

Tema 5

Sistema de Ficheros



Índice



1. Introducción
2. Gestión del disco
3. Diseño del sistema de ficheros
4. Protección de archivos
5. Casos de estudio

Índice



1. Introducción

2. Gestión del disco

3. Diseño del sistema de ficheros

4. Protección de archivos

5. Casos de estudio



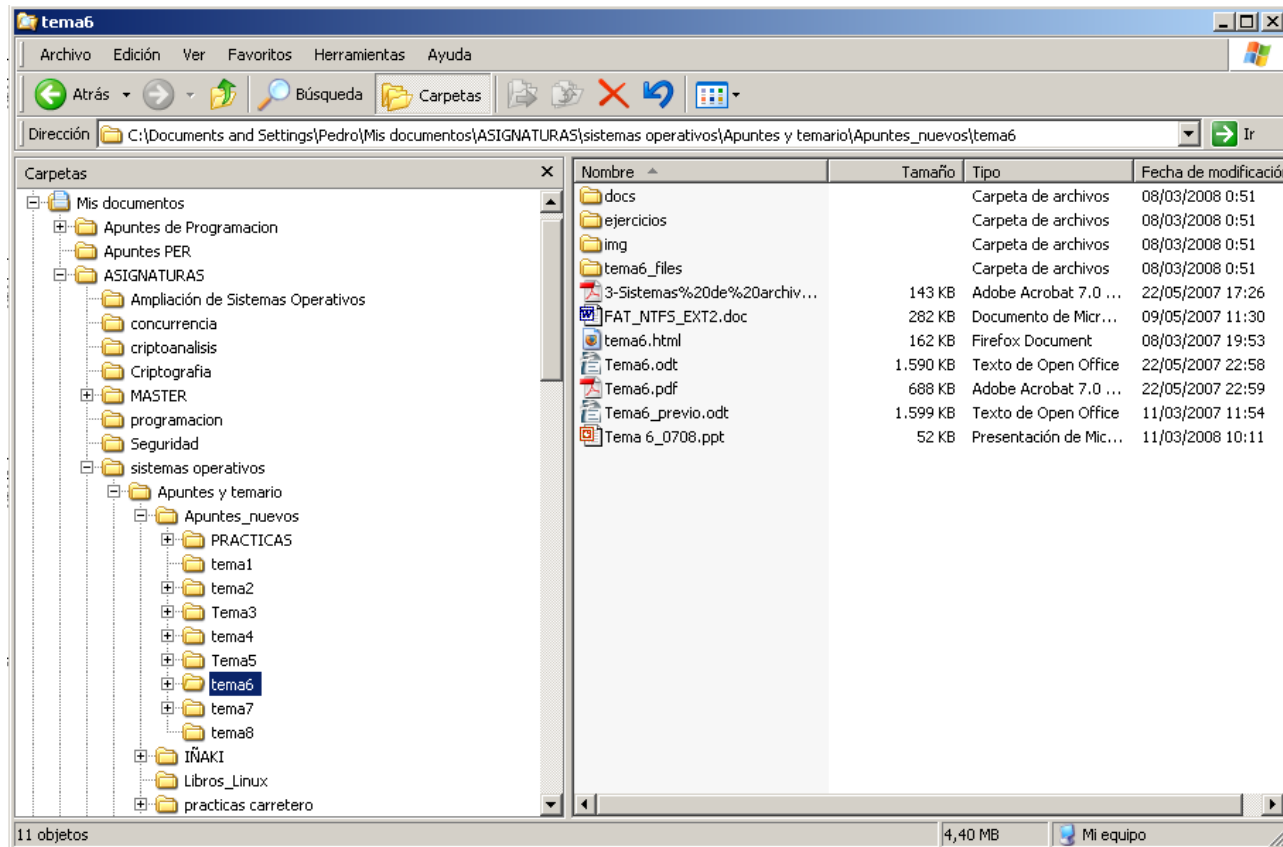
1. Introducción

- ¿Qué podemos hacer si un proceso necesita gran cantidad de espacio?
 - Asignarle espacio virtual.
- Problemas:
 - ¿Qué pasa si el proceso requiere más espacio del que existe? (i. e. aplicaciones de base de datos).
 - El almacenamiento en memoria es volátil.
- Para solucionar estos problemas optamos por guardar los datos en un almacenamiento secundario.
 - Este almacenamiento se realiza a través del concepto de **fichero o archivo**.
 - La parte del sistema operativo que se encarga de gestionar este almacenamiento es el sistema o **servidor de ficheros**.

1. Introducción

1.1 Visión del usuario

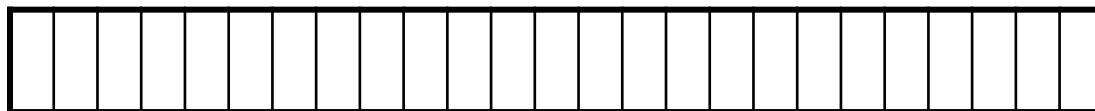
- Un conjunto de ficheros
- Un conjunto de directorios



1. Introducción

1.1 Visión del usuario. Ficheros

- ***Un fichero es una unidad de almacenamiento lógico no volátil que agrupa un conjunto de información relacionada entre sí bajo un mismo nombre.***
- Según el sistema Operativo, los nombres deben adecuarse a ciertas reglas.
 - Fichero.txt
 - Practica De sistemas.operativos.tar.gz
 - Etc ...
- Internamente, la información puede estar estructurada de diversas formas:
 - **En bytes:** serie de bytes no estructurada que proporciona una máxima flexibilidad (UNIX, Windows).

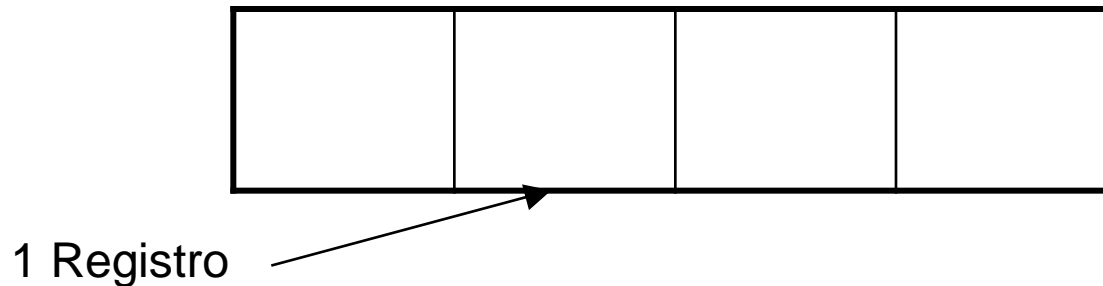


1 Byte

1. Introducción

1.1 Visión del usuario. Ficheros

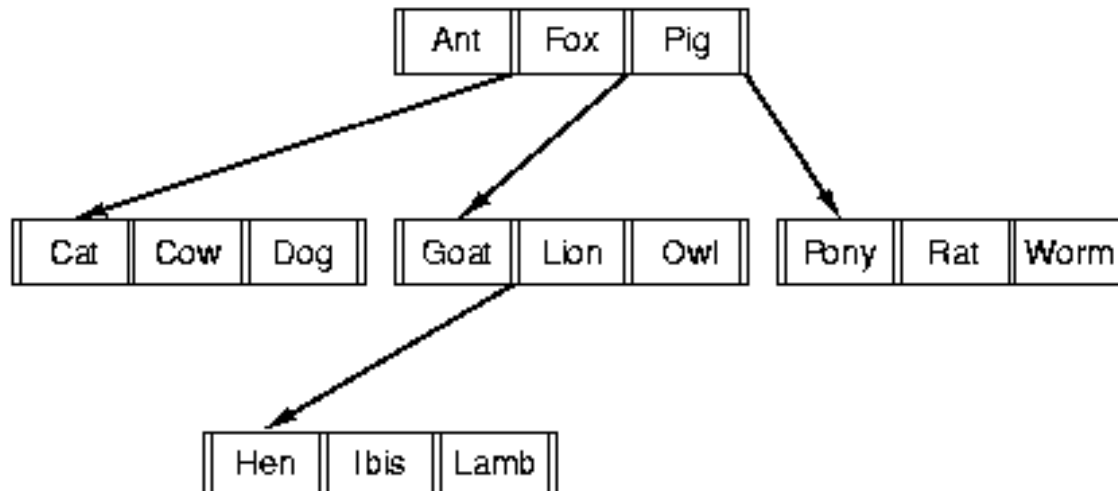
- Internamente, la información puede estar estructurada de diversas formas:
 - **En registros:** se comportan como una secuencia de registros de longitud fija y cierta estructura interna. Las operaciones de lectura/escritura se hacen con registros (CPM).



1. Introducción

1.1 Visión del usuario. Ficheros

- Internamente, la información puede estar estructurada de diversas formas:
- **Jerárquica:** consta de un árbol de registros de distinta longitud. Cada registro tiene un campo que se usa como clave. El **árbol** se ordena mediante este campo para mejorar la búsqueda.



1. Introducción

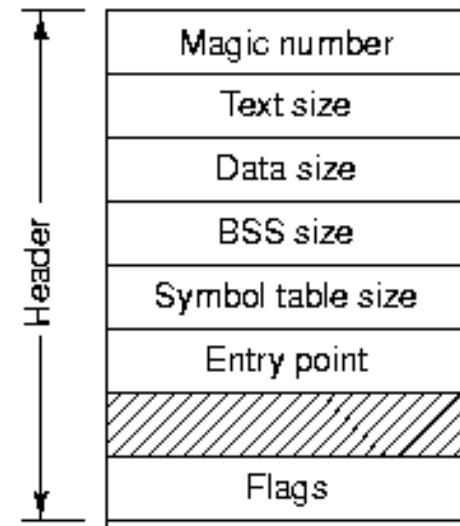
1.1 Visión del usuario. Ficheros

- Internamente el SO puede dar soporte a cada una de estas estructuras o sólo soportar una estructura lo suficientemente flexible (la estructura secuencial) para permitir desarrollar sobre ella cualquier otra estructura.
- Una vez que hay información almacenada en el fichero, ¿**Cómo** podemos **acceder** a ella?
- Hay dos maneras básicas:
 - **Acceso secuencial** (cintas): los bytes se leen en orden empezando siempre desde el comienzo.
 - **Acceso directo o aleatorio** (dispositivos de acceso directo): el archivo se considera como un conjunto de registros. Se puede acceder a ellos de forma desordenada. Internamente, la información puede estar estructurada de diversas formas:

1. Introducción

1.1 Visión del usuario. Ficheros

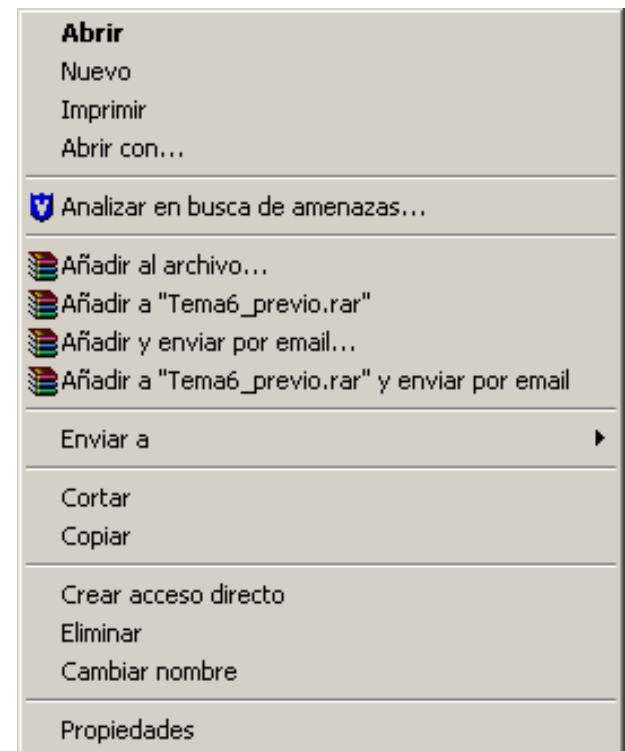
- La mayoría de los S.O disponen de **tipos** de ficheros distintos:
 - Regulares o de usuario
 - Directorios.
 - Especiales de caracteres, dispositivos de E/S (ratones, impresoras)
 - Especiales de bloques, discos.
 - Tuberías para la comunicación entre procesos.
 - Enlaces.
- ¿Cómo podemos saber qué tipo de información contiene un fichero regular?
 - Como parte del propio nombre del fichero. Usamos la extensión. (.pas, .c, .bas)
 - Como parte del contenido del fichero. Este empieza con una cabecera en donde se indica lo que contiene el fichero.



1. Introducción

1.1 Visión del usuario. Ficheros

- **Operaciones** que puede realizar un usuario sobre los ficheros:
 - Crear
 - Escribir
 - Borrar
 - Añadir
 - Abrir
 - Buscar (ficheros con acceso aleatorio)
 - Cerrar
 - Obtener atributos
 - Leer
 - Establecer atributos
 - Renombrar



1. Introducción

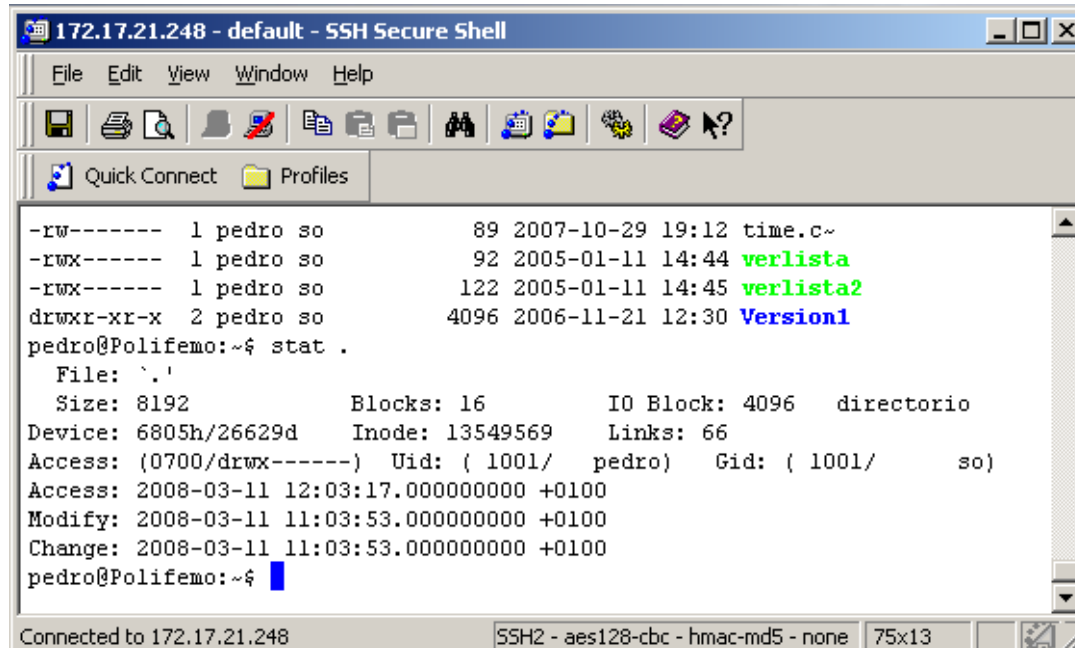
1.1 Visión del usuario. Ficheros

- ***El servidor de ficheros es la parte del sistema operativo que se ocupa de facilitar el manejo de los dispositivos periféricos, ofreciendo una visión lógica simplificada de los mismos en forma de archivos.***
- Las **funciones** que debe cumplir el sistema de ficheros son:
 - Ocultar aspectos específicos de los dispositivos de almacenamiento (independencia del dispositivo).
 - Posibilitar la operación sobre los ficheros (copiar, crear, eliminar, renombrar, listar contenido, cambiar atributos o permisos).
 - Posibilitar la compartición de los ficheros de forma controlada.
 - Proporcionar una estructura flexible a los ficheros.
 - Proporcionar utilidades de respaldo y recuperación de la información.
 - Proporcionar mecanismos de cifrado y descifrado de la información.

