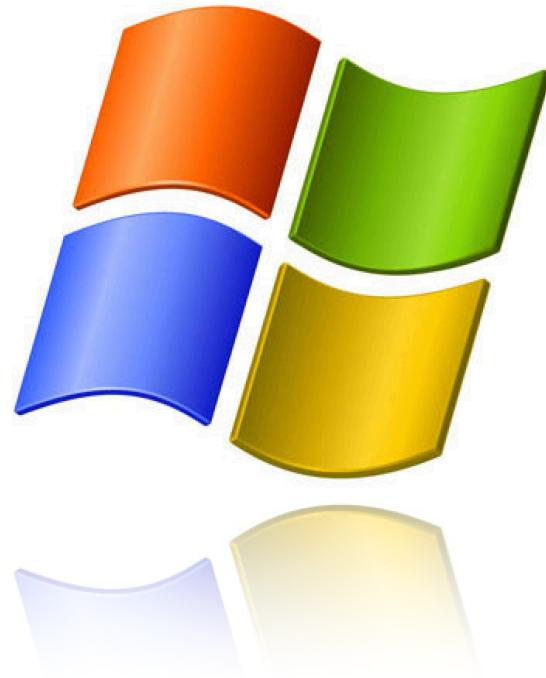
- Concepto: Secuencia de bytes alojados en disco y visibles al usuario, organizados en un espacio lógico secuencial.
- Tipos: El formato o la estructura impuesta de un fichero suele venir reflejado en su extensión o sufijo. Ejemplos:

Extensión	Significado
.bak	Copia de seguridad o <i>backup</i>
.c	Programa fuente en lenguaje C
.o	Fichero objeto (salida del compilador, aún no enlazado)
.gif	Imagen en formato GIF (Graphical Interchange Format)
.html	Página Web en formato HTML (HyperText Markup Language)
.pdf	Documento en formato PDF (Portable Document Format)
.zip	Fichero comprimido
.txt	Fichero de texto

Atributos del fichero

- Constituyen información adicional que el sistema operativo mantiene para cada fichero: **Metadatos** (datos acerca de los datos). Ejemplos:

Atributo	Significado
Protección	Quién tiene permisos para acceder a un fichero y de qué manera
Propietario	Usuario al que pertenece el fichero
Tamaño	Número de bytes que contiene
Fecha de creación	Día y hora en que fue creado el fichero
Último acceso	Día y hora en que fue accedido por última vez
Último modificación	Día y hora en que cambiaron sus contenidos por última vez



II. Tecnologías de almacenamiento masivo

Tecnología magnética

- Es la del disco duro tradicional.

- Modelos más populares:

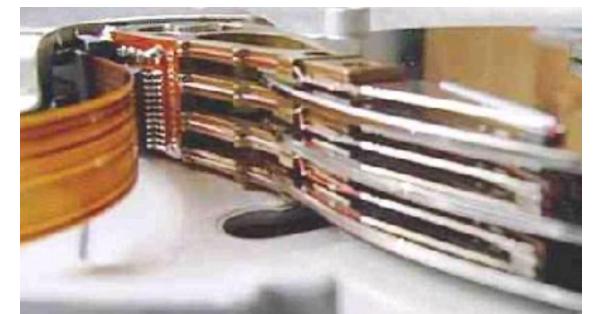
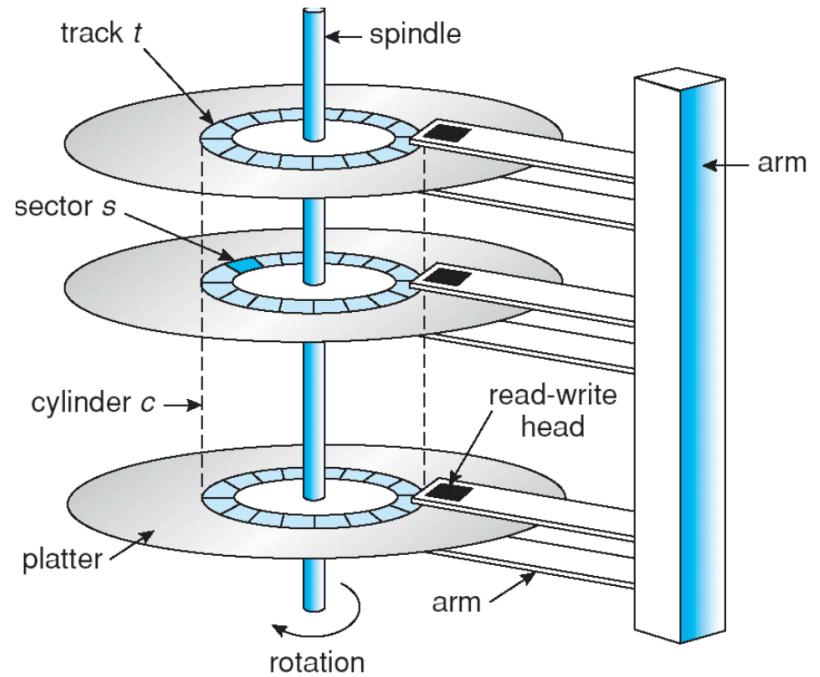
- Barracuda de Seagate.
- Blue/Green de Western Digital (WD).

- La información se organiza en tres planos:

- Vertical: Cilindros y platos (*platters*).
- Horizontal: Pistas (*tracks*).
- Circular: Sectores (de 4 Kbytes en UNIX).

