



Sistemas Operativos 2021-2022

TEMA 5

Gestión de ficheros

5.1. Introducción

5.2. Ficheros

5.3. Directorios

5.4. Estructuras de datos para la gestión de ficheros

5.5. Implementación del sistema de ficheros



Introducción

Sistema de gestión de ficheros

- Módulo del SO encargado del manejo y organización de los ficheros
 - **En el nivel superior: Implementa operaciones** y estructuras de datos necesarios para hacer corresponder la visión del usuario programador con el sistema físico de almacenamiento.
 - Operaciones de ficheros (open, close, read, write, etc)
 - **En el nivel inferior: Administra la memoria secundaria** y estructura el (los) **sistema(s) de ficheros** (se definirá más adelante) sobre ella
 - ¿Cómo está organizado un disco duro?

5.2.1. Nombrado

5.2.2. Estructura lógica de los ficheros

5.2.3. Tipos de ficheros

5.2.4. Acceso a ficheros



Ficheros Nombrado

- Cuando un proceso crea un fichero le **asigna un nombre**
 - Otros procesos pueden acceder a él utilizando dicho nombre
- Las reglas de construcción de nombres varían entre SO

Estructura lógica de los ficheros

- **Secuencia no estructurada de bytes**
 - **El SO sólo ve bytes (UNIX y Windows)**
 - Cualquier estructura interna será impuesta por los procesos de usuario
 - Flexibilidad máxima
 - Pero el SO no ayuda
 - Operación básica
 - Leer/escribir N bytes



Ficheros

Tipos de ficheros

- Muchos SO soportan varios tipos de ficheros (UNIX y Windows, por ejemplo)
 - **Regulares**
 - Contienen información del usuario
 - ASCII/texto
 - Binarios
 - Estructura interna conocida por los programas que los usan
 - Ejecutables: estructura interna conocida por el SO
 - **Directorios**: estructura interna conocida por el SO
 - **Especiales** (UNIX)
 - De carácter
 - De bloque



Ficheros

Acceso a ficheros

■ Acceso secuencial

- La información se procesa en orden (byte o registro)
 - El fichero se puede “rebobinar”
- La lectura/escritura de un byte/registro avanza automáticamente el puntero en el fichero

■ Acceso directo

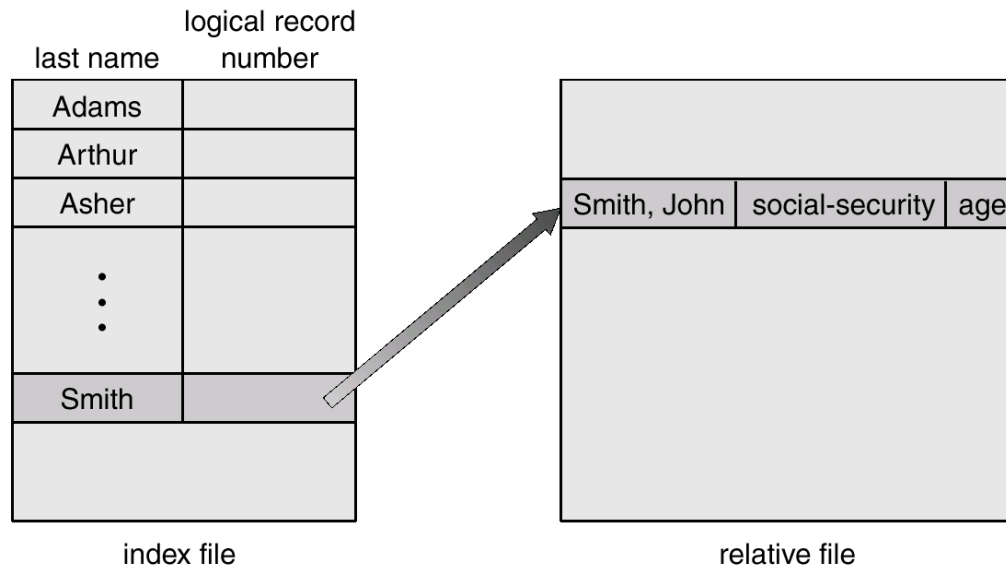
- Con dispositivos de acceso directo
- Acceso directo a cualquier byte/registro dada su dirección lógica/relativa
 - Especificada en la propia operación de lectura/escritura
 - O bien, operación especial para cambiar la posición del puntero en el fichero
- Compatible en muchos SO con el acceso secuencial