1. Introducción

1.1 Visión del usuario. Directorios

- ***Los directorios son ficheros especiales que contienen información sobre otros ficheros y que posibilitan al usuario organizar sus ficheros creando una entrada por cada fichero.***
- Información que contiene el directorio:
 - Tipo de fichero
 - Tamaño
 - Propietario
 - Permisos
 - Fecha y hora
 - Estadísticas
 - Bloques de disco usados



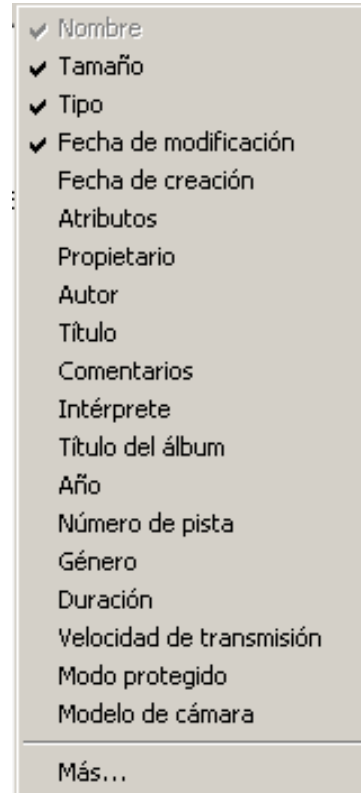
```
172.17.21.248 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
-rw----- 1 pedro so      89 2007-10-29 19:12 time.c~
-rwx----- 1 pedro so     92 2005-01-11 14:44 verlista
-rwx----- 1 pedro so    122 2005-01-11 14:45 verlista2
drwxr-xr-x 2 pedro so   4096 2006-11-21 12:30 Version1
pedro@Polifemo:~$ stat .
  File: `.'
  Size: 8192          Blocks: 16          IO Block: 4096   directorio
Device: 6805h/26629d Inode: 13549569   Links: 66
Access: (0700/drwx-----)  Uid: ( 1001/   pedro)   Gid: ( 1001/    so)
Access: 2008-03-11 12:03:17.000000000 +0100
Modify: 2008-03-11 11:03:53.000000000 +0100
Change: 2008-03-11 11:03:53.000000000 +0100
pedro@Polifemo:~$
```

Connected to 172.17.21.248 SSH2 - aes128-cbc - hmac-md5 - none 75x13

1. Introducción

1.1 Visión del usuario. Directorios

- ***Los directorios son ficheros especiales que contienen información sobre otros ficheros y que posibilitan al usuario organizar sus ficheros creando una entrada por cada fichero.***
- Información que contiene el directorio:
 - Tipo de fichero
 - Tamaño
 - Propietario
 - Permisos
 - Fecha y hora
 - Estadísticas
 - Bloques de disco usados



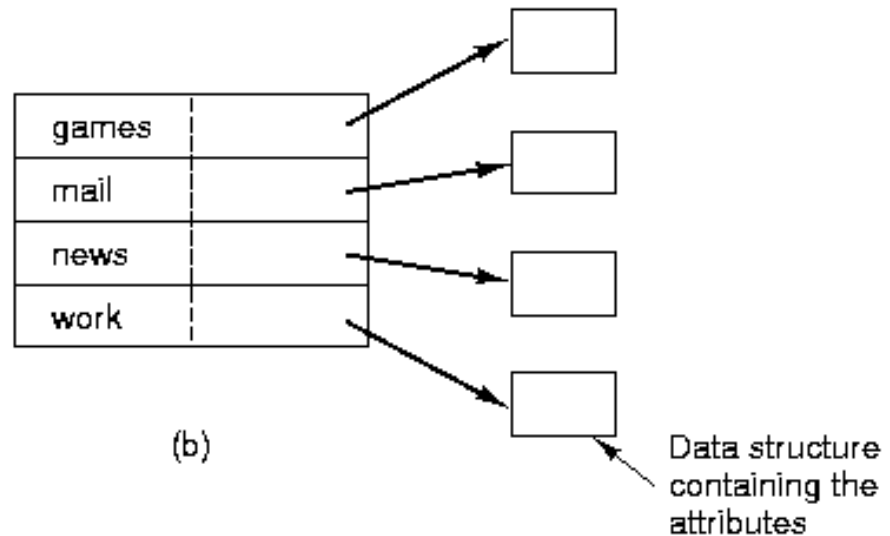
1. Introducción

1.1 Visión del usuario. Directorios

- Formas de almacenar la información en el directorio:
 - a) En cada entrada del directorio almacenamos el nombre del archivo y una serie de atributos.
 - b) En cada entrada del directorio almacenamos el nombre del archivo y un puntero a una estructura de datos donde están contenidos los atributos.

games	attributes
mail	attributes
news	attributes
work	attributes

(a)

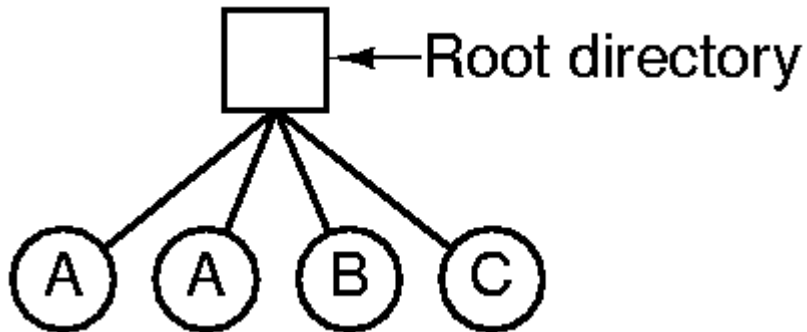


(b)

1. Introducción

1.1 Visión del usuario. Directorios

- **Organización a un nivel.** En su forma más sencilla el sistema sólo tiene un directorio, que incluye todos los ficheros de todos los usuarios.



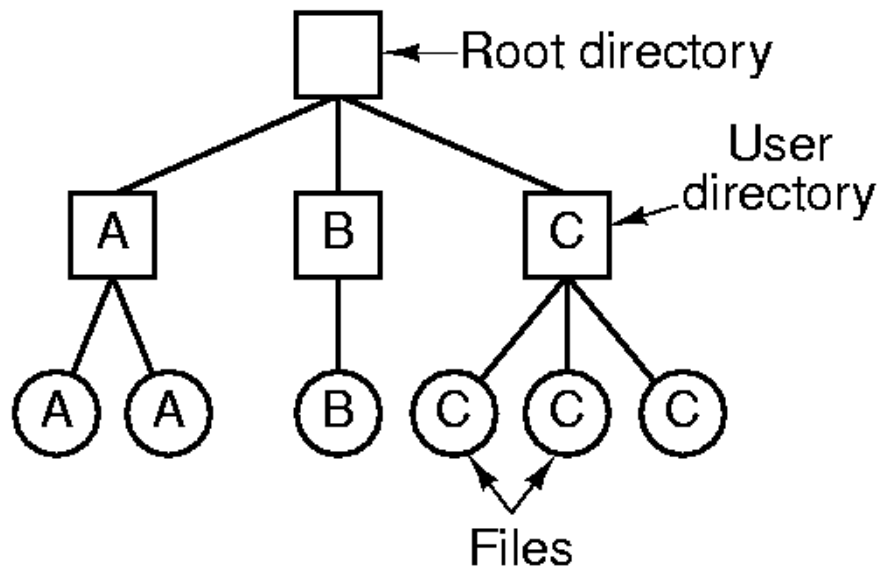
Problemas:

- Protección
- Confusiones
- Conflictos
- Mismos nombres

1. Introducción

1.1 Visión del usuario. Directorios

- **Organización a dos niveles.** Un directorio por usuario.



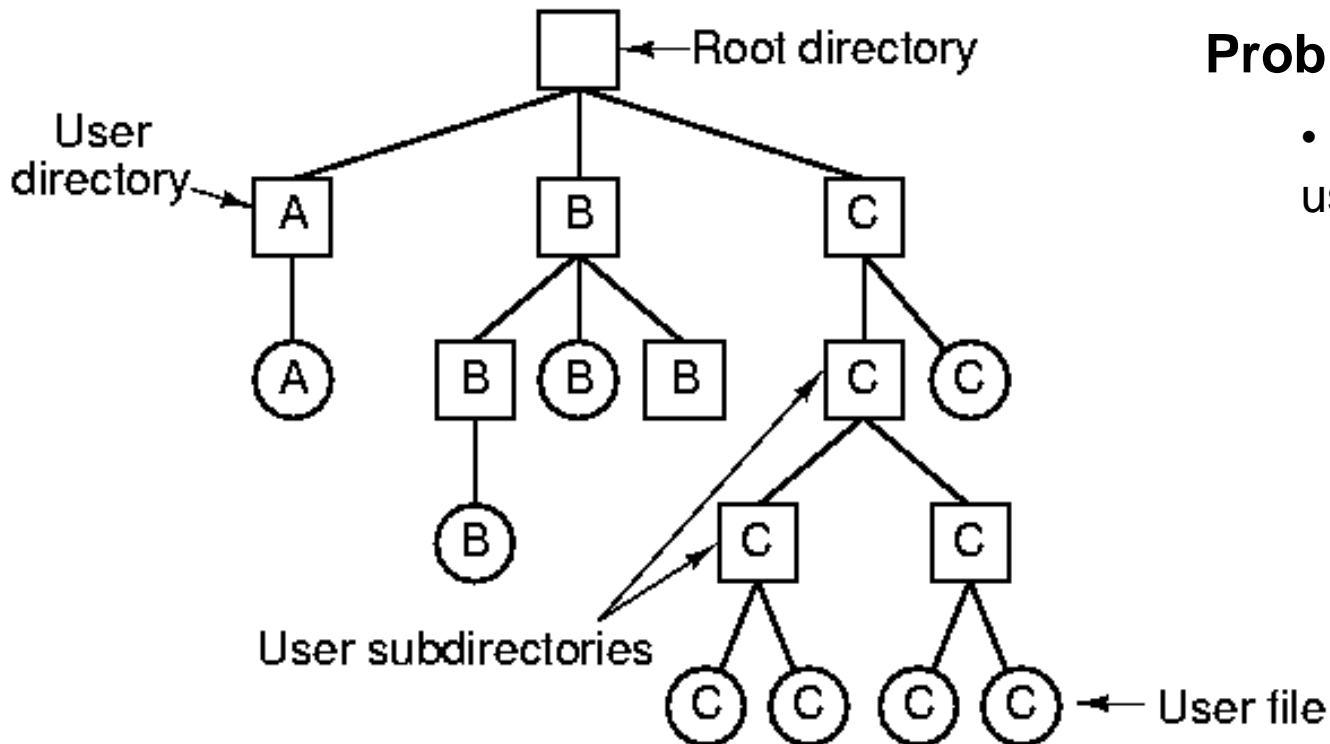
Problemas:

- Organización para cada usuario

1. Introducción

1.1 Visión del usuario. Directorios

- Organización en árbol.



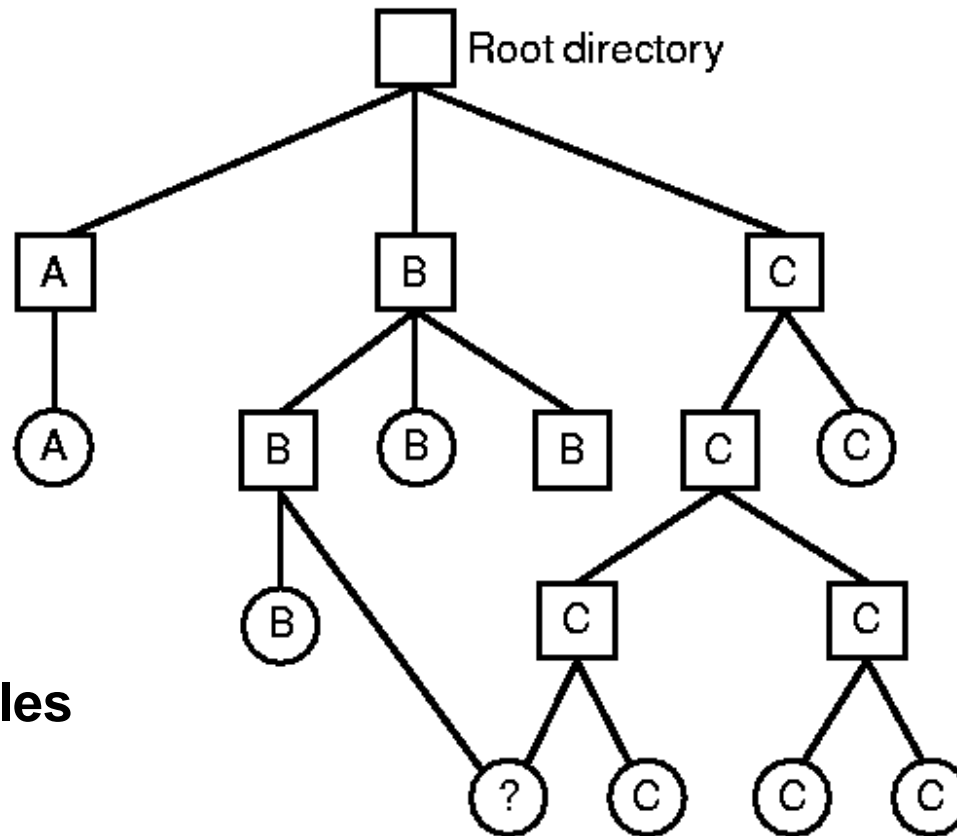
Problemas:

- Compartir ficheros entre usuarios

1. Introducción

1.1 Visión del usuario. Directorios

- **Grafos Acíclicos.** Permiten la realización de enlaces y enlaces simbólicos.