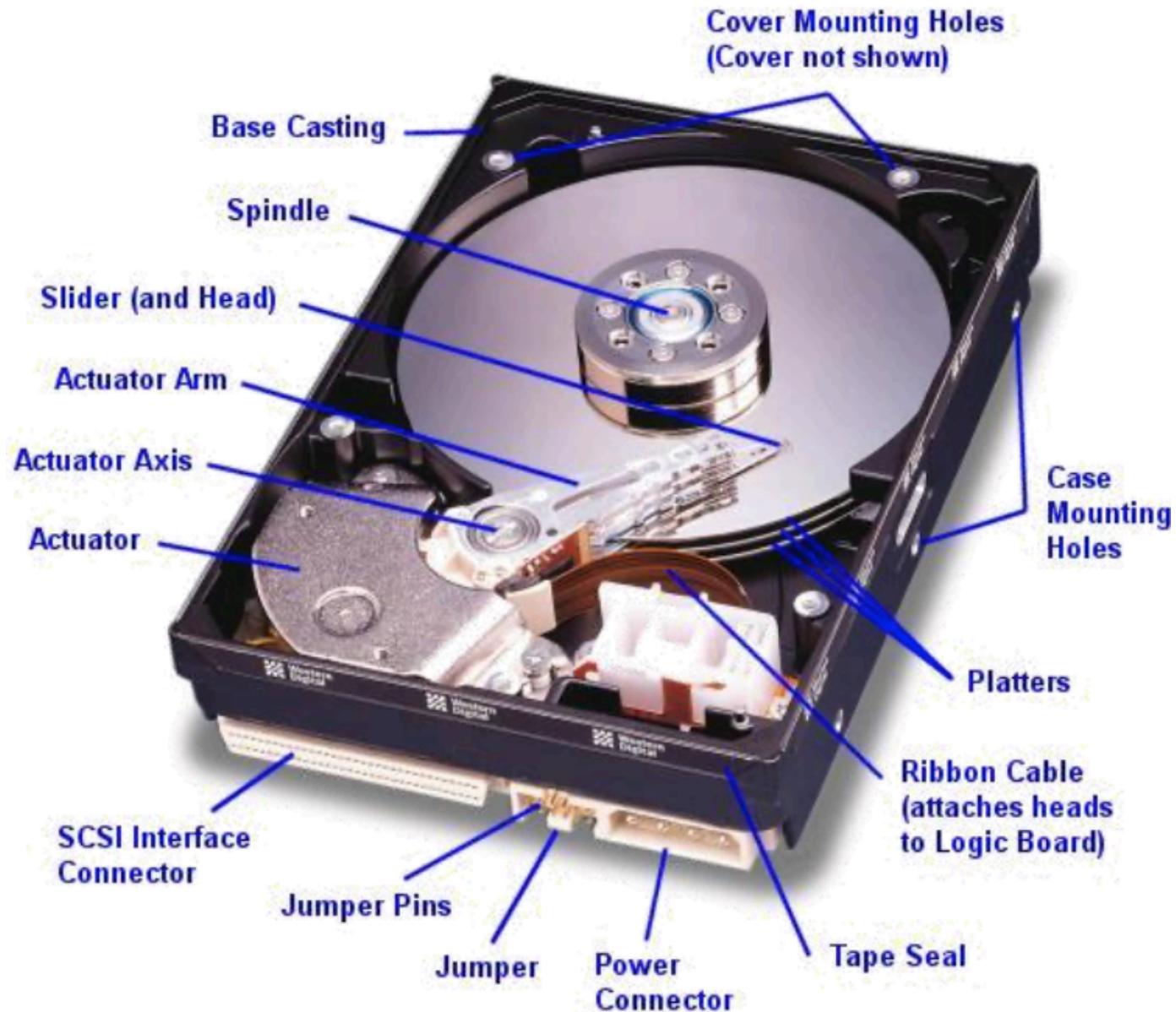
- La unidad mínima de almacenamiento es el *cluster*, que agrupa sectores para amortizar latencia con el ancho de banda.

Para 3 platos y 16 sectores/pista:





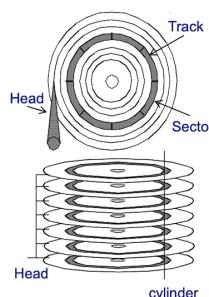
Detalles internos de un disco duro SCSI de Western Digital



Parámetros de funcionalidad y rendimiento

- Capacidad en Terabytes (2^{40} bytes): Aprox. 1 billón de bytes.
- Latencia: Tiempo de acceso al primer sector.

● Principal causa de su lentitud (1-5 msg.) por el posicionamiento mecánico. Vista la geometría del disco, sumaríamos 3 tiempos:



- Latencia vertical. El brazo mueve 2 cabezales por plato (envés/revés) a la vez, por lo que no necesita desplazarse verticalmente, ahorrando este tiempo.
- Latencia horizontal (seek time). Tiempo para ubicarse en la pista circular.
- Latencia rotacional (rotational latency). Tiempo para que el primer sector pase bajo el cabezal. Para un disco de 10.000 RPM, será un máximo de:
 - El disco da 10.000 vueltas cada 60 sg. → 166.66 vueltas/sg. → tarda 1/166.66 sg. en completar una vuelta, que son 6 msg. de latencia rotacional máxima (y 3 msg. de media).

- Ancho de banda (*transfer rate*): Es el ritmo al que envía datos al PC. En la memoria lo veíamos como *bandwidth*.

- Si rota a 10.000 RPM y posee 128 sectores de 8 KB. en cada pista, el disco recorre una pista de $128 \times 8\text{KB} = 1\text{MB}$ en 6 milisegundos: $1\text{ MB} / 6\text{ msg} = 166.66\text{ MB/sg}$.

Análisis de rendimiento del disco

- Para leer o escribir en el disco, entran en juego 3 tiempos:
 - Latencia horizontal (seek time), para el posicionamiento en la pista.
 - Latencia rotacional (rotational latency), o espera hasta que el sector deseado pase bajo el cabezal.
 - Tiempo de transferencia, en función del ancho de banda del disco.
- Ejemplo: Para el disco anterior de 10.000 RPM, sectores de 8 KB y 128 sectores por pista, donde ya calculamos:
 - Latencia rotacional media (3 msg.) y ancho de banda (166 MB/sg.).
 - Si tenemos un *seek time* típico de 5 msg. y queremos leer un fichero de 1 MB cuyos sectores se guardan todos consecutivos en disco, tardaríamos:
 - $\text{Time} = \text{Seek} + \text{Rotational} + \text{Transfer} = 5 \text{ msg.} + 3 \text{ msg.} + (1 \text{ MB} / 166 \text{ MB/sg.}) = 5 \text{ msg.} + 3 \text{ msg.} + 0.006 \text{ sg.} = 14 \text{ msg.}$

Direccionamiento a disco

- El sector 0 corresponde a la pista más externa del cabezal más superior. A partir de ahí, para posicionarnos en un sector del disco son necesarias 3 coordenadas:
 - Número de **cabezal**: Hay dos cabezales por plato y entre 2 y 4 platos, para un máximo de 8 cabezales en total.
 - Número de **cilindro**: Hay miles de ellos, y si el cabezal ya viene dado, en realidad este número determina la pista.
 - Número de **sector**: Suele haber 64-128 sectores por pista. Se utiliza un factor de compensación para equilibrar pistas externas e internas.
- Ejemplo: Calcular la capacidad de un disco que tiene 8 cabezales, 51.200 cilindros, 128 sectores/pista y sectores de 8 Kbytes.
 - Capacidad = $8 \times 51.200 \times 128 \times 8 \text{ Kbytes} = 400 \text{ Gbytes}$.