Ficheros

Acceso a ficheros

■ Acceso indexado

- Construido sobre el método de acceso directo
- Se construye(n) fichero(s) de índice(s) para el fichero
 - Relacionan valores con direcciones lógicas
 - Para acceder al fichero, se busca en el índice



5.3.1. Introducción

5.3.2. Sistema de directorios jerárquico



Directorios

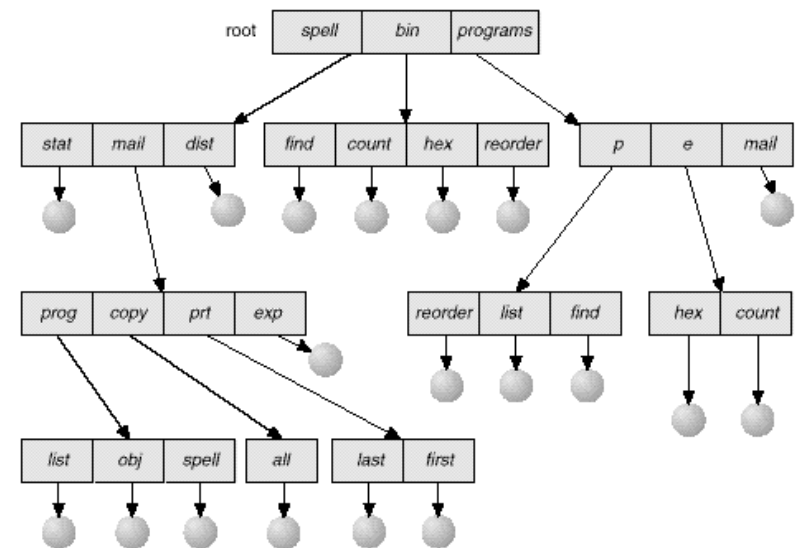
Introducción

- Son ficheros en muchos SO
- Desde el punto de vista del usuario, los directorios proporcionan una correspondencia entre el nombre de un fichero y el fichero propiamente dicho
- Formados por un conjunto de entradas, cada una relativa a un fichero contenido en el directorio

Directorios

Sistema de directorios jerárquico

- Cada entrada de un directorio apunta a un fichero o directorio
- El usuario puede definir la estructura de directorios
- Directorio de trabajo
- Nombres de directorio especiales
 - "." y ".."
- Nombres de camino
 - Absoluto y relativo





Estructuras de datos para la gestión de ficheros

Índice

- 5.4.1. El bloque descriptor de un fichero (BDF)
- 5.4.2. Tabla de ficheros abiertos del sistema
- 5.4.3. Tabla de BDFs en memoria principal
- 5.4.4. Tabla de ficheros abiertos por proceso
- 5.4.5. Llamadas al sistema para la gestión de ficheros.
- 5.4.6. Llamadas al sistema para la gestión de directorios.



Estructuras de datos para la gestión de ficheros

Bloque descriptor de fichero

- Es el equivalente al PCB de los procesos pero para ficheros.
- Guarda TODA la información necesaria para el SO para ese fichero (¡OJO! Aquí no van los datos del fichero)
- Para hacer cualquier gestión con un fichero se necesita PRIMERO obtener su BDF
- Los BDFs de todos los ficheros de un sistema de ficheros (partición) se almacenan en el propio sistema de ficheros (normalmente al principio)
- Sólo estarán en MP los BDFs de los ficheros abiertos (operación *open*)