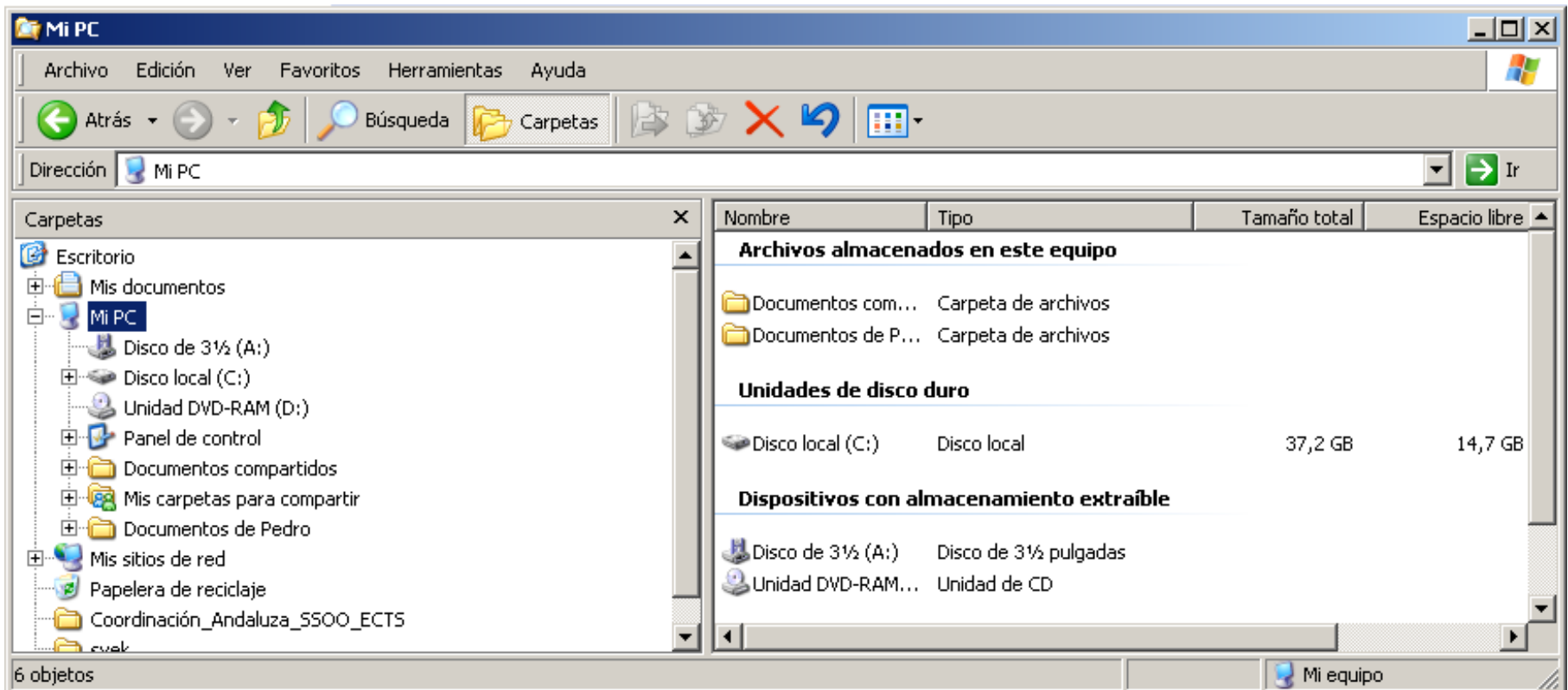
Shared file

- **Grafos Generales**

1. Introducción

1.1 Visión del usuario. Directorios

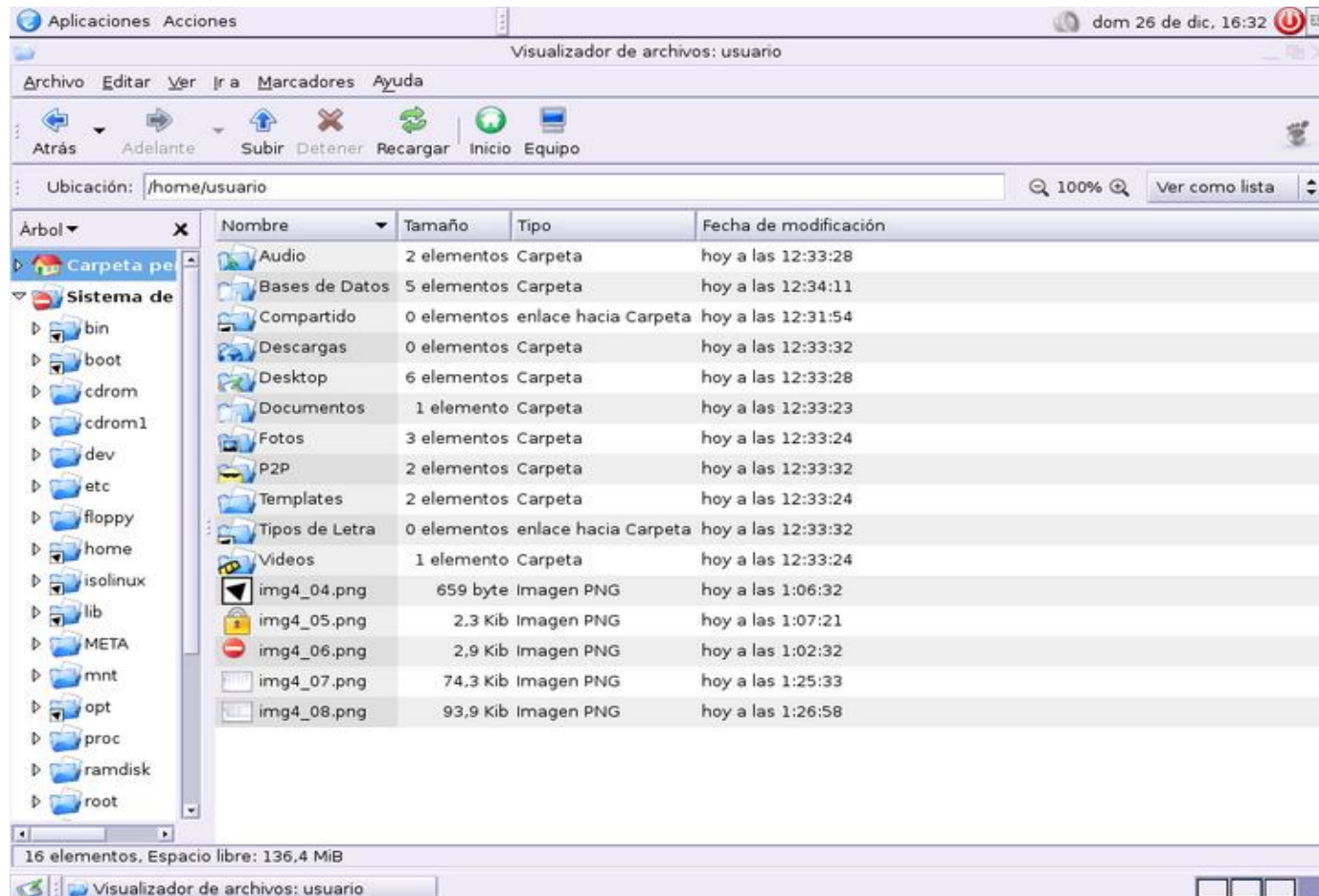
- **Árbol único vs Árboles Múltiples.**



1. Introducción

1.1 Visión del usuario. Directorios

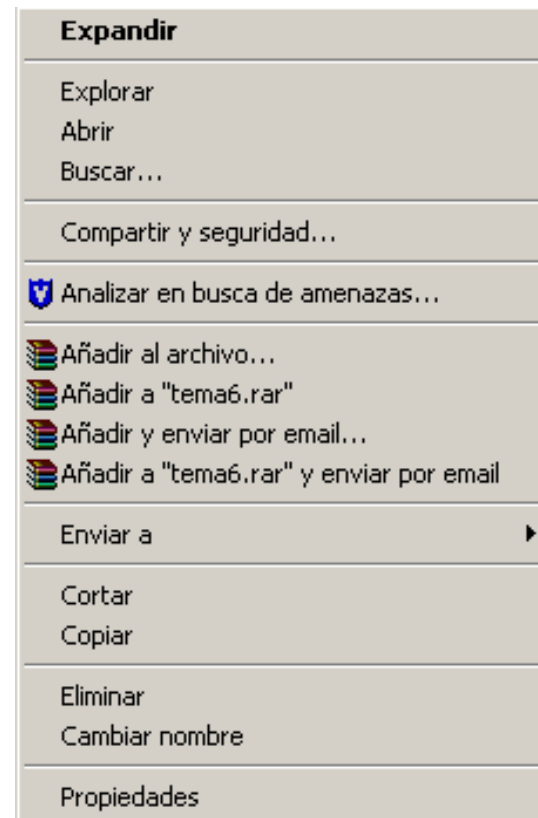
- **Árbol único vs Árboles Múltiples.**



1. Introducción

1.1 Visión del usuario. Directorios

- **Operaciones** que podemos realizar sobre los directorios:
 - Crear
 - Leer
 - Borrar
 - Renombrar
 - Abrir
 - Enlazar
 - Cerrar
 - Desenlazar





1. Introducción

1.2 Visión del diseñador

- La preocupación del diseñador ha de ser el cómo se **implementa** el sistema de ficheros.
- Las tareas que deberán tener en cuenta son:
 - Elección del **tamaño** adecuado de **bloque** lógico (debería ser múltiplo del espacio físico).
 - Organización del disco (políticas de **acceso**, ¿por qué orden se atienden las peticiones de lectura y escritura que llegan?).
 - Gestión del **espacio libre** (cuándo se incrementa el espacio ocupado, qué bloques de disco se le asocian).
 - Elección de las **estructuras** de almacenamiento.
 - Gestión del **espacio ocupado** (debemos evitar conceder a un fichero espacio ocupado por otro).

Índice



1. Introducción
- 2. Gestión del disco**
3. Diseño del sistema de ficheros
4. Protección de archivos
5. Casos de estudio



2. Gestión del Disco

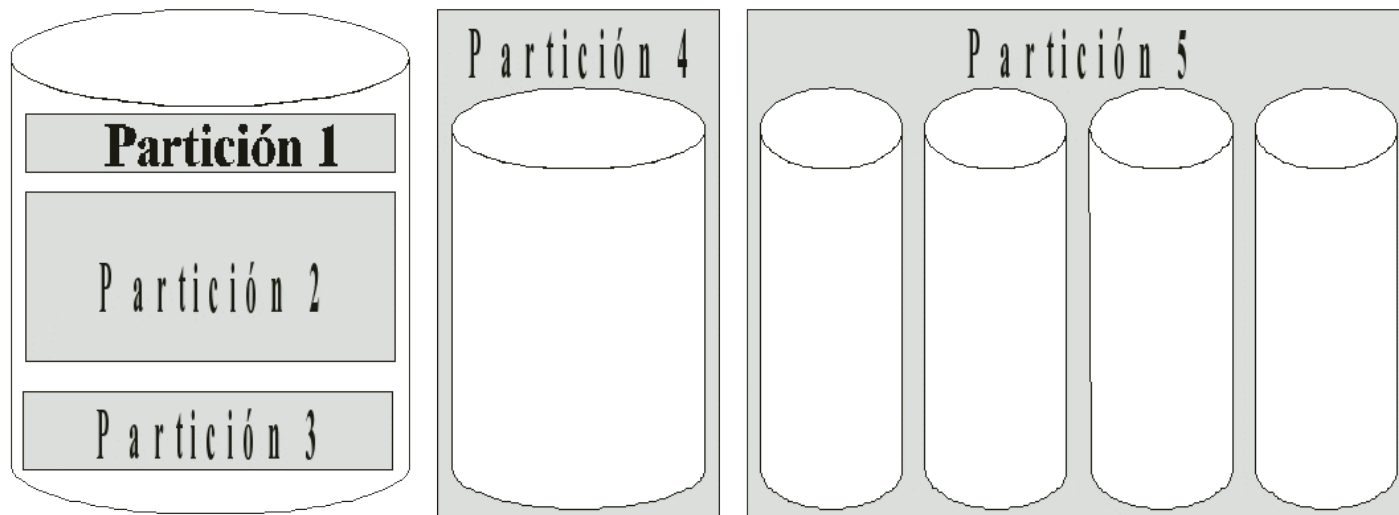
- Los archivos y directorios se almacenan en dispositivos de almacenamiento secundario.
- En estos dispositivos debemos distinguir entre:
 - **Sector:** Unidad de acceso. Unidad mínima en la que dividimos el disco.
 - **Bloque:** Unidad de asignación de transferencia. N° de sectores que se transfieren cada vez que traemos o llevamos información al disco.
 - **Registro:** Unidad lógica de trabajo del usuario. Registro lógico de almacenamiento de información del usuario. Es muy buena idea que el registros sea múltiplo del tamaño del bloque. A nivel programación sería así:

```
Datos = Record  
Nombre: Array [1..100] of char  
...  
...  
End;
```

2. Gestión del Disco

2.1 Estructura lógica del disco

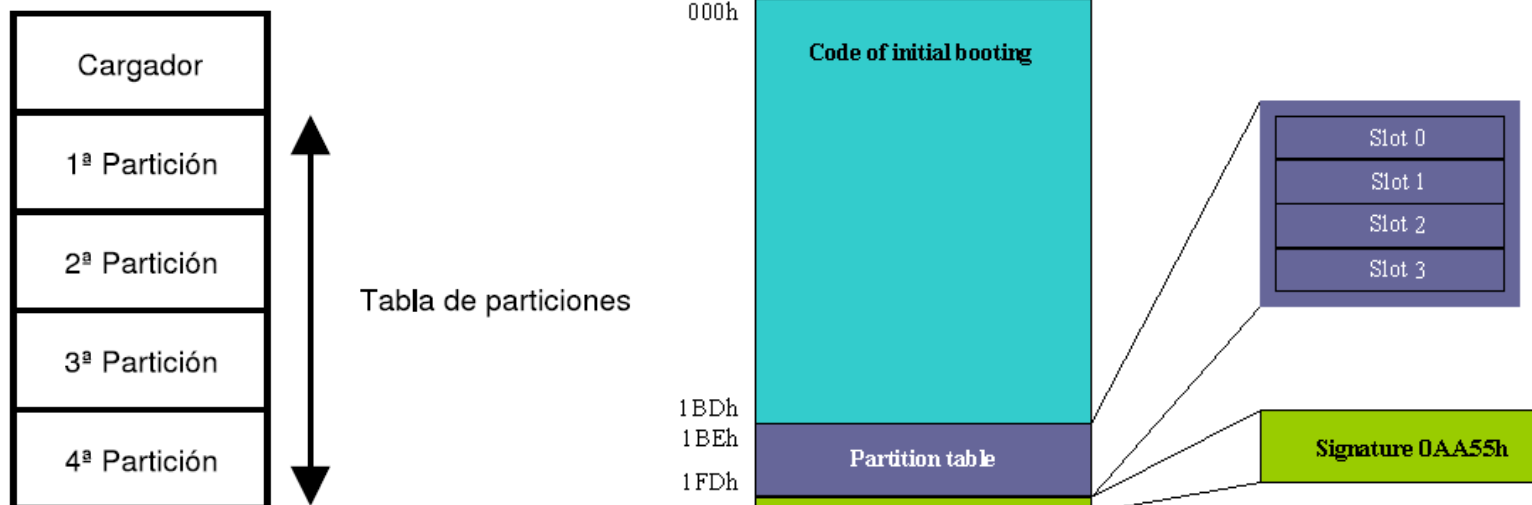
- Estos dispositivos vienen vacíos por lo que antes de instalar el sistema de ficheros es necesario dividirlos lógicamente en particiones o volúmenes.
- ***Una partición es una porción de un disco a la que se la dota de una identidad propia y que puede ser manipulada por el sistema operativo como una entidad lógica independiente***