Otras propiedades del disco

- Es preciso distinguir entre el **controlador** del host que dialoga por el bus con el controlador hardware del disco, y el controlador software o *driver* del disco, que se encarga de configurar el dispositivo y transmitir las instrucciones al HW.
- La **conexión** del disco al host puede ser:
 - Para servidores: Fibre Channel ha reemplazado a SCSI.
 - Para PCs: USB y SATA-1,2,3 han reemplazado a EIDE.
- El **formato** determina el área que ocupa, y puede ser:
 - 5.25 pulgadas para los discos más viejos.
 - 3.5 pulgadas para los discos más recientes.
 - 2.5 pulgadas para los discos de los equipos portátiles.
- El disco posee una **caché** o *buffer* interno de unos 64 MB.

Tecnología de estado sólido

- Basada en matrices de celdas, al igual que DRAM y Flash.

- Frente a la tecnología magnética:

- +++ Mucho más fiable, al no tener:

- Partes móviles.
 - Componentes mecánicos.
 - Estrés térmico.

- ++ Latencia: 100 veces inferior (10^{-5} sg. vs. 10^{-3} sg.).

- + Ancho de banda: El doble (500 MB/sg. vs. 250 MB/sg.).

- + Más pequeño y robusto.

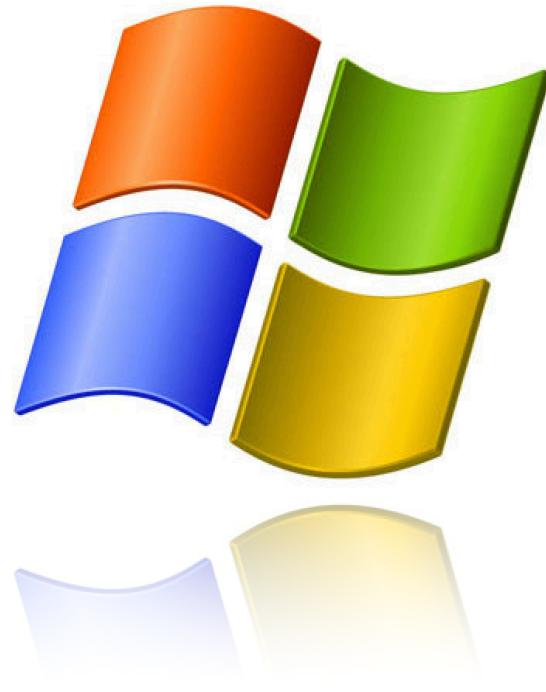
- + Menor consumo.

- Sus celdas pueden reescribirse un número limitado de veces.

- Precio medio más elevado (aprox. 0.1€/GB. vs. 0.05€/GB.).

- Fabricantes más populares: Kingston, Samsung, Toshiba y Crucial. En general, la mayoría de fabricantes de DRAM.





III. Implementaciones del sistema de ficheros

La visión lógica del Sistema Operativo

- El controlador hardware y software del disco se encargan de transmitir al Sistema Operativo una visión que oculta casi todos los detalles de su circuitería.
- El formateado del disco descompone su espacio útil en una serie continua de *clusters* direccionables en una sola dimensión (LBA - *Logical Block Addressing*), para que sea direccionable de forma similar a como vimos para la DRAM.
- El *cluster* puede agrupar varios sectores del disco, siendo el sector la unidad de trabajo para el controlador. Cada sector tiene una cabecera con metadatos, su área de datos y un campo final ECC para la corrección de errores.

Cabecera	Datos	ECC
----------	-------	-----

Organización del disco

- Un disco se divide en **particiones**, cada una de ellas formateada con un sistema de ficheros, que puede ser:
 - Linux (ext2, ext3, ext4, ...).
 - UNIX (UFS - *Unix File System*).
 - Windows:
 - FAT12, FAT16, FAT32, NTFS (a partir de Windows NT - *NT File System*).
 - MacOS: Tiene el suyo propio, HFS+, y además soporta FAT y lee NTFS.
- Cada entidad que alberga un sistema de ficheros (disco o partición) se denomina **volumen**. Estos volúmenes pueden estructurarse en RAID (*Redundant Array of Independent Disks*) para mejorar rendimiento (RAID 0) y protección (RAID 1) - otros números son esquemas intermedios.
- Las particiones contienen **ficheros** agrupados en **directorios**.

Estructuras de datos en disco

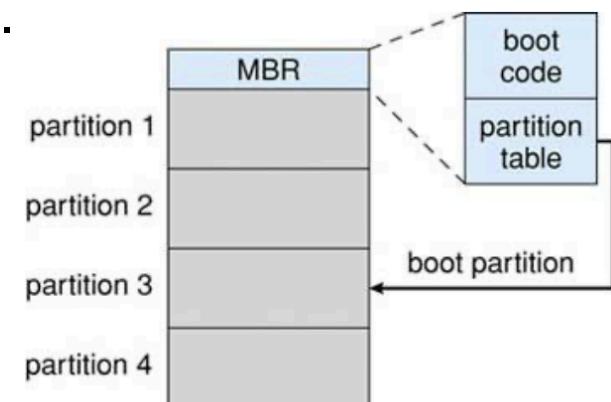
• EL S.O. habilita un bloque de control de fichero (FCB) para gestionarlo y registrar sus *clusters* en disco:

- i-nodos en UNIX.
- Master File Table en NTFS y FAT en sus precursores.