Estructuras de datos para la gestión de ficheros

Bloque descriptor de fichero

- Información normalmente almacenada en un BDF:
 - Identificador interno
 - Número de nombres
 - Tipo
 - *Ubicación en el dispositivo de almacenamiento de los datos del fichero*
 - Tamaño
 - Protección
 - Instantes de creación, última modificación y último uso
 - Propietario(s)



Estructuras de datos para la gestión de ficheros

Tabla de ficheros abiertos del sistema

- La **tabla de ficheros abiertos del sistema** almacena información de todos los ficheros abiertos en el SO
 - **Solo hay una en el sistema** y reside en MP
 - Cada entrada almacena la siguiente información:
 - Directa o indirectamente una copia del BDF del fichero
 - Punteros de lectura/escritura. Se pueden compartir o NO compartir
 - N^o veces que se comparte la entrada (por ejemplo -1 si no)
 - Si no se comparten los punteros de lectura/escritura de un mismo fichero → Se crea SIEMPRE una entrada nueva
 - Si se comparten los punteros de lectura/escritura se utiliza la entrada compartida
 - Los distintos modos de apertura de un fichero se consideran independientes (L, E, L/E)

Estructuras de datos para la gestión de ficheros

Tabla de ficheros abiertos del sistema

open(modo,path,shared)

P1: open("r","fichero1",1)

P1: open("r","fichero1",0)

P2: open("rw","fichero2",1)

P2: open("r","fichero1",1)

P1: open("w","fichero2",1)

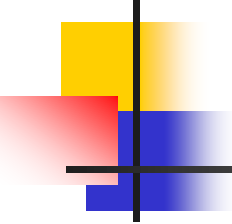
P. L	N Comp: 2	BDF1 ó
P. L	N Comp: -1	BDF1 ó
P. L/E	N Comp: 1	BDF2 ó
P. E	N Comp: 1	BDF2 ó

BDF1

BDF1

BDF2

BDF2



Estructuras de datos para la gestión de ficheros

Tabla de BDFs en memoria principal

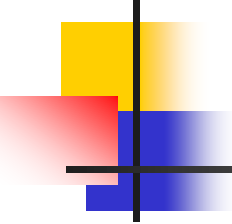
- La **tabla de BDFs en memoria principal** mantiene una copia de los BDFs de fichero en memoria secundaria
 - Solo hay una en el sistema y es opcional (no todos los SO la tienen)
 - El objetivo principal es evitar duplicar información en la tabla anterior (evita tener varias copias del mismo BDF)
 - Solo guarda copia de los ficheros que estén abiertos por algún proceso
 - Si un fichero es abierto varias veces solo hay una copia de su BDF en esta tabla.



Estructuras de datos para la gestión de ficheros

Tabla de BDFs en memoria principal

- Cada entrada tiene dos campos
 1. El BDF del fichero
 2. Un contador de referencias a la entrada → N° de entradas de la tabla de ficheros abiertos que apuntan a esta entrada.
 - Cuando el contador llega a cero se libera la entrada.



Estructuras de datos para la gestión de ficheros

Tabla de BDFs en memoria principal

Cuando NO HAY Tabla de BDFs en MP:

P. L	N Comp: 2	BDF1
P. L	N Comp: -1	BDF1
P. L/E	N Comp: 1	BDF2
P. E	N Comp: 1	BDF2

Tabla Ficheros abiertos

Estructuras de datos para la gestión de ficheros

Tabla de BDFs en memoria principal

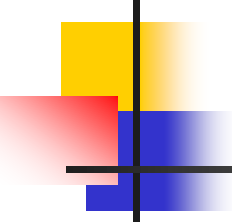
Cuando HAY Tabla de BDFs en MP:

P. L	N Comp: 2	
P. L	N Comp: -1	
P. L/E	N Comp: 1	
P. E	N Comp: 1	

Tabla Ficheros abiertos

2	BDF1
2	BDF2

Tabla de BDFs en MP



Estructuras de datos para la gestión de ficheros

Tabla de ficheros abiertos por proceso

- La **tabla de ficheros abiertos por proceso** almacena información de todos los ficheros abiertos por un proceso
 - Hay tantas como procesos y residen en MP
 - En mucho SO las tres primeras entradas están ya ocupadas y tienen un significado “especial”
 - Cada entrada almacena la siguiente información:
 - Un puntero a la entrada correspondiente en la *Tabla de ficheros abiertos del sistema*
 - Algunos SO ponen aquí los punteros de lectura/escritura en lugar de la tabla de ficheros abiertos del sistema
 - Complica compartir ficheros abiertos



Estructuras de datos para la gestión de ficheros

Ejemplo

Veamos un ejemplo de cómo quedarían las TRES estructuras:

P1: open("r","fichero1",1)

P1: open("r","fichero1",0)

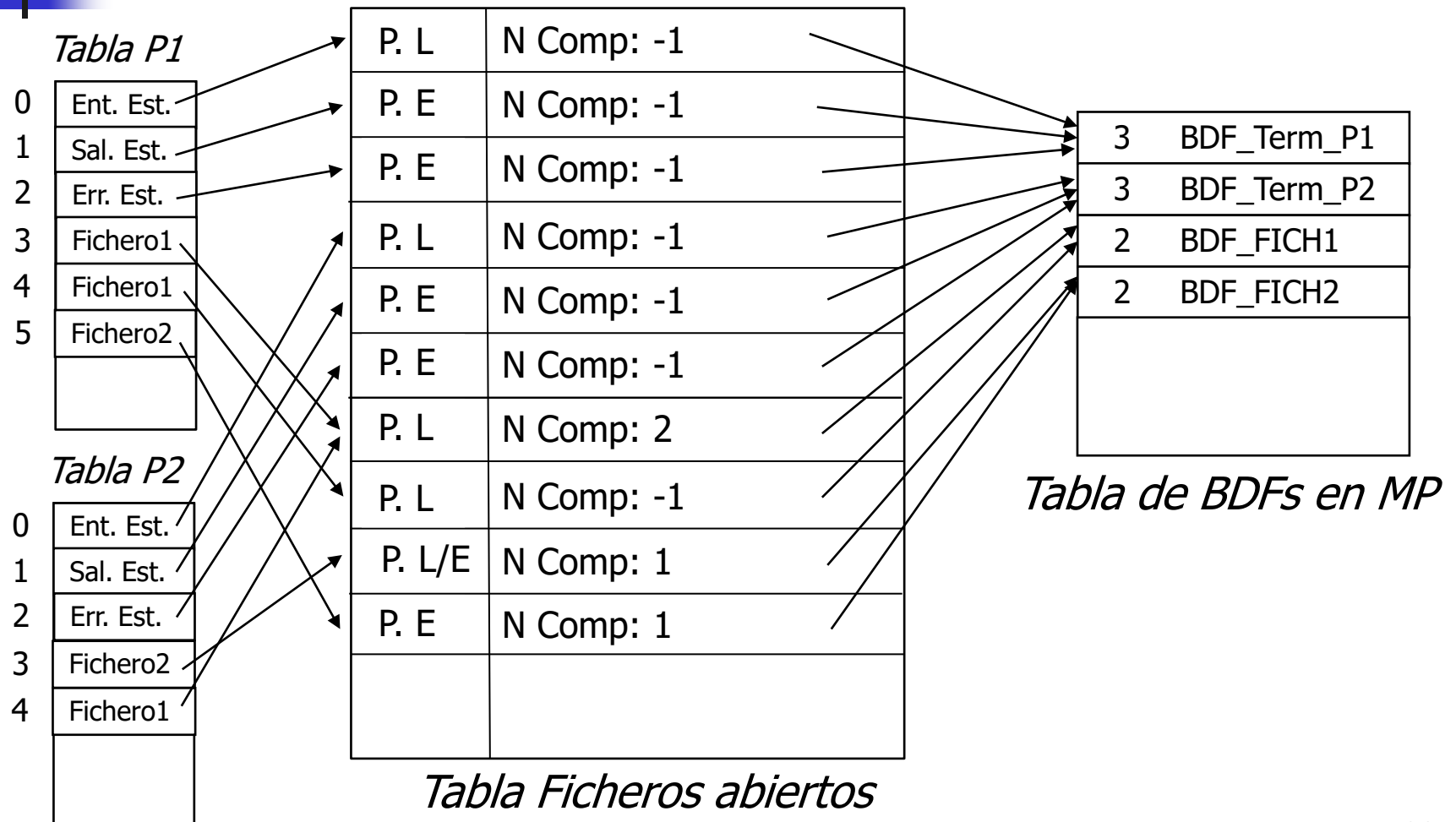
P2: open("rw","fichero2",1)