2. Gestión del Disco

2.1 Estructura lógica del disco

- Al finalizar esta operación tendremos en el HD tres elementos:
 - Sector de arranque o MBR (Master Boot Record).
 - Espacio particionado.
 - Espacio sin particionar.
- EL MBR es el primer sector de todo disco duro (cilindro 0, cabeza 0, sector 1)



2. Gestión del Disco

2.1 Estructura lógica del disco

GParted

GPtred Edit View Device Partition Help

New Delete Resize/Move Copy Paste Undo Apply

/dev/sda (232.88 GB)

ntfs unknown ext3 extended linux-swap fat32 xfs used unused

Partition	Filesystem	Size	Used	Unused	Flags
/dev/sda1	ntfs	7.81 GB	---	---	boot
/dev/sda2	unknown	9.77 GB	---	---	
/dev/sda3	ext3	7.81 GB	2.39 GB	5.43 GB	
▼ /dev/sda4	extended	207.49 GB	---	---	
/dev/sda5	linux-swap	1.95 GB	---	---	
/dev/sda6	ext3	9.76 GB	568.52 MB	9.20 GB	
/dev/sda7	fat32	97.66 GB	13.28 GB	84.38 GB	
/dev/sda8	xfs	98.11 GB	7.76 GB	90.36 GB	

0 operations pending

2. Gestión del Disco

2.1 Estructura lógica del disco

```
172.17.21.248 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

pedro@Polifemo:~$ clear
pedro@Polifemo:~$ sudo fdisk /dev/cciss/c0d0

El número de cilindros para este disco está establecido en 17709.
No hay nada malo en ello, pero es mayor que 1024, y en algunos casos
podría causar problemas con:
1) software que funciona en el inicio (p.ej. versiones antiguas de LILO)
2) software de arranque o particionamiento de otros sistemas operativos
(p.ej. FDISK de DOS, FDISK de OS/2)

Orden (m para obtener ayuda): p

Disco /dev/cciss/c0d0: 145.6 GB, 145667358720 bytes
255 cabezas, 63 sectores/pista, 17709 cilindros
Unidades = cilindros de 16065 * 512 = 8225280 bytes

    Disposit. Inicio    Comienzo      Fin      Bloques  Id Sistema
/dev/cciss/c0d0p1         1         24       192748+   83  Linux
/dev/cciss/c0d0p2        25       1240      9767520   83  Linux
/dev/cciss/c0d0p3       1241       1483     1951897+   82  Linux swap / Solaris
/dev/cciss/c0d0p4       1484       17709    130335345   5  Extendida
/dev/cciss/c0d0p5       1484       16086     117298566   83  Linux
/dev/cciss/c0d0p6      16087       17709     13036716   83  Linux

Orden (m para obtener ayuda):
```

Connected to 172.17.21.248 SSH2 - aes128-cbc - hmac-md5 - none 125x25 NUM

2. Gestión del Disco

2.1 Estructura lógica del disco

- ¿Qué pasos realizan un ordenador previos a la carga del sistema operativo?
 1. La BIOS y la CPU hacen diversos test, power-on self test (POST).
 2. La BIOS busca el dispositivo de arranque que normalmente es el primer disco.
 3. De este dispositivo carga en memoria el MBR y le pasa el control al MBC.
 4. El MBC busca en la tabla de particiones la marcada como activa (BOOT=*).
 5. Carga en memoria el sector de arranque de la partición activa.
 6. Transfiere el control al código contenido en dicho sector.

Índice



1. Introducción
2. Gestión del disco
- 3. Diseño del sistema de ficheros**
4. Protección de archivos
5. Casos de estudio

3. Diseño del sistema de ficheros

- Una vez creadas las particiones, el sistema operativo debe crear las estructuras de los sistemas de archivos dentro de esas particiones (format, mkfs).
- El resultado final de esta operación depende del sistema operativo.
- Estructura de los sistemas de archivos de los sistemas operativos más importantes:

BOOT	1º COPIA DE LA FAT	2º COPIA DE LA FAT	DIRECTORIO RAÍZ	DATOS Y DIRECTORIOS	MS-DOS
------	-----------------------	-----------------------	--------------------	---------------------	---------------

BOOT	Superbloque	Mapas de bits	i-nodos	DATOS Y DIRECTORIOS	UNIX
------	-------------	---------------	---------	---------------------	-------------

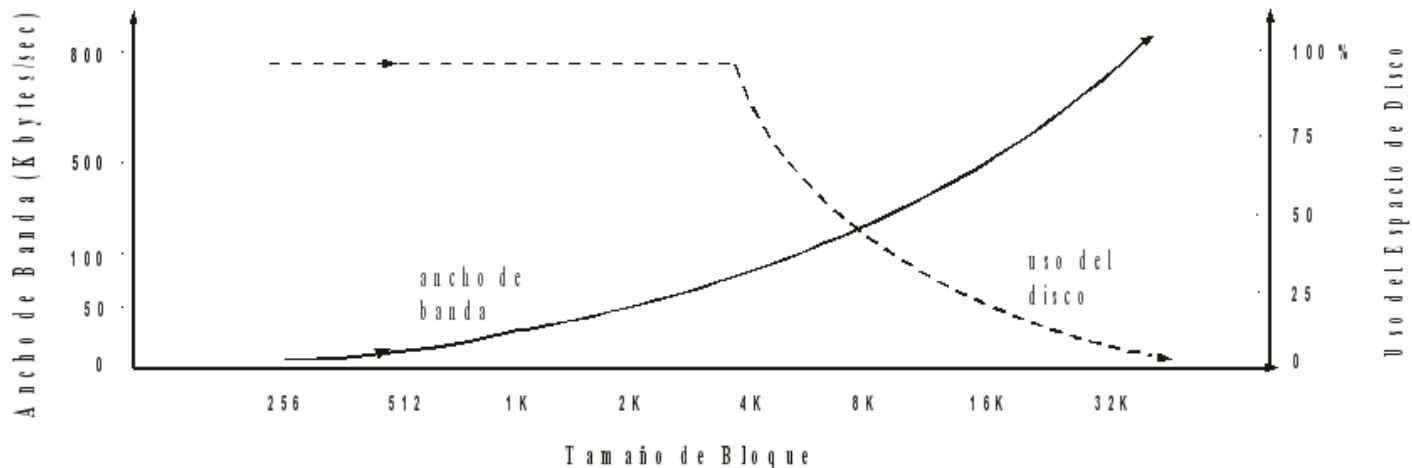
BOOT	MFT	DATOS Y DIRECTORIOS	WINDOWS-NT
------	-----	---------------------	-------------------

3. Diseño del sistema de ficheros

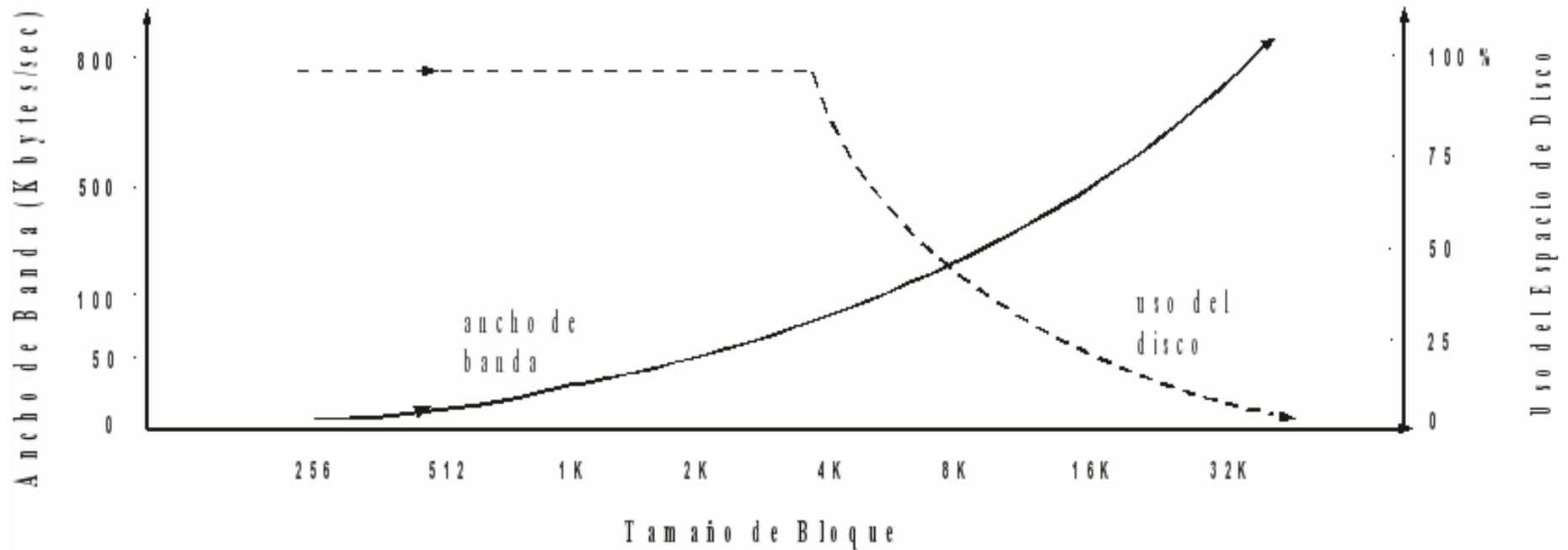
- ¿Cómo se almacena la información en estas particiones?
 - **Almacenamiento secuencial contiguo de bloques**, se nos plantea un problema al modificar el tamaño de los ficheros (traslado completo del archivo en busca de un hueco)
 - **Almacenamiento secuencial no contiguo de bloques**, el inconveniente al que se enfrentaría el SO sería conocer qué bloques utilizan cada uno de los archivos; para esto se definirán distintas estructuras en función del S.O. (UNIX, DOS, etc.).
- ¿Cuándo se crean estos bloques? Cuando se formatean las particiones.
- El tamaño de un sistema de ficheros se establece en bloques.

3. Diseño del sistema de ficheros

- El bloque es la mínima unidad de información que maneja el SO.
- Es muy importante establecer un tamaño de bloque adecuado, de ello dependerá la eficacia de las lecturas y el aprovechamiento que se haga del disco.
- Debemos encontrar el equilibrio entre las siguientes cuestiones de implementación:
 - Tamaño de los ficheros almacenados.
 - Porcentaje de disco usado / desperdiciado (fichero - bloque).
 - Velocidad de transferencia.



3. Diseño del sistema de ficheros



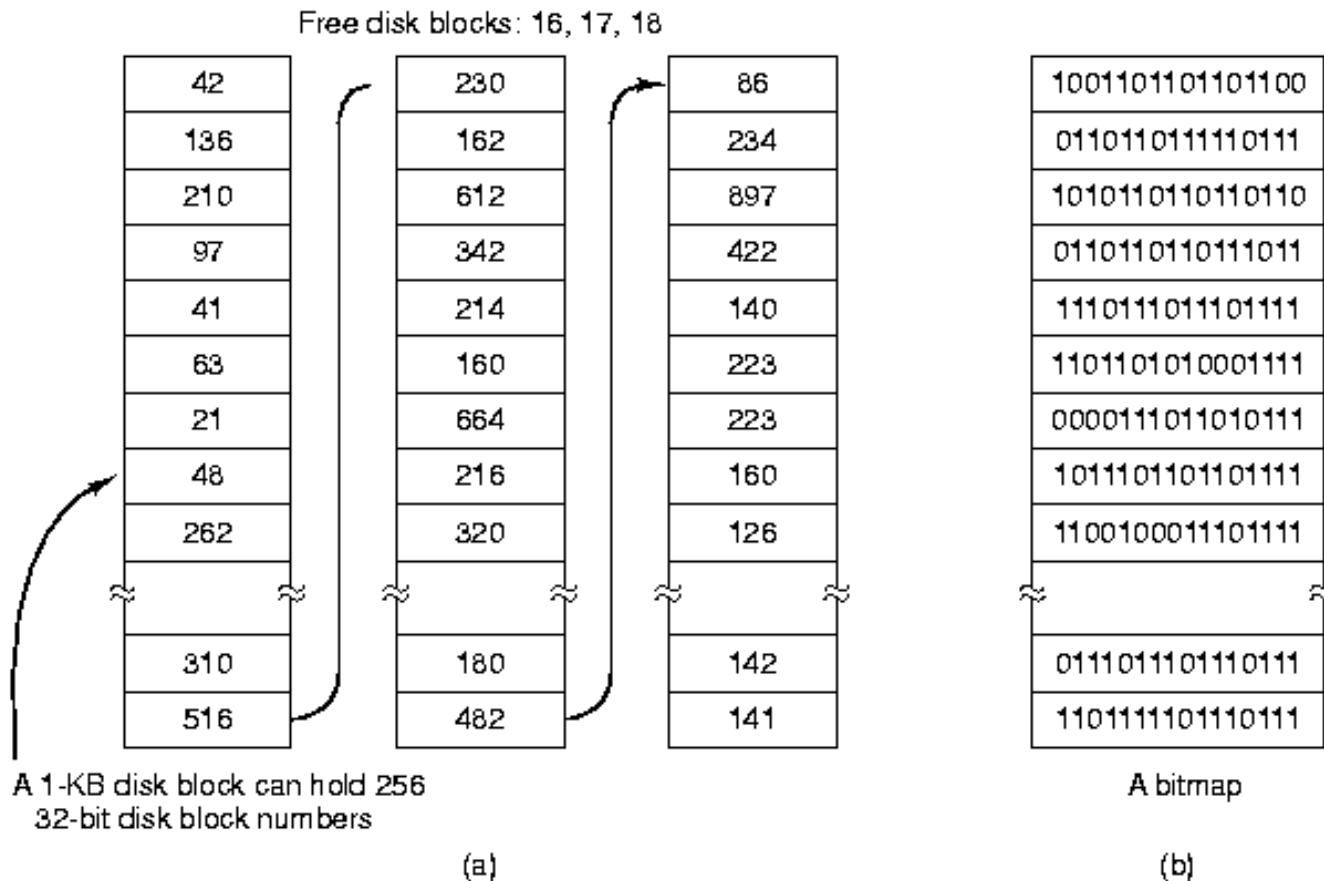
3. Diseño del sistema de ficheros

3.1 Gestión del espacio libre

- Es necesario gestionar los **bloques libres** que nos quedan en los dispositivos de almacenamiento.
- ¿Cómo podemos implementar esta gestión?
 - **Mapa de Bits**: En una tabla de bits definimos si un bloque está ocupado o no lo está. Necesitamos 1 bit por cada bloque del disco. (Por ejemplo 1= ocupado y 0=libre).
 - **Lista Enlazadas** (MS-DOS): El SO conoce la dirección de un bloque en el que están almacenados punteros a bloques libres. Si se necesitan más bloques, la última posición apuntará al siguiente bloque con apuntadores a bloques libres.