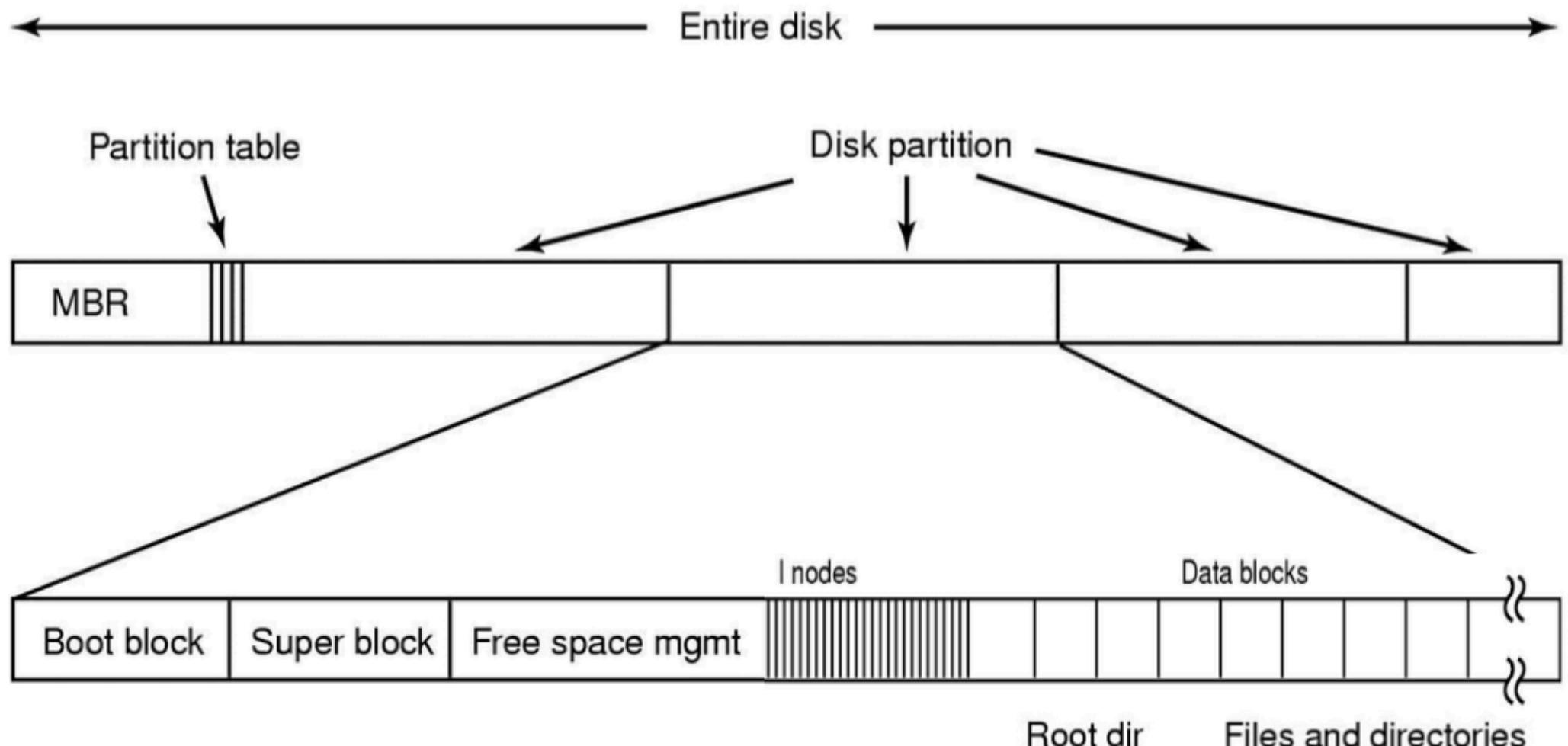
• También se registran los *clusters* libres para poder asignarlos a los ficheros.

• Para cada volumen de disco se habilita:

- Un bloque de control de arranque:
 - En el caso del PC se conoce como **MBR (Master Boot Record)**. Ejemplos: LILO, GRUB.
 - A través de la tabla de particiones, lanza pequeños programas, los bloques de arranque de cada una de las particiones, para inicializar las mismas.
- Un bloque de control de volumen:
 - En UNIX se llama **superbloque**, y en NTFS, forma parte del Master File Table (MFT).
 - Contiene metadatos, como el número de *clusters*, su tamaño, el espacio libre en disco y punteros a los FCB de los ficheros.

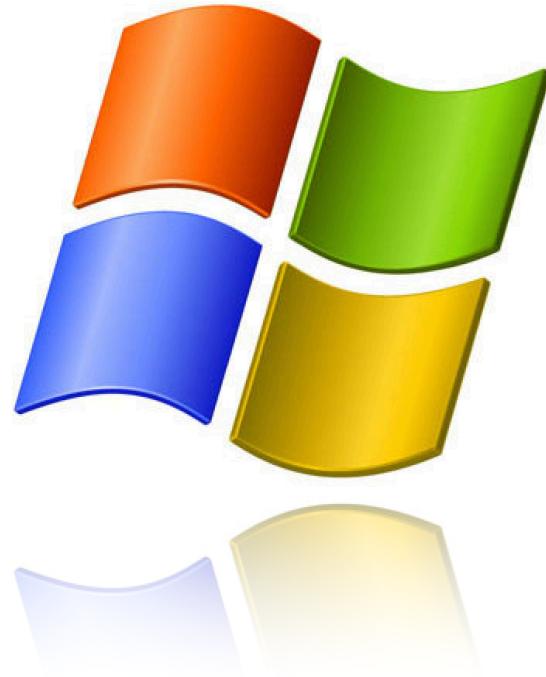


Visión global del sistema de ficheros UNIX



Estructuras de datos en memoria

- Cuando el S.O. monta un disco para usarlo, aloja en DRAM algunas estructuras de datos con un doble fin:
 - Llevar una gestión eficiente de las operaciones en disco.
 - Mejorar el rendimiento utilizando la DRAM como una caché (de ahí la obligación de avisar al S.O. para que lo desmonte antes de desconectarlo si no queremos tener inconsistencias en los datos).
- Estas estructuras de datos comprenden, entre otras:
 - Un tabla que registra cada volumen montado.
 - Una caché para los *clusters* más recientemente solicitados.
 - Tablas que registran los FCB de todos los ficheros abiertos:
 - Una tabla global a nivel de sistema y otra para cada proceso.
 - Copias de los FCB cuando se leen/escriben a disco.