P2: open("r","fichero1",1)

P1: open("w","fichero2",1)

Estructuras de datos para la gestión de ficheros

Ejemplo



Llamadas al sistema para la gestión de ficheros

- Crear fichero
 - Se asignan BDF y valores iniciales para los atributos
- Borrar fichero
 - Se elimina el fichero y su BDF
- Abrir fichero (no shared)
 1. Se crea una entrada en la tabla de ficheros abiertos del sistema (campo *Nº comp* con -1)
 2. Si hay Tabla de BDFs:
 - ❖ Si existe entrada con BDF → Incrementa *cont. de ref.*
 - ❖ Si no existe entrada → Se crea y *cont. de ref.* a 1
 - ❖ Entrada paso 1 apuntando a esta entrada
 3. Se crea una entrada en la tabla de ficheros abiertos del proceso apuntando a entrada paso 1 y retorna nº de entrada → **descriptor de fichero**

Llamadas al sistema para la gestión de ficheros

- Abrir fichero (shared)

1A Si existe entrada en la tabla de ficheros abiertos del sistema → Incrementa campo *Nº comp.*

1B Si no existe → Se crea y campo *Nº comp.* a 1

2B Si hay Tabla de BDFs:

- ❖ Si existe entrada con BDF → Incrementa *cont. de ref.*
- ❖ Si no existe entrada → Se crea y *cont. de ref.* a 1
- ❖ Entrada paso 1B apuntando a esta entrada

2. Se crea una entrada en la tabla de ficheros abiertos del proceso apuntando a entrada paso 1 y retorna nº de entrada → **descriptor de fichero**

Llamadas al sistema para la gestión de ficheros

- Cerrar fichero (no shared)
 1. Se libera entrada en la tabla de ficheros abiertos del sistema
 2. Si hay Tabla de BDFs, en la entrada apuntada por paso 1:
 - ❖ Decrementa *cont. de ref.* → Se libera entrada si llega a 0
 3. Se libera entrada en la tabla de ficheros abiertos del proceso

Llamadas al sistema para la gestión de ficheros

- Cerrar fichero (shared)
 1. Decrementa *Nº comp* de la tabla de ficheros abiertos del sistema. Si llega a cero:
 - ❖ Libera la entrada de la tabla de ficheros abiertos del sistema.
 - ❖ Si hay Tabla de BDFs, en la entrada apuntada por paso 1:
 - ❖ Decrementa *cont. de ref.* → Se libera entrada si llega a 0
 2. Se libera entrada en la tabla de ficheros abiertos del proceso
- Leer, escribir, añadir, posicionarse
- Obtener y actualizar atributos
- Renombrar, truncar

Llamadas al sistema para la gestión de directorios

- Crear directorio
 - Se asignan BDF y valores iniciales para los atributos y directorios especiales "." y ".."
- Borrar, renombrar
- Abrir, cerrar
- Leer
- Enlazar (*link*), desenlazar (*unlink*)
 - Crean y eliminan entradas en el directorio
 - No implican, necesariamente, la creación de ficheros
- Obtener y actualizar atributos
- Algunas de las llamadas anteriores pueden presentarse de manera única para ficheros y directorios