3. Diseño del sistema de ficheros

3.1 Gestión del espacio libre



3. Diseño del sistema de ficheros

3.1 Gestión del espacio libre

- Sólo si el disco está casi completo la lista enlazada requiere menos bloques que el mapa de bits.
- Si se puede guardar en memoria todo el mapa de bits es preferible este método.
- En otro caso es preferible la lista enlazada (porque un bloque de la lista enlazada tiene siempre bloques libres, y un solo bloque del mapa de bits puede que no tenga ninguno libre).
- Una posible optimización consiste en reunir los bloques libres en agrupaciones o **clúster** y tener un mapa de bits (Unix, Linux, NT) o una lista enlazada a estas **agrupaciones**.
- En el caso de usar agrupaciones, la **mínima unidad** que se puede asignar es una agrupación.

3. Diseño del sistema de ficheros

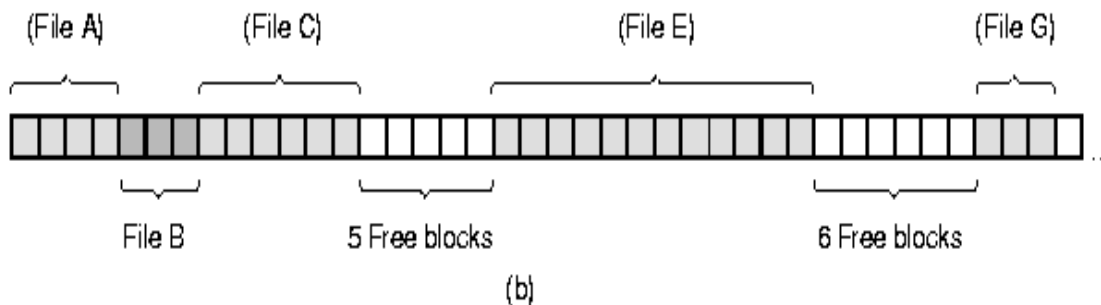
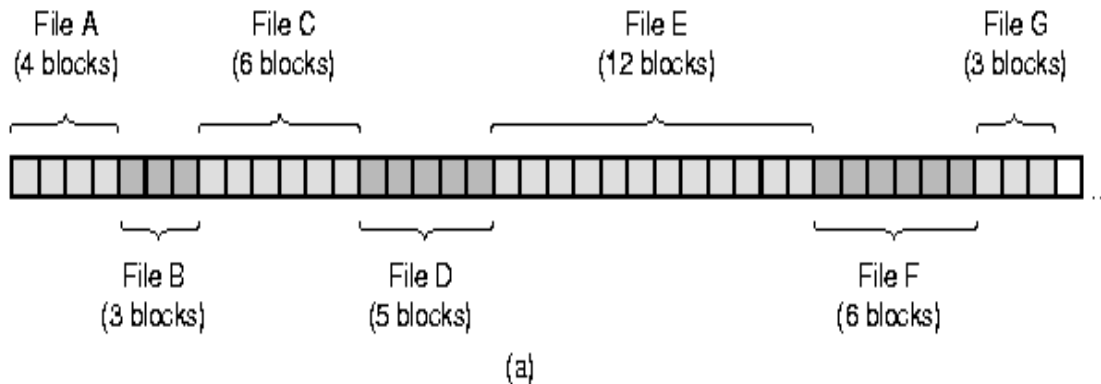
3.2 Almacenamiento de ficheros

- Un fichero está compuesto de una sucesión de bloques. El sistema operativo debe conocer los bloques de cada archivo.
- Hay distintas aproximaciones que intentan solucionar este problema respondiendo a las preguntas:
 - Cuando se crea un nuevo archivo, **¿se asigna de una sola vez el máximo espacio que necesite?** Es muy difícil saber el tamaño máximo de un fichero, se suelen hacer sobreestimaciones que provocan un derroche de espacio.
 - El espacio se asigna a un archivo en forma de una o más unidades contiguas, que se llaman agrupaciones. El tamaño de una sección puede variar desde un único bloque a un archivo entero. **¿Qué tamaño de agrupación debería usarse para asignar archivos?**
 - **¿Qué tipo de estructura de datos se usará para guardar constancia de las agrupaciones asignadas a un archivo?**

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Bloques consecutivos

- También se le denomina de asignación previa.



Ventajas:

- Sencillo de implementar
- Rendimiento de E/S muy alto.

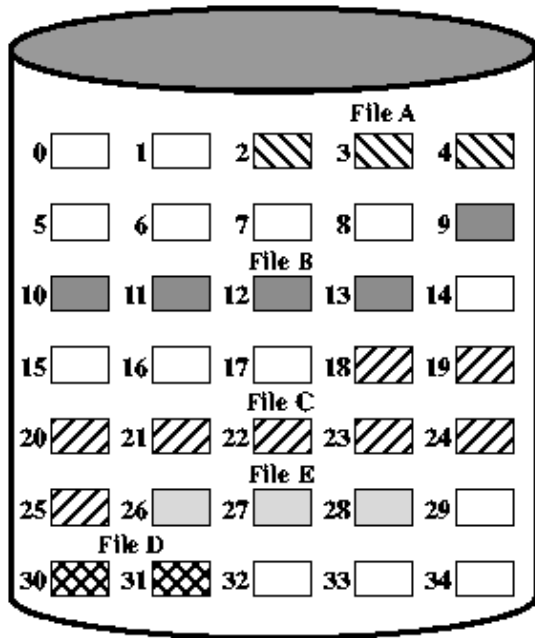
Desventajas:

- No conocemos el tamaño máximo del fichero → **reubicación**
- Fragmentación externa → **compactación**

3. Diseño del sistema de ficheros

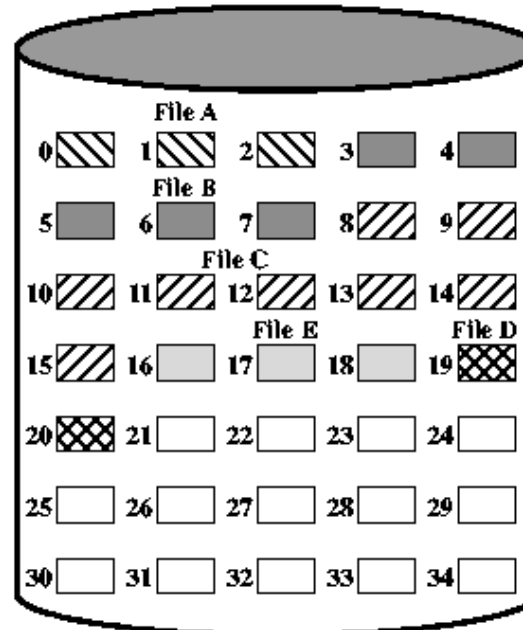
3.2 Almacenamiento de ficheros. Bloques consecutivos

- Compactación



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



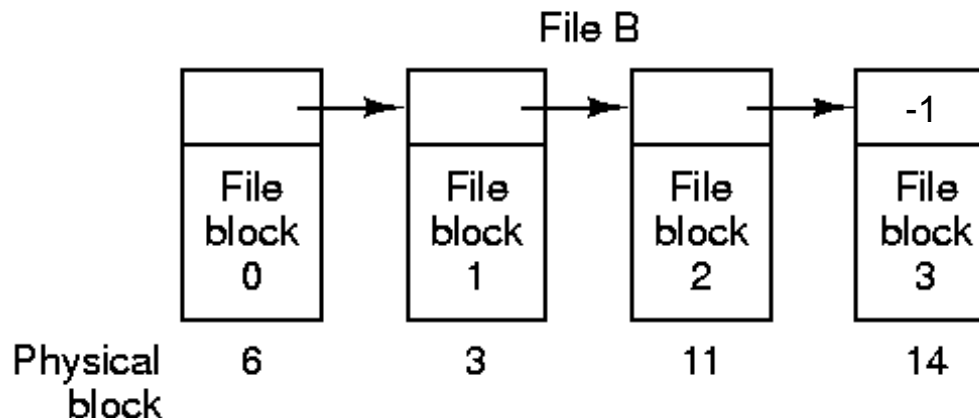
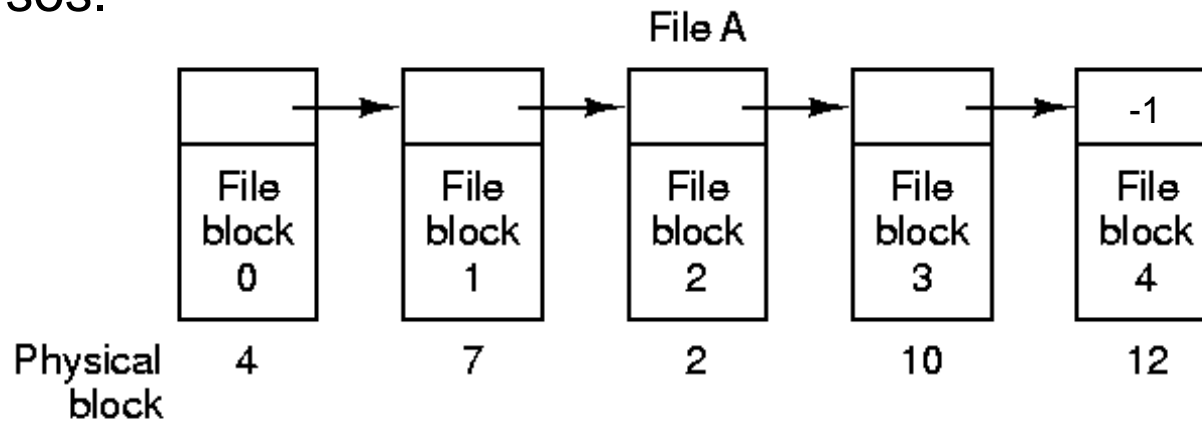
File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Listas enlazadas

- Utiliza la última parte del bloque como un apuntador al siguiente bloque, el archivo estará formado por una serie de bloques dispersos.



3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Listas enlazadas

- En el descriptor del fichero sólo es necesario almacenar el primer bloque.
- Problemas:
 - Es más complicado calcular el tamaño real del fichero.
 - El tamaño de la información del bloque ya no es potencia de 2.
 - El acceso directo es muy costoso de implementar (se recorre secuencialmente la lista).
 - Poco fiable, ¿qué pasa si perdemos un bloque?
- Solución al problema de eficiente: cargar los punteros a ficheros en memoria. → ÍNDICE ENLAZADO

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado

- Cuando se crea el SF se almacena en una parte especial del mismo una **tabla** que contiene una **entrada por cada bloque de disco** y que está **indexada por número de bloque**.

F A T

FAT (File Allocation Table)

x	x	EOF	13	2	9	8	FREE	4	12	3	FREE	EOF	EOF	FREE	BAD	
---	---	-----	----	---	---	---	------	---	----	---	------	-----	-----	------	-----	--

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

archivo A : 6 → 8 → 4 → 2

archivo B : 5 → 9 → 12

archivo C : 10 → 3 → 13

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado

- Cada vez que se crea un archivo se incluye en su **descriptor** (que estará contenido en el directorio donde se localiza el fichero) el índice de la tabla que apunta al **primer bloque del archivo**.

F A T

FAT (File Allocation Table)

x	x	EOF	13	2	9	8	FREE	4	12	3	FREE	EOF	EOF	FREE	BAD	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

archivo A : 6 → 8 → 4 → 2

archivo B : 5 → 9 → 12

archivo C : 10 → 3 → 13

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado

- A medida que se asignan **nuevos bloques** al archivo **se apunta a ellos desde la última entrada de la tabla** asociada al archivo:

F A T

FAT (File Allocation Table)

x	x	EOF	13	2	9	8	FREE	4	12	3	FREE	EOF	EOF	FREE	BAD	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

- ¿Dónde se almacena la FAT?
 - Inicialmente está en una ubicación específica de la partición.
 - Para ser operativas, tras el arranque, deberían mantenerse en memoria.

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado

- Ejemplo: Dada la siguiente FAT y entradas de directorio:

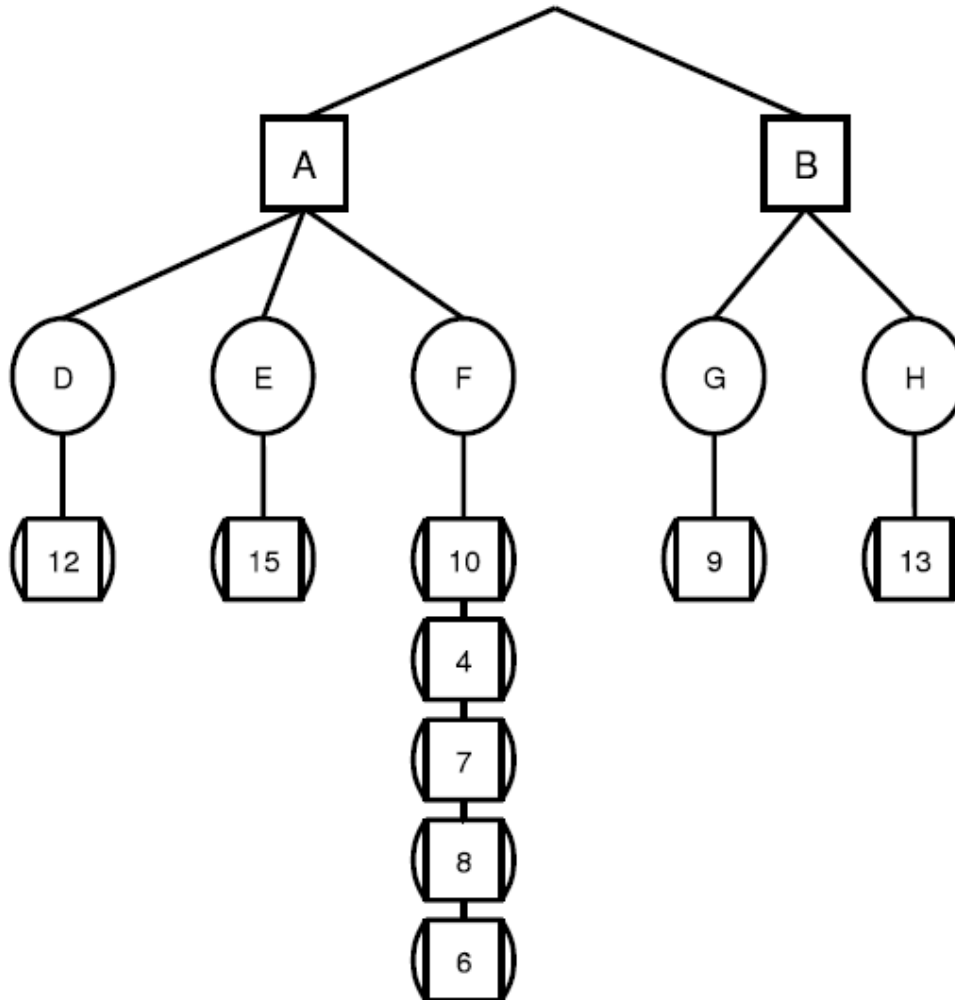
Raiz			Bloque 5			Bloque 2		
A	Dir	5	D	Dat	12	G	Dat	9
B	Dir	2	E	Dat	15	H	Dat	13
			F	Dat	10			

FAT	X	X	E O F	E O F	7	E O F	E O F	8	6	E O F	4	E O F	E O F	E O F	E O F	E O F
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

- ¿Cuál es el árbol de directorios asociado?. Incluya los bloques que ocupa cada fichero.