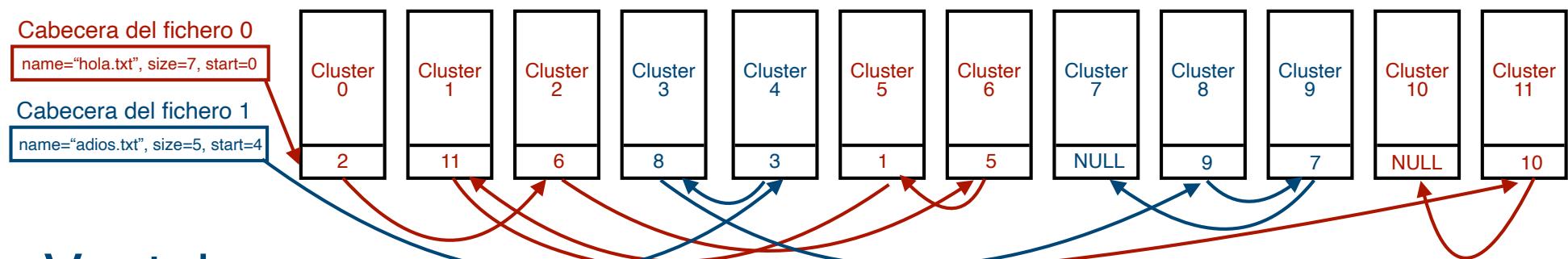
IV. Métodos de alojamiento

Concepto y estrategias

- El alojamiento de ficheros consiste en registrar los *clusters* que pertenecen a cada uno de ellos, de forma que se pueda:
 - Acceder aleatoriamente a la información que contienen.
 - Gestionar sus operaciones básicas: Creación, borrado, ampliación, ..
- Existen 3 métodos de alojamiento sucesivamente más complejos: Contiguos, enlazados e indexados.
 - Contiguos: Alojan los clusters de un fichero de forma consecutiva en el espacio de disco. Esto impide al fichero crecer tras su creación (o tener que realojarlo), lo que no es nada realista.
 - Enlazados: Se han utilizado desde los primeros PC bajo MS-DOS y sus sucesores de Microsoft (hasta Windows NT), siendo el ejemplo más extendido las FAT (File Allocated Table).
 - Indexados: Utilizados en UNIX desde sus orígenes.

Alojamiento enlazado

- Cada fichero es una lista enlazada de bloques de disco que se dispersan por el espacio de almacenamiento. Ej:



- Ventajas:

- Los ficheros pueden crecer dinámicamente.
- La lista de *clusters* libres puede gestionarse como un fichero más.

- Inconvenientes:

- El acceso secuencial requiere un seguimiento de la cadena, pero sobre todo, el acceso aleatorio es muy ineficiente.
- Es poco fiable: Si falla un enlace, se pierde el resto del fichero.

Ejemplo de alojamiento enlazado: FAT (File Allocated Table) de MS-DOS

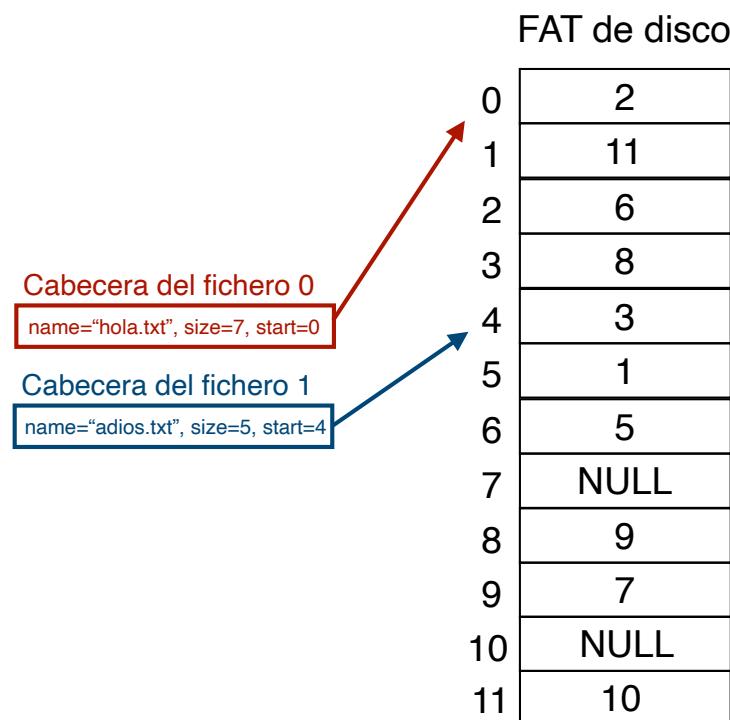
- Los enlaces no van en los *clusters*, sino que se agrupan todos en la FAT, una tabla aparte que puede alojarse en caché para agilizar consultas y actualizaciones.
- Para el ejemplo anterior, la FAT sería:

FAT de disco

0	2
1	11
2	6
3	8
4	3
5	1
6	5
7	NULL
8	9
9	7
10	NULL
11	10

Cabecera del fichero 0
name="hola.txt", size=7, start=0

Cabecera del fichero 1
name="adios.txt", size=5, start=4



Versiones de FAT

- **FAT12:** La inicial de 1977 para buses de dirs. de 12 bits.
 - Se usó para los primeros discos flexibles. Muy obsoleta.
- **FAT16:** La que empleó MS-DOS en 1987.
 - Extendió el tamaño del cluster a 32 y 64 Kbytes para los últimos S.O. que lo incorporaron (Windows XP, 2000 y NT).
- **FAT32:** La versión en uso actualmente.
 - Amplía a 32 bits la dirección de cada cluster para llegar a Terabytes.
 - En la práctica, Microsoft limitó a 28 bits y espacios más reducidos por limitaciones de su software.

Alojamiento indexado

- Para cada fichero se crea una tabla de punteros a los *clusters* que alojan su información en uso.

Ventajas:

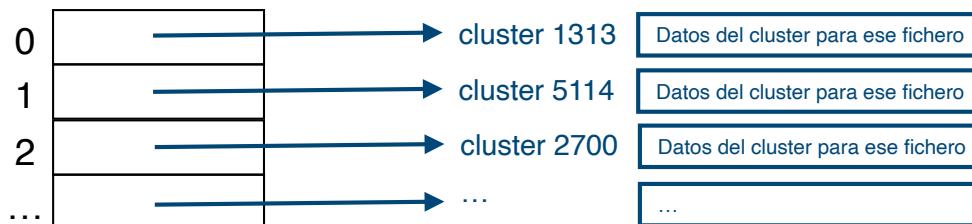
- El acceso aleatorio es muy eficiente.
- El fichero crece dinámicamente sin padecer fragmentación externa.

Inconvenientes:

- La longitud de la tabla define el máximo tamaño del fichero.
- Los *clusters* pueden estar diseminados por el disco, lo que requiere múltiples posicionamientos del brazo, con la consiguiente ineficiencia.