Implementación del sistema de ficheros

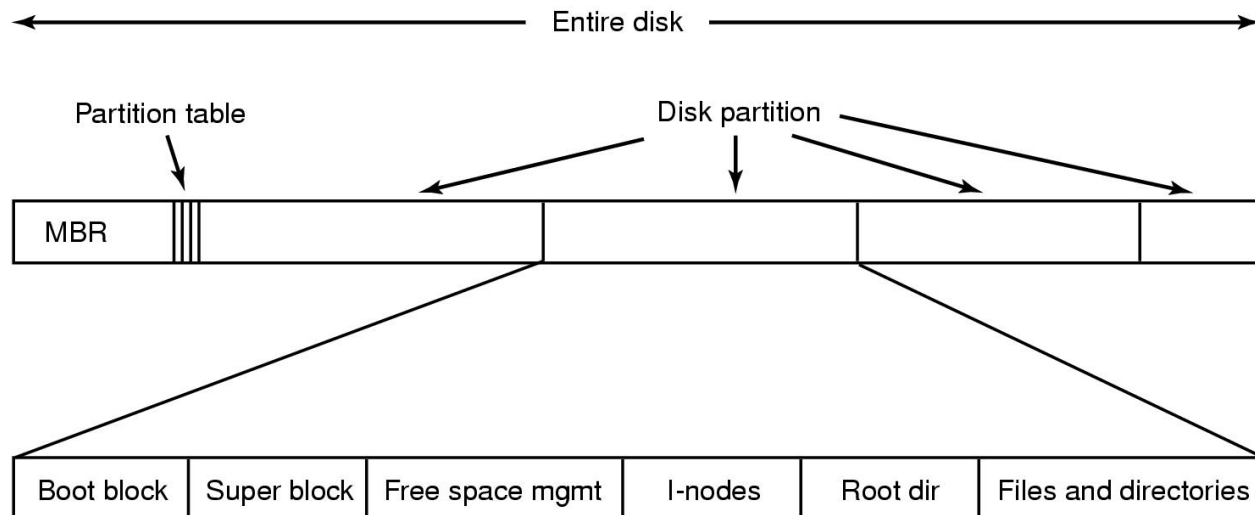
Índice

- 5.5.1. Estructura del sistema de ficheros
- 5.5.2. Implementación de ficheros
- 5.5.3. Implementación de directorios
- 5.5.4. Gestión del espacio libre
- 5.5.5. Consistencia del sistema de ficheros

Implementación del sistema de ficheros

Estructura del sistema de ficheros

- Un **sistema de ficheros** es un modo de organización de la información (ficheros) en memoria secundaria
 - Cada partición de un dispositivo de almacenamiento puede contener un sistema de ficheros
 - Para ello, es necesario **dar formato** a la partición
 - *format*, *mkfs* u otra herramienta





Implementación del sistema de ficheros

Estructura del sistema de ficheros

- Tamaño de bloque de datos
 - La mediana del tamaño de ficheros varía entre 1KB y 4KB (en función del sistema de ficheros)
 - Bloque grande
 - Fragmentación interna grande
 - Bloque pequeño
 - Más accesos al disco para acceder al fichero
 - Mayor tamaño de las estructuras de datos