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado



3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado

- Algunas de estas evoluciones son:
 - **FAT 12:** Apuntadores de 12 bits, con lo que $2^{12} = 4096$ posibles bloques, permitiendo direccionar hasta 4 Megas como máximo. En MSDOS se usa un tamaño de bloque de 1K, la tabla FAT para los disco de 320K ocupa:

$$Tam\ FAT = 320 \times 12 / 8 = 480\ bytes$$

y 360K formateados con este sistema es:

$$Tam\ FAT = 360 \times 12 / 8 = 540\ bytes$$

- **FAT 16:** Apuntadores de 16 bits, con lo que $2^{16} = 65536$ posibles bloques, permitiendo direccionar hasta 64 Megas. En un disco con esta capacidad tendremos la FAT que ocupará:

$$Tam\ FAT = 65536 \times 16 / 8 = 128\ Kbytes$$

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado

- Problemas:
 - Ocupa bastante espacio de memoria, habrá que descargarla a disco, al menos parcialmente, con lo que pierde su eficacia (se puede aminorar el problema usando agrupaciones o clúster).
 - ¿Cómo podemos direccionar discos más grandes? Aumentando el tamaño del bloque (hasta un máximo de 32 kb → 2Gb).
- Ejemplo: Tamaño máximo de las particiones en sistemas FAT :

Tamaño bloque	FAT-12	FAT-16	FAT-32
0,5 KB	2 MB		
1 KB	4 MB		
2 KB	8 MB	128 MB	
4 KB	16 MB	256 MB	1 TB
8 KB		512 MB	2 TB
16 KB		1024 MB	4 TB
32 KB		2048 MB	8 TB



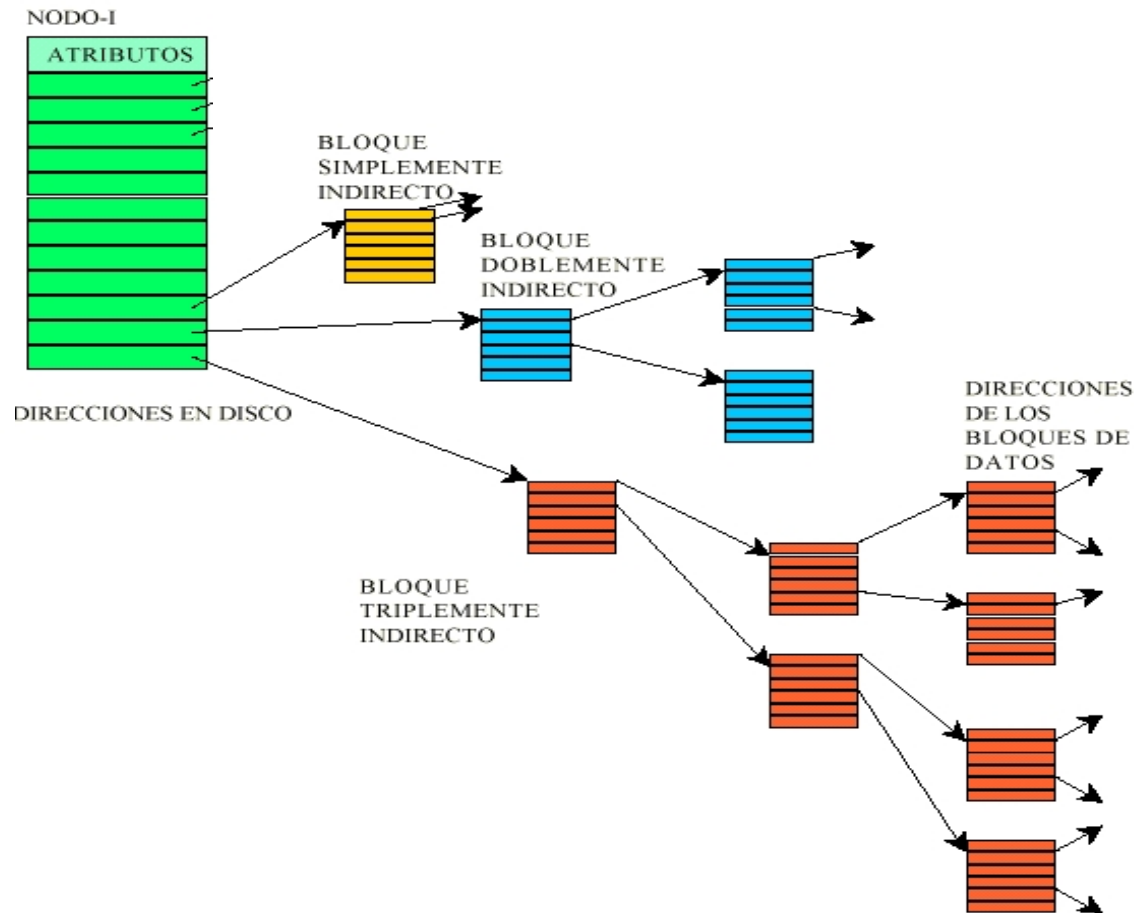
3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado multinivel

- Los problemas anteriores se solucionan usando una estructura que guarde la información del fichero.
- El nodo-i es una estructura de datos de almacenamiento en disco de UNIX / LINUX que contiene la información de un fichero.
 - ID de nodo-i.
 - ID del dispositivo
 - Tipo de fichero.
 - Número de enlaces.
 - UID del propietario.
 - GID del grupo.
 - Tamaño.
 - Fecha de creación.
 - Fecha de la última modificación y/o acceso.
 - Protección y/o permisos.
 - Otras informaciones.
 - 10 Apuntadores directos
 - Apuntador Indirecto Simple: Apunta a un bloque que contiene apuntadores a bloques de datos.
 - Apuntador Indirecto Doble: Apunta a un bloque que contiene apuntadores que apuntan a bloques que contienen apuntadores a bloques de datos.
 - Apuntador Indirecto Triple: Se añade un nivel más de apuntadores que en el caso anterior.

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado multinivel





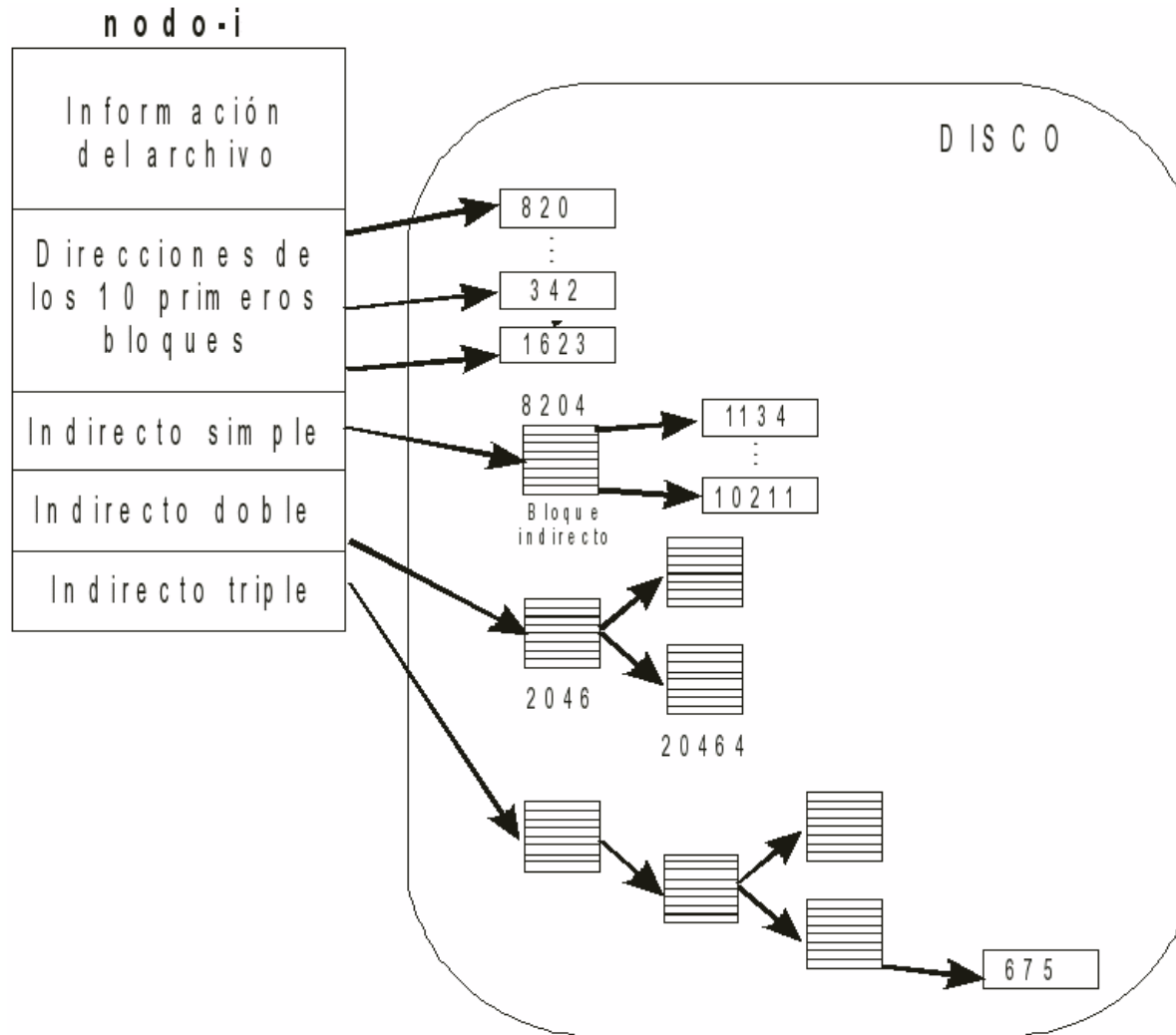
3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado multinivel

- Este esquema presenta un uso ineficiente de espacio para archivos pequeños ya que necesitamos crear un i-nodo y un bloque de índice.
- Para solucionar esto al nodo-i se le incluyó la siguiente información:
 - 10 Apuntadores Directos: Apuntan a bloques de datos.
- Se consiguen las siguientes ventajas:
 - Los apuntadores indirectos no se utilizan si no son necesarios.
 - La mayoría de los ficheros suelen tener menos de 10 Kbytes, por lo que no necesitamos apuntadores dobles.
 - El número máximo para cualquier referencia a disco son 3 (sin contar el I-NODE y el dato).

3. Diseño del sistema de ficheros

3.2 Almacenamiento de ficheros. Índice enlazado multinivel



3. Diseño del sistema de ficheros

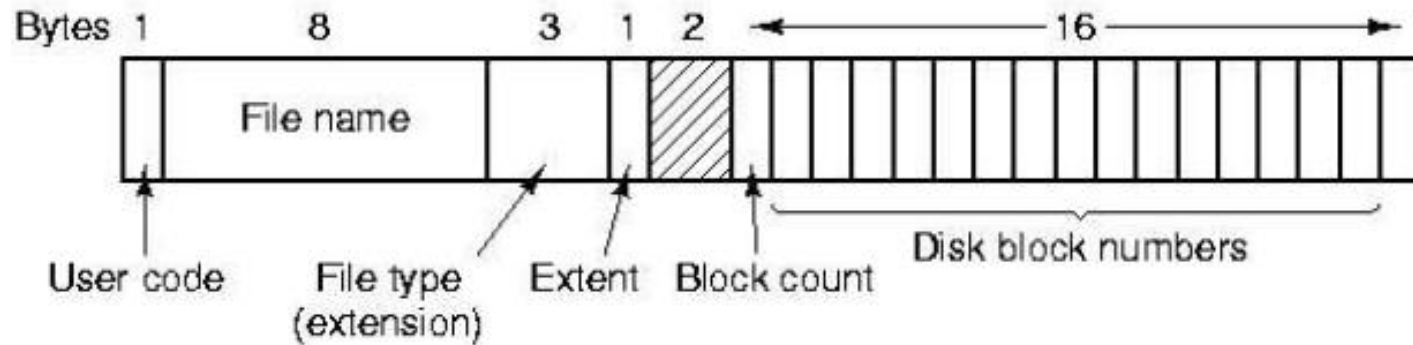
3.2 Almacenamiento de ficheros. Índice enlazado multinivel

- **Ejemplo:** Cuál será el tamaño máximo de un fichero en un SF que tiene un tamaño de bloque de 1 KB, usa un nodo-i con 10 apuntadores directos, 1 simple, 1 doble y 1 triple y las direcciones a disco son de 32 bits.
 - Por los 10 apuntadores libres: $10 \text{ bloques} * 1 \text{ kb por bloque} = 10 \text{ Kb}$
 - En cada bloque de 1k se pueden almacenar $1024/32=256$ direcciones de bloques.
 - Por el apuntador simple: $256 \text{ bloques de } 1 \text{ Kb cada uno} = 256 \text{ Kb}$
 - Por apuntador doble: $65.536 \text{ bloques de } 1 \text{ Kb} = 64 \text{ Mb}$
 - Por apuntador triple: $16.777.216 \text{ bloques de } 1 \text{ Kb} = 16 \text{ Gb}$
- **Total** = $10 \text{ kb} + 256 \text{ Kb} + 64 \text{ Mb} + 16 \text{ Gb} = 16.06 \text{ Gb}$.