Visión lógica:

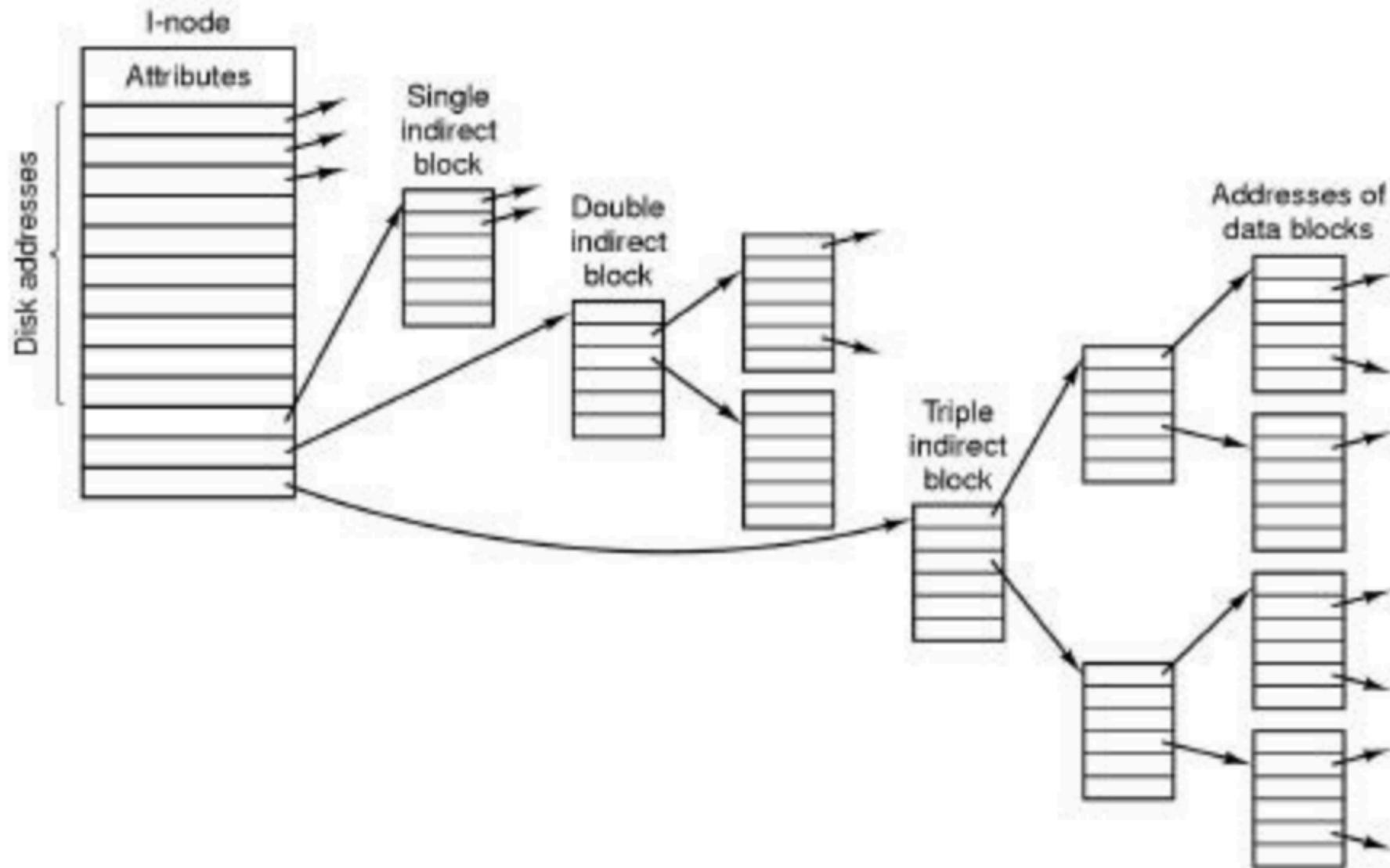
Tabla de punteros
en la cabecera
del fichero



i-nodos en UNIX (1)

- El i-nodo es la cabecera de un fichero UNIX. Utiliza una tabla de 10 punteros para el acceso directo a los primeros *clusters*. A partir de ahí, 3 punteros de indexación múltiple son sucesivamente más lentos pero permiten abarcar más:
 - El puntero indirecto simple apunta a un *cluster* lleno de punteros.
 - El puntero indirecto doble (P.I.D.) apunta a un *cluster* lleno de P.I.S.
 - El puntero indirecto triple (P.I.T.) apunta a un *cluster* lleno de P.I.D.
- La idea es similar a la de las tablas de páginas multinivel.
- Lo mejor es verlo con un ejemplo:
 - A partir de la versión 4.1 de UNIX BSD, se usó este sistema con:
 - Tamaño de *cluster* de 1 Kbyte.
 - Punteros de 32 bits (4 bytes) para los *clusters*.
 - Esto permite particiones de hasta 4 Terabytes (2^{32} *clusters* de 1 Kbyte cada uno).

i-nodos en UNIX (2)

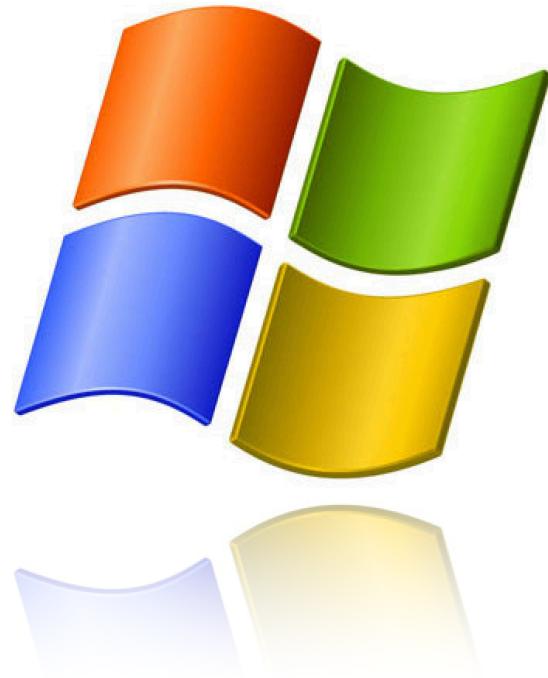


Caso estudio sobre el UNIX BSD 4.1

- Con los 10 punteros directos podemos acceder a ficheros de hasta 10 Kbytes, que es el espacio más frecuente.
- El puntero indirecto simple apunta a un *cluster* de 1 KB, en el que caben 256 punteros de 32 bits, para 256 KB más.
- El puntero indirecto doble apunta a un cluster en el que caben 256 P.I.S. de 32 bits, para $256 * 256 \text{ KB} = 64 \text{ MB}$.
- El puntero indirecto triple apunta a un cluster que contiene 256 P.I.D. de 32 bits, para $256 * 64 \text{ MB} = 16 \text{ GB}$. adicionales.
- En total, el máximo tamaño de fichero es de:
 - 16 GB + 64 MB + 266 KB.
 - ... dentro de una partición máxima que ya vimos de 4 TB.