Implementación del sistema de ficheros

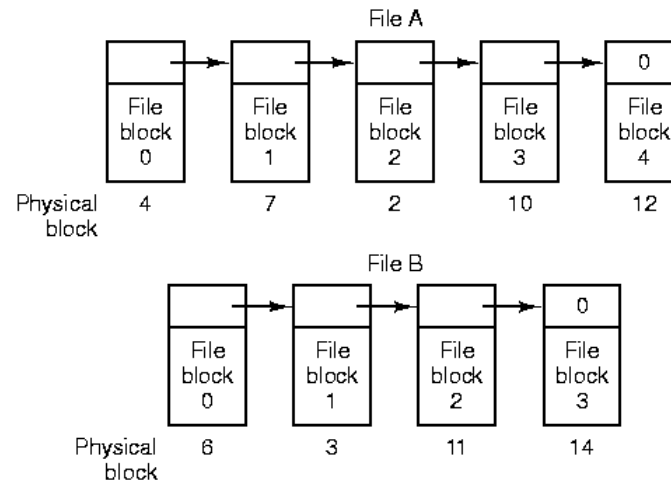
Implementación de ficheros

- Los datos de los ficheros se almacenan en memoria secundaria en los bloques de datos.
- Los bloques de datos de un fichero no tienen por qué ir seguidos ni estar en orden.
- Se necesita alguna manera de obtener dichos bloques de datos en orden.
- **Distintas implementaciones**

Implementación del sistema de ficheros

Implementación de ficheros

■ Asignación por lista enlazada



- Una parte del bloque de datos contiene el nº del siguiente
- **El BDF tiene el primer número de bloque**
- Acceso directo complejo e ineficiente ☹
- Se desperdicia espacio en el bloque de datos ☹

Implementación del sistema de ficheros

Implementación de ficheros

- Asignación por lista enlazada utilizando una tabla
 - Se sacan los nº de los bloques y se colocan en **una tabla única para todos los ficheros**
 - **BDF contiene el primer nº**
 - Todo el bloque para los datos 😊
 - Acceso directo sencillo 😊
 - Para buena eficiencia, la tabla debería estar completa en memoria principal 😊

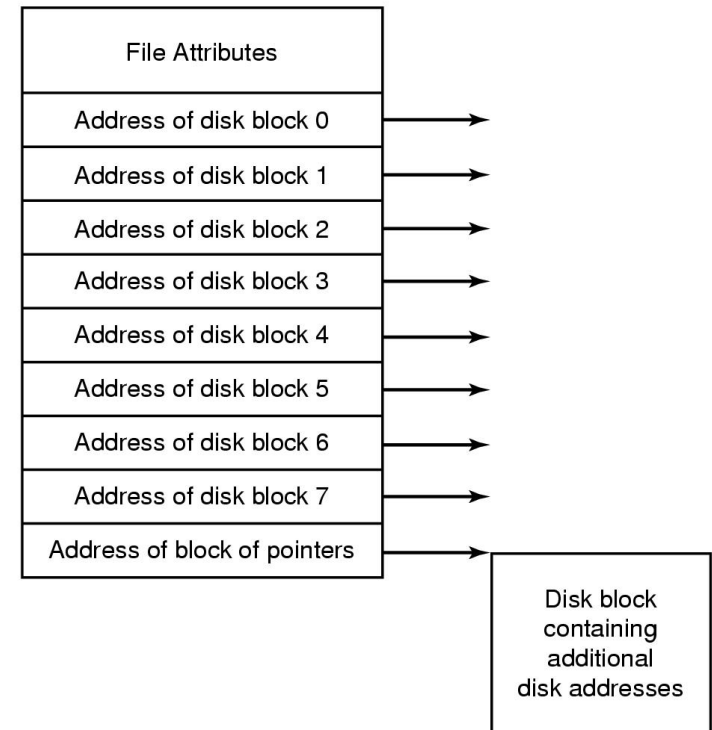
Physical block		
0		
1		
2	10	
3	11	
4	7	← File A starts here
5		
6	3	← File B starts here
7	2	
8		
9		
10	12	
11	14	
12	-1	
13		
14	-1	
15		← Unused block

Implementación del sistema de ficheros

Implementación de ficheros

■ Asignación indexada

- El BDF contiene un cierto n^o fijo de n^o de bloques
 - Uno o varios de ellos apuntan, realmente, a bloques que contienen más n^o de bloques del fichero
 - Esta indirección puede llegar a ser doble, triple, etc.
- Tamaño del fichero limitado por el índice ☹
- Tiempo de acceso a un bloque no igual para todos ☹