3. Diseño del sistema de ficheros

3.3 Estructura del directorio. CP/M

- Estructura muy simple, sólo tiene un directorio, que es el único que existe. Al encontrar la entrada en el directorio ya tenemos también los números de bloques ya que todo se almacena ahí:



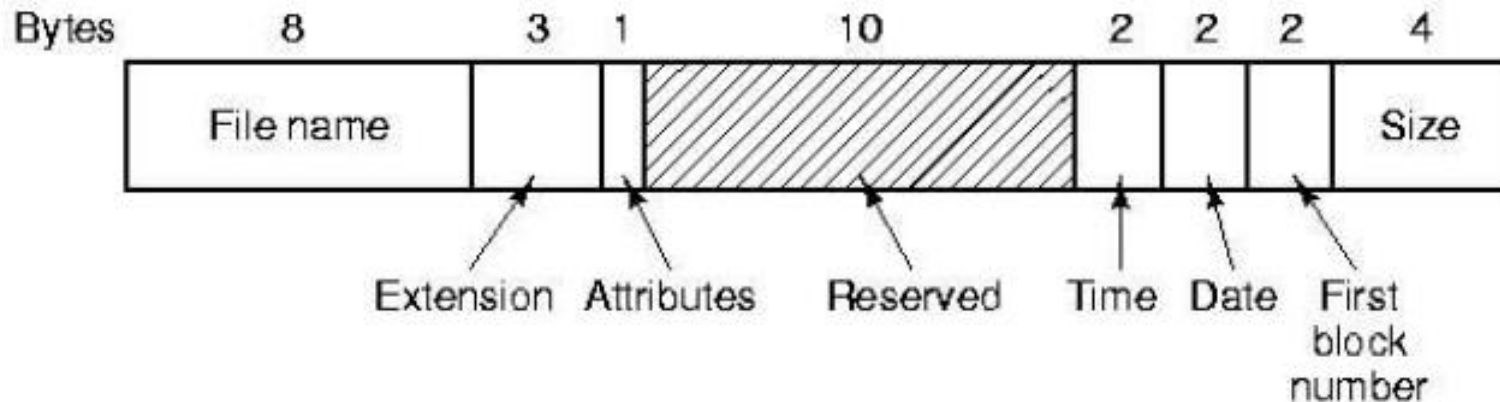
- En esta estructura, para poder almacenar más de 16 bloques de fichero utilizaremos otra entrada del directorio, con el mismo nombre , código de usuario, tipo, pero con un grado diferente.

A	txt	1	R	16	7	14	23	67	28	34	60	36	12	16	83	92	47	24	28	33	44
A	txt	2	R	2	5	8	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?

3. Diseño del sistema de ficheros

3.3 Estructura del directorio. MS-DOS

- Las entradas de directorio seguirán el siguiente esquema:

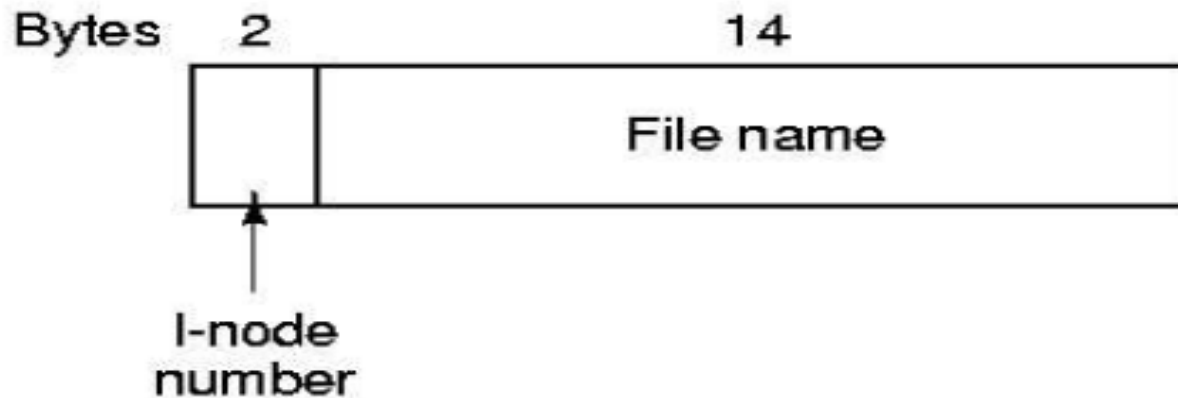


- Aparte de dos entrada fijas que hacen referencia al mismo directorio ('.') y al padre ('..') existirán tantas entradas como ficheros contenga el directorio.
- Un dato que debe tener en cuenta es que en MS-DOS el directorio raíz tiene un número de entradas fijo.
- Las entradas que empiezan con ? indican ficheros borrados.

3. Diseño del sistema de ficheros

3.3 Estructura del directorio. UNIX- LINUX

- Las entradas de directorio seguirán el siguiente esquema:

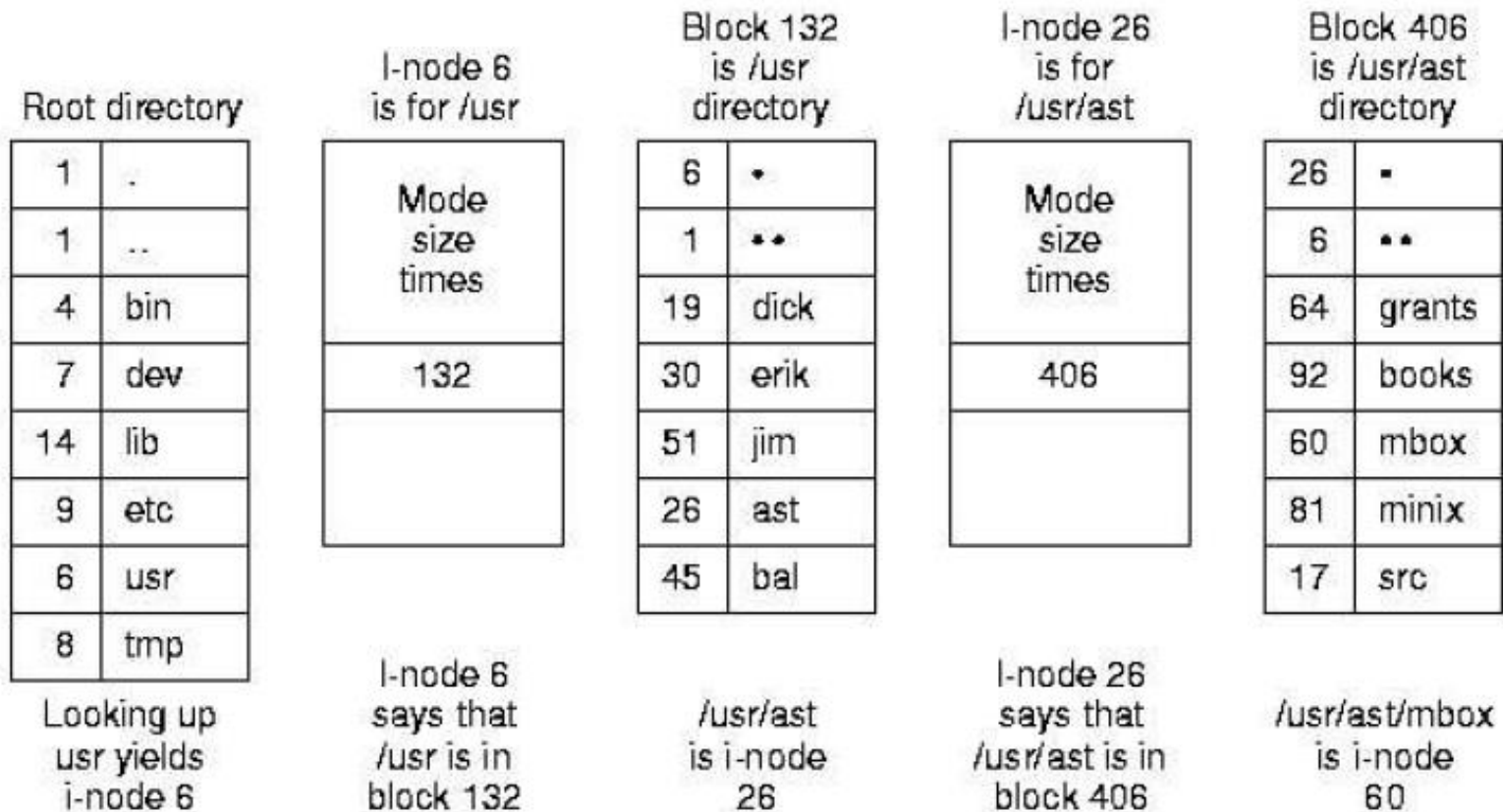


- Toda la información del archivo se guarda en el nodo i por lo que en UNIX un directorio es un caso especial de archivo que contiene registros con dos campos.
- Las entradas con los nombres '.' y '..' apuntan al propio directorio y al directorio padre respectivamente.

3. Diseño del sistema de ficheros

3.3 Estructura del directorio. UNIX- LINUX

- Pasos para buscar el fichero **/usr/ast/mbox**, la posición del i-nodo raíz es fija:



3. Diseño del sistema de ficheros

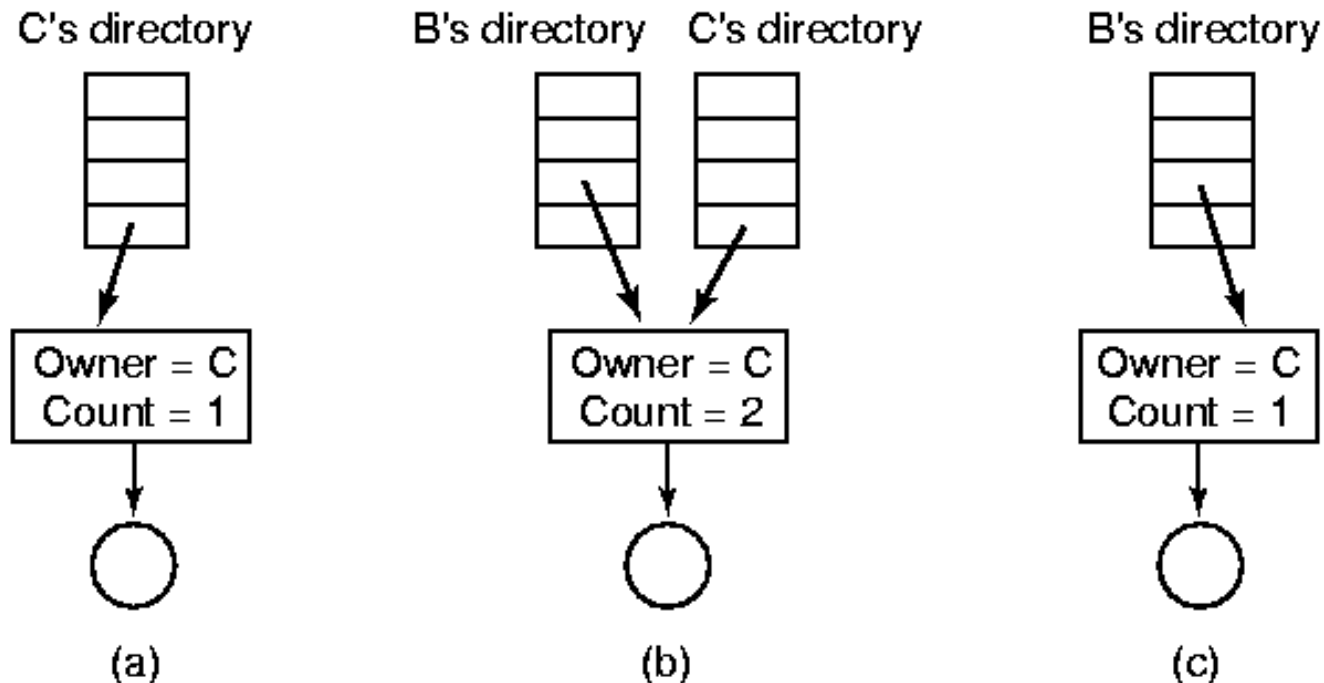
3.4 Compartir archivos

- Llamamos **enlace** o **link** a la capacidad de asociar un mismo archivo a dos o más directorios padres.
- De esta forma, una misma información puede ser accedida desde distintos puntos del árbol de directorios.
- La estructura de árbol vista hasta ahora pasa a ser un grafo acíclico (sin ciclos).
- En **Windows** nos podemos encontrar con los **accesos directos**, que serán una estructura parecida a la utilizada en otros sistemas operativos UNIX, concretamente un acceso directo es el equivalente a lo que vamos a denominar **enlace blando**.

3. Diseño del sistema de ficheros

3.4 Compartir archivos

- En UNIX nos podemos encontrar con dos tipos de enlaces:
 - **Enlace Duro (Hard Link):** En la entrada de directorio se tiene un apuntador a una estructura que guarda los datos relativos al fichero (apuntador a un i-nodo).



3. Diseño del sistema de ficheros

3.4 Compartir archivos

- En UNIX nos podemos encontrar con dos tipos de enlaces:
 - **Enlace Duro (Hard Link)**
 - Problema al realizar el borrado.
 - Cuando se borra un fichero, se deberá eliminar la entrada del directorio, liberar el i-node que ocupa y marcar los bloques de datos de dicho fichero como libres.
 - El otro enlace entonces apuntará a un i-nodo vacío.
 - Para corregir esto, al borrar un fichero se elimina su entrada del directorio y se decrementa en 1 el número de enlaces del i-nodo, y sólo si el resultado de esa operación es 0, se procede con el resto de las acciones
 - La única pega es la propiedad del fichero.



3. Diseño del sistema de ficheros

3.4 Compartir archivos

- En UNIX nos podemos encontrar con dos tipos de enlaces:
 - **Enlace Blando (Soft Link):** Se crea un nuevo fichero con la ruta al fichero al cual se enlaza. El nuevo fichero (enlace blando) es de un tipo especial (link) y su contenido es la ruta al fichero al cual apunta.
 - Cuando se intenta acceder al fichero de tipo link, se detecta que es un enlace blando y el acceso se realiza a la ubicación indicada por dicho fichero.
 - En este caso, si el fichero original es borrado, el enlace queda roto, con lo cual los futuros intentos de accesos provocarán un error.

3. Diseño del sistema de ficheros

3.4 Compartir archivos

- Comparativa entre ambos métodos:

	Hard Link	Soft Link
Ventajas	Ahorra espacio Más eficiente	Sin problemas al borrar
Inconvenientes	No se puede borrar si no tienes permisos	Más accesos a disco Ocupa más espacio en Disco

Índice



1. Introducción
2. Gestión del disco
3. Diseño del sistema de ficheros
- 4. Protección de archivos**
5. Casos de estudio

4. Protección de ficheros

- En el momento en que se almacena información dentro de un sistema informático, dicha información debe ser protegida:
 - Contra daños físicos (**fiabilidad**).
 - La fiabilidad se consigue realizando **copias de seguridad** o backups (lógicos y/o físicos) de los ficheros del sistema
 - Contra el acceso indebido (**protección**)
 - Los mecanismos de protección facilitan la labor de permitir un **acceso controlado** (permiten o deniegan acciones sobre objetos).
 - Mecanismos de protección:
 - **Espacio de nombres:** dependen de la imposibilidad de los usuarios para acceder a un fichero que no se puede nombrar.
 - **Contraseñas:** asociar una contraseña a cada fichero.
 - **Control de acceso:** el acceso depende de la identidad del usuario.