Información que se aloja en un i-nodo

Campo	Bytes	Descripción
Modo	2	Tipo de fichero, bits de protección, bits de config. de usuario y grupo.
N-links	2	Número de entradas de directorio apuntando a este i-nodo.
UID	2	ID del usuario propietario del fichero.
GID	2	ID del grupo propietario del fichero.
Size	4	Tamaño del fichero (en bytes).
Addr	39	13 punteros (10 directos, 1 indirecto simple, otro doble y otro triple).
Gen	1	Número de usos (se incrementa cada vez que el i-nodo se reutiliza).
A-time	4	Fecha y hora en que se accedió por última vez al fichero.
B-time	4	Fecha y hora de creación del fichero.
M-time	4	Fecha y hora en que fue modificado por última vez el fichero.
C-time	4	Fecha y hora en que el i-nodo se modificó por última vez.



V. Directorios

Concepto

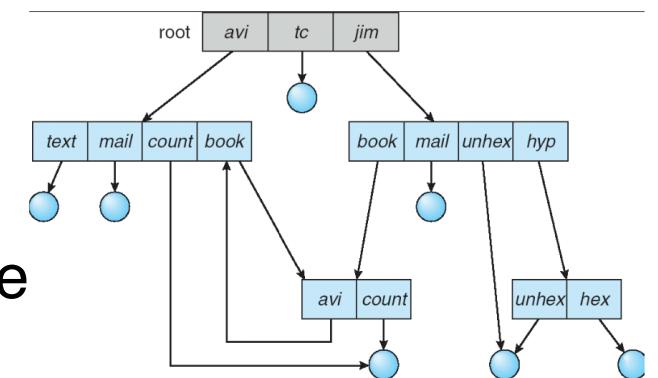
Los directorios son carpetas que permiten organizar los ficheros, agrupándolos según cierta afinidad establecida por el usuario. Ejemplos:

- Un directorio para todos mis programas Java.
- Un directorio para la asignatura S.O.

Ventajas:

- Permite localizar la información más fácilmente
- Versatilidad:
 - Un mismo fichero puede tener varias copias y nombres en distintas ubicaciones.
 - El mismo nombre puede ser usado por distintos usuarios para distintos ficheros.
 - Podemos establecer una jerarquía con directorios dentro de directorios. Ejemplo: /home/clases/ETSII/SO/lenguajeC/proyectoShell/codigo.c.

Esto último se conoce como la ruta o *path* de un fichero.



Implementación

- Originalmente, se gestionaban como ficheros especiales, a los que se aplicaban sus mismas llamadas al sistema.
- Posteriormente, se crearon nuevas llamadas al sistema para ellos: `mkdir` (creación), `rmdir` (borrado), ...
- Finalmente, se añadieron los enlaces físicos y simbólicos:
 - Un enlace físico (hard) permite establecer rutas alternativas para un fichero desde otros directorios. Se usa para ello el comando `ln`.
 - Se pueden crear muchos enlaces físicos, pero siempre dentro de una partición.
 - Si borramos un fichero, sigue existiendo para el resto de enlaces físicos.
 - Un enlace simbólico (soft) sólo es un atajo para acceder al fichero.
 - Se usa para ello el comando `ln -s`. A diferencia del enlace físico, esta opción `-s` se puede usar para un directorio o para un fichero en otro PC vía NFS.
 - Si borramos el fichero, todos sus enlaces simbólicos quedan inservibles.

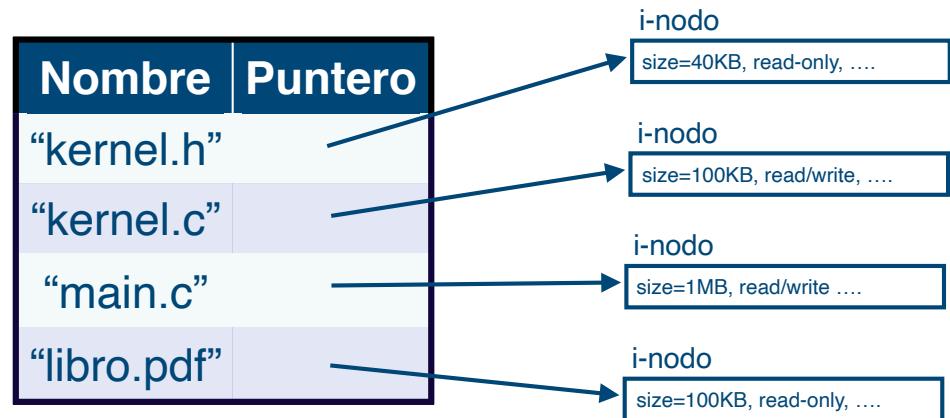
Estructuras de datos

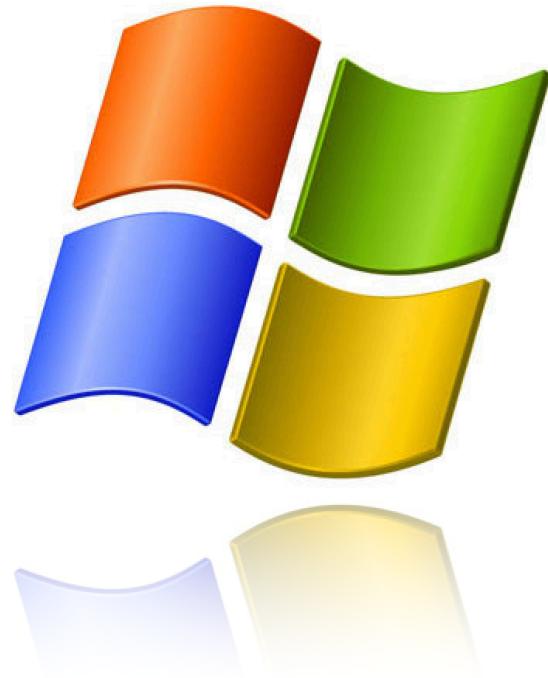
- Cada directorio organiza en una tabla sus archivos, donde hay una fila por fichero.
- En las FAT de Microsoft, dicha fila contiene el nombre y los atributos.
- En los S.O. UNIX, la fila contiene el nombre y un puntero al i-nodo del fichero, donde estarán los atributos.

FAT (Microsoft)

Nombre	Atributos
“kernel.h”	size=40KB, read-only, ...
“kernel.c”	size=100KB, read/write, ...
“main.c”	size=1MB, read/write, ...
“libro.pdf”	size=7MB, read-only, ...