Índice



1. Introducción
2. Gestión del disco
3. Diseño del sistema de ficheros
4. Protección de archivos
- 5. Casos de estudio**



5. Casos de estudio

5.1 FAT16

- Fue inventada en 1977 por Bill Gates & Marc McDonald
- Hereda características de CP/M.
- Originalmente muy buena porque permitía a la FAT estar en memoria
- Las restricciones de los nombres son heredadas de CP/M.
- Comienza con FAT12, que puede direccionar $2^{12} = 4096$ bloques de disco, es decir con sectores de 512 bytes hasta 2 Mbytes.
- La FAT16 puede direccionar $2^{16} = 65535$ bloques de disco, es decir con sectores de 512 bytes hasta 32 Mbytes.
- Los discos mayores se pueden manejar agrupando en clusters (de 8 o 16 Kbytes). Para discos de 1 Gbyte necesitamos clusters de 32 Kbytes.

5. Casos de estudio

5.1 FAT16

- Hay dos copias de la FAT. La segunda sólo se emplea si la 1ª no se puede leer (error físico del disco)
- El directorio / tiene un tamaño máximo. (En un disquete, un fichero ocupa como mínimo 512 bytes, y caben 224 archivos en el directorio /).
- Las entradas de los directorios no están ordenadas.
- El directorio / y la FAT están al comienzo del disco (borde exterior).
- La estructura de una partición FAT es la siguiente:

BOOT	1º COPIA DE LA FAT	2º COPIA DE LA FAT	DIRECTORIO RAÍZ	DATOS Y DIRECTORIOS
------	-----------------------	-----------------------	--------------------	---------------------

5. Casos de estudio

5.2 FAT32

- Surge con la aparición del sistema operativo **Windows 95**
- Aprovechamiento más eficiente del espacio de disco.
- FAT32 es un sistema de ficheros más robusto y flexible.
- Minimiza el efecto de la fragmentación de archivos.
- El principal cambio de una FAT a otra radica en la ampliación del tamaño de las entradas, que pasa de 16 a **32** bits.
- Con una entrada de FAT de 32 bits se permitirían discos de casi 4 Tbytes.
- FAT32 usa clústers más pequeños, en concreto de 4 Kb, por lo que la eficiencia aumenta y el espacio desperdiciado disminuye.

5. Casos de estudio

5.2 FAT32

- FAT32 tiene la habilidad de **recolocar el directorio raíz** y utilizar la copia de seguridad de la FAT de forma efectiva.
- El boot record ha sido expandido para incluir una copia de seguridad de los datos críticos del sistema.
- El directorio raíz en un volumen FAT32 es ahora una simple cadena de clústers, que puede ser localizada en cualquier lugar de la unidad. Por esta razón, la limitación en FAT16 sobre el número de entradas del directorio raíz ahora ha desaparecido.
- **La copia espejo de la FAT puede ser deshabilitada**, permitiendo que otra copia de una FAT sea considerada como activa.

5. Casos de estudio

5.3 NTFS

- Es el sistema de archivos de Windows ideado para aprovechar al máximo **discos grandes y procesadores rápidos**.
- Significa **New Technology File System**.
- Proporciona una combinación de fiabilidad y compatibilidad que no se encuentran en FAT.
- Desde el punto de vista del sistema de ficheros, todo en un volumen NTFS es un fichero o parte de él.
- La estructura de un volumen NTFS básicamente es la siguiente:

BOOT	MFT (TABLA MAESTRA DE FICHEROS)	AREA DE DATOS
------	------------------------------------	---------------



5. Casos de estudio

5.3 NTFS

- **BOOT**

- El boot puede ocupar varios sectores. (hasta 16 sectores de longitud).
- Contiene la disposición del volumen y la estructura del sistema de ficheros
- Hay un duplicado del boot en el centro lógico del disco.
- Contiene la localización de la MFT.
- Contiene el código de arranque.

5. Casos de estudio

5.3 NTFS

- **MFT (MASTER FILE TABLE)**

- La (MFT), es una lista de **todos los contenidos de este volumen** NTFS organizada como un conjunto de filas en una estructura de base de datos relacional.
- Contiene al menos una **entrada por cada fichero existente** en el volumen, incluyendo la propia MFT.
- Suele ocupar el **12,5%** del espacio disponible en el volumen.
- Es la zona donde se guarda la información sobre los ficheros y directorios almacenados en el volumen.
- Está **organizada en forma de tabla**, donde cada entrada contiene información sobre un fichero o directorio.
- La MFT **almacena información sobre sí misma**, las 16 primeras entradas están reservadas para información 'especial' (metainformación).

5. Casos de estudio

5.3 NTFS

- MFT**
(0-15)

Fichero	Nombre	Registro de la MFT	Descripción del fichero
Master file table	\$Mft	0	Autodescribe a la propia MFT
Master file table 2	\$MftMirr	1	Es un duplicado de los primeros registros de la MFT. Esto garantiza el acceso a la MFT en caso de fallo en el sector.
Log file	\$LogFile	2	Contiene información sobre las transacciones. Se usa para recuperaciones rápidas tras fallos del sistema. El tamaño dependerá del tamaño del volumen.
Volume	\$Volume	3	Contiene información sobre el volumen, como la etiqueta del volumen o la versión.
Attribute definitions	\$AttrDef	4	Son los nombres de los atributos, su número y su descripción.
Root file name index	\$	5	Es el directorio raíz.
Cluster bitmap	\$Bitmap	6	Mapa de bits de los clusters del volumen.
Boot sector	\$Boot	7	Incluye información para montar el volumen y el código de arranque.
Bad cluster file	\$BadClus	8	Es la lista de clusters defectuosos del volumen.
Security file	\$Secure	9	Contiene el descriptor de seguridad para todos los archivos dentro de un volumen.
Uppcase table	\$Uppcase	10	Convierte los caracteres minúsculas para adaptarlos a los caracteres mayúsculas Unicode.
NTFS extension file	\$Extend	11	Usado para extensiones opcionales, tales como cuotas, identificadores de objetos y datos sobre puntos de recuperación.
		12-15	Reservado para futuros usos.

5. Casos de estudio

5.3 NTFS

- **MFT (MASTER FILE TABLE). FICHEROS**

- A partir de la entrada 17ª, están colocados el resto de archivos y directorios.
- La MFT reserva una cantidad de espacio para cada registro del fichero. Los atributos del fichero son escritos en ese espacio dentro de la MFT. Los ficheros y directorios pequeños (habitualmente 1500 bytes o menos) pueden ser ubicados completamente dentro de su entrada de la MFT.
- Cada entrada de la MFT contiene la siguiente información:

Cabecera	Información Estándar	Nombre del fichero	Descriptor de seguridad	Datos
----------	----------------------	--------------------	-------------------------	-------

- La cabecera es el número de entrada en la MFT

5. Casos de estudio

5.3 NTFS

- **MFT (MASTER FILE TABLE). FICHEROS**

- Si los datos del fichero caben dentro de la entrada correspondiente de la MFT se colocan en el área de Datos.
- Si no, se saca la información de la MFT de la siguiente forma:
 - Se hacen los **datos no residentes**. Para ello la entrada de datos de la MFT no contiene los datos del fichero, sino **punteros a extensiones**. **Una extensión es un conjunto de bloques (clústers) consecutivos**, que contienen los datos del fichero. Cada puntero a una extensión contiene un apuntador al **longitud de la misma y al 1º bloque de la extensión**. Como son contiguas no es necesario leer uno para saber cual es el siguiente. Con esto se reduce la fragmentación.
 - Si no hay suficiente espacio para almacenar todos los punteros, se colocan punteros a otros registros de la MFT que tendrán apuntadores a las extensiones. En este caso, el primer registro para el fichero (el registro base del fichero) contiene la localización del resto de los registros.

5. Casos de estudio

5.3 NTFS

- **MFT (MASTER FILE TABLE). DIRECTORIOS**
 - **Los atributos y características de los ficheros se guardan en los propios ficheros**, y no en los directorios. Los directorios sólo guardan información del directorio.
 - Los directorios son considerados como ficheros y tienen sus propias entradas en la MFT.
 - La entrada del directorio es similar a la del fichero:

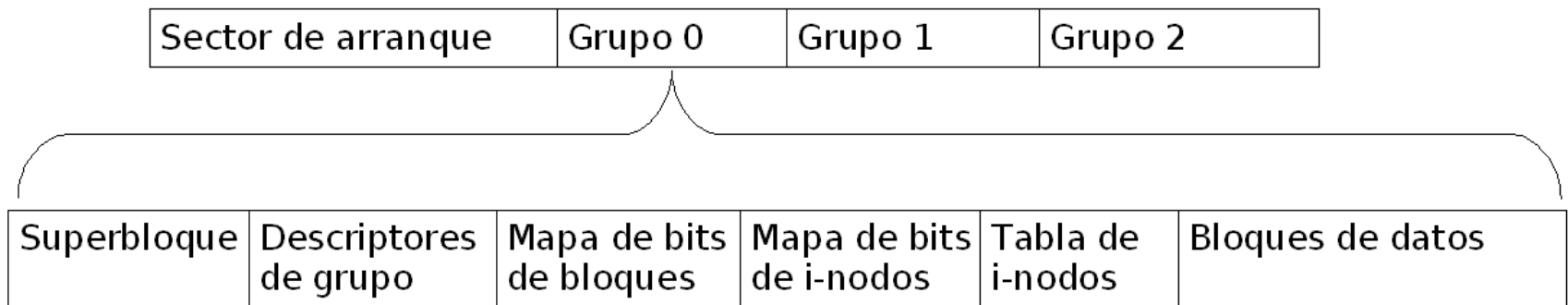
Cabecera	Información Estándar	Nombre del directorio	Descriptor de seguridad	Indices
----------	-------------------------	--------------------------	----------------------------	---------

- En lugar de almacenar datos, los directorios **almacenan los índices (nº de cabecera) de los ficheros que contienen.**
- Para mejorar la búsqueda en directorios grandes, NTFS almacena la información de las entradas del directorio en forma de **árbol-B**, que localiza la información mucho más rápidamente.

5. Casos de estudio

5.4 EXT2 Y EXT3

- Se trata de los sistemas de archivos más empleados en Linux. Ext3 es básicamente un sistema de archivos Ext2 al que se le ha añadido un registro de transacciones. Nos referiremos a ellos de forma genérica como Ext.



- Los primeros 1024 bytes del volumen están reservados. Pueden contener el código de arranque del volumen cuando el sistema operativo arranca desde el volumen.
- Los bloques se organizan en grupos, conteniendo todos ellos el mismo número de bloques, excepto el último que puede ser más pequeño.

5. Casos de estudio

5.4 EXT2 Y EXT3

Superbloque	Descriptores de grupo	Mapa de bits de bloques	Mapa de bits de i-nodos	Tabla de i-nodos	Bloques de datos
-------------	-----------------------	-------------------------	-------------------------	------------------	------------------

- El **superbloque** tiene un tamaño de 1024 bytes (aunque muchos de estos bytes no los usa) y contiene los parámetros del sistema de archivos.
- El superbloque contiene **información** como el **tamaño del bloque**, el **número de bloques** del sistema, **nombre** del volumen, **fecha** y **hora** en la que se montó por última vez el volumen, etc.

5. Casos de estudio

5.4 EXT2 Y EXT3

Superbloque	Descriptores de grupo	Mapa de bits de bloques	Mapa de bits de i-nodos	Tabla de i-nodos	Bloques de datos
-------------	-----------------------	-------------------------	-------------------------	------------------	------------------

- Es una tabla de tamaño **1 bloque** situada después del superbloque y comenzando en un bloque nuevo.
- Contiene entradas de 32 bytes que **describen el grupo** del que forma parte. Cada una de estas entradas proporciona:
 - Número de bloque de comienzo del mapa de bits de bloques,
 - Número de bloque de comienzo del mapa de bits de i-nodos
 - Número de bloque de comienzo de la tabla de i-nodos
 - Número de bloques que aún no han sido asignados en el grupo.
 - Número de i-nodos que aún no han sido asignados en el grupo.
 - Número de directorios en el bloque.

5. Casos de estudio

5.4 EXT2 Y EXT3

Superbloque	Descriptores de grupo	Mapa de bits de bloques	Mapa de bits de i-nodos	Tabla de i-nodos	Bloques de datos
-------------	-----------------------	-------------------------	-------------------------	------------------	------------------

- La contabilidad de los bloques que han sido asignados dentro de cada grupo se lleva a cabo con el **mapa de bits de bloques**. El mapa de bits de bloques comienza justo después de la tabla de descriptores del grupo y tiene un tamaño de **1 bloque**.
- El **mapa de bits de i-nodos** tiene un tamaño de **1 bloque** y funciona de forma análoga al mapa de bits de bloques. Cada entrada de la tabla de i-nodos tiene asociado un bit que se activa cuando la entrada es usada o se trata de una entrada reservada.

5. Casos de estudio

5.4 EXT2 Y EXT3

Superbloque	Descriptores de grupo	Mapa de bits de bloques	Mapa de bits de i-nodos	Tabla de i-nodos	Bloques de datos
-------------	-----------------------	-------------------------	-------------------------	------------------	------------------

- La tabla de i-nodos comienza en el bloque indicado en el descriptor del grupo y tiene un **tamaño especificado en el superbloque**.
- Cada una de sus entradas es un i-nodo que hace referencia a un archivo o directorio.
- **Las primeras entradas de la tabla están reservadas.** Por ejemplo, la entrada número 2 contiene el i-nodo del directorio raíz.

5. Casos de estudio

5.5 Sistema de ficheros con *journaling*

- Para **mejorar el rendimiento** de las operaciones de **E/S**, los datos del disco son temporalmente almacenados en la **memoria RAM** a través del **page-cache** y **buffer-cache** (caso de linux).
- **Problema:** Si hay un corte de suministro eléctrico antes que los datos modificados en la memoria (dirty buffers) sean grabados nuevamente al disco, se producen inconsistencias
- Soluciones:
 - Herramienta *fsck* (*file system check*). Es lenta, debe comprobar toda la partición
 - Añadir *journaling* al sistema de ficheros

5. Casos de estudio

5.5 Sistema de ficheros con *journaling*

- Un sistema con *journaling* es un sistema de ficheros **tolerante a fallos** en el cual la integridad de los datos está asegurada porque las modificaciones de la meta-información de los ficheros es primero grabada en un **registro cronológico** (log o journal) antes que los bloques originales sean modificados.
- En el caso de un fallo del sistema el módulo de recuperación analizará el registro y **sólo repetirá las operaciones incompletas** en aquellos ficheros inconsistentes, es decir que la operación registrada no se haya llevado a cabo finalmente.
- **Desventajas** de los sistemas con journaling:
 - Se realizan más operaciones de entrada/salida.
 - Como consecuencia de la desventaja anterior se produce más fragmentación.

5. Casos de estudio

5.5 Sistema de ficheros con *journaling*

- Algunos sistemas de ficheros que soportan journaling:
 - En Linux
 - ReiserFS (de Namesys).
 - XFS (de Silicon Graphics).
 - JFS (de IBM).
 - Ext3 (de Stephen Tweedie).
 - En MACOS:
 - Apple's HPS
 - En Solaris:
 - UFS
 - En Windows:
 - NTFS

Dudas



CONSEJO:

No dejar para
el final la realización
del test del tema
publicado en Moodle