UNIX





VI. Otros aspectos

Gestión del espacio libre

- Idea 1: Lo más simple es habilitar un **mapa de bits**, con un bit por cada cluster, donde: 0=libre / 1=ocupado.

- Espacio total ocupado:

- Para un disco de 1 Terabyte con *clusters* de 4 Kbytes, el mapa de bits ocuparía 256 Mbits, esto es, 32 Mbytes.

- Su principal inconveniente es la lentitud, a no ser que sacrificemos estos 32 Mbytes en caché para alojar el mapa de bits íntegramente.

- Idea 2: Usar una **lista enlazada** de *clusters* libres, tal y como hace la FAT para enlazar los *clusters* de cada fichero.

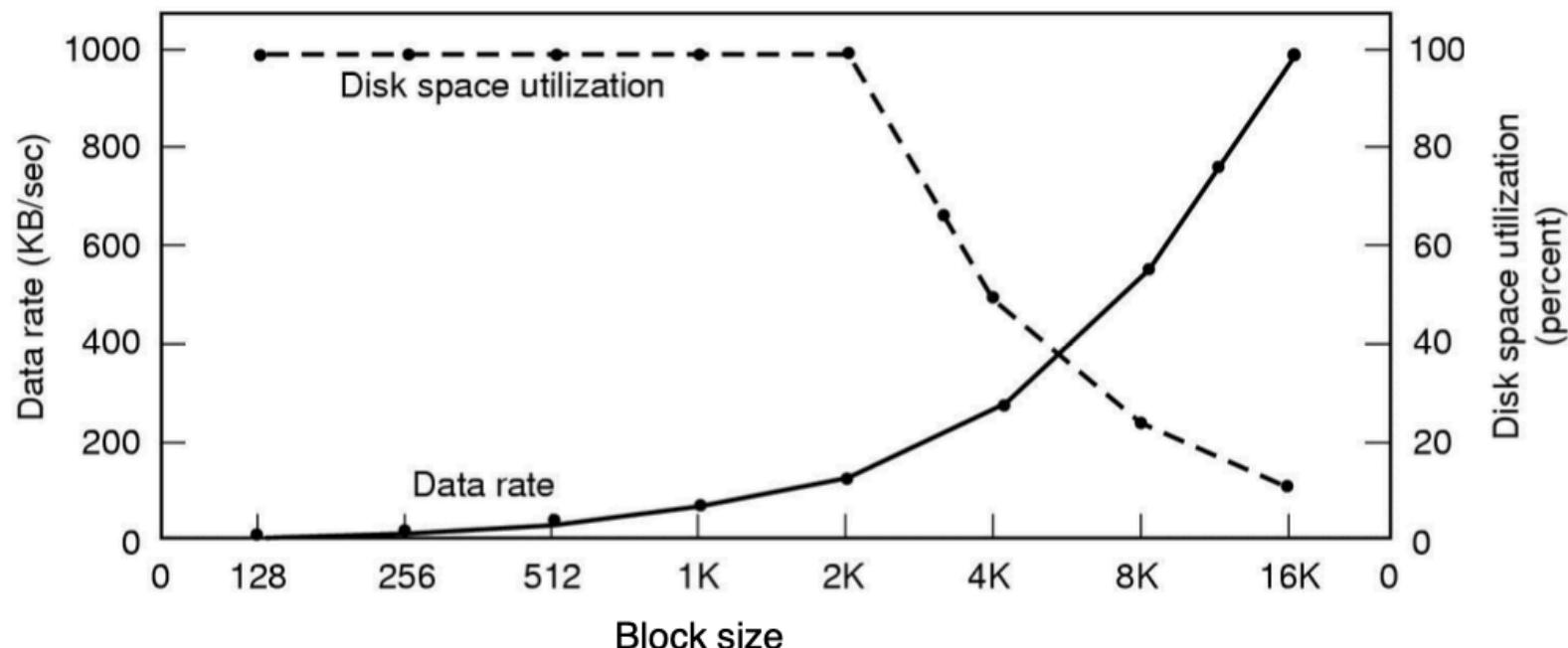
- No se desperdicia espacio, pero sigue siendo lento y cuesta tener el espacio libre consecutivamente en disco.

- Idea 3: Usar una **tabla** de acceso indexado a cada *cluster*.

- Es el esquema más rápido, a costa de sacrificar espacio.

Elección del tamaño del *cluster*

- El tamaño del *cluster* tiene la misma incidencia que el tamaño de la página en el sistema de memoria:
 - Si es pequeño, se aprovecha mejor el espacio, pero no se amortiza la latencia del disco y la gestión del alojamiento ocupa más espacio.
 - Si es grande, se desperdicia más espacio por fragmentación interna pero se aprovecha mejor el ancho de banda en el acceso a los datos.



Mejoras de rendimiento y consistencia

- El disco incluye una memoria DRAM de unos 64 Mbytes que actúa de **caché** para el sistema de almacenamiento, reteniendo los *clusters* que más se refencian.
 - Los *clusters* compiten por la caché de igual forma que las páginas por los marcos en memoria virtual (localidad, reemplazo LRU, ...).
- Organizar el disco en **grupos de cilindros**, tratando de albergar un i-nodo y sus *clusters* de datos en pistas vecinas.
 - Reduce el tiempo de posicionamiento del brazo del disco (seek).
- **Journaling:** Anotar cada operación antes de realizarla en el disco, lo que mantiene una copia de seguridad.
 - Si falla el sistema de ficheros, puede reconstruirse el disco.
 - Esta idea se incorporó a partir del sistema de ficheros “ext3” de Linux, que posteriormente también sería adoptada por el NTFS de Windows.

Bibliografía

- Se recomienda la lectura del libro que colocamos en la portada de la asignatura en el Campus Virtual UMA:
 - Operating Systems Concepts, 8th Edition, de los autores A. Silberschatz, G. Gagne, P.B. Galvin, publicado por Wiley en 2008.
- Este libro ofrece sus contenidos íntegros on-line para los estudiantes de la UMA, a través del enlace:
 - <https://www.oreilly.com/library/view/operating-system-concepts/9780470128725/?ar>
- Para este cuarto capítulo del temario de S.O. debe consultarse la cuarta parte del libro, principalmente sus dos primeros capítulos:
 - Chapter 10: File System Interface (para nuestras secciones I y II)
 - Chapter 11: File System Implementation (para las secs. III, IV, V y VI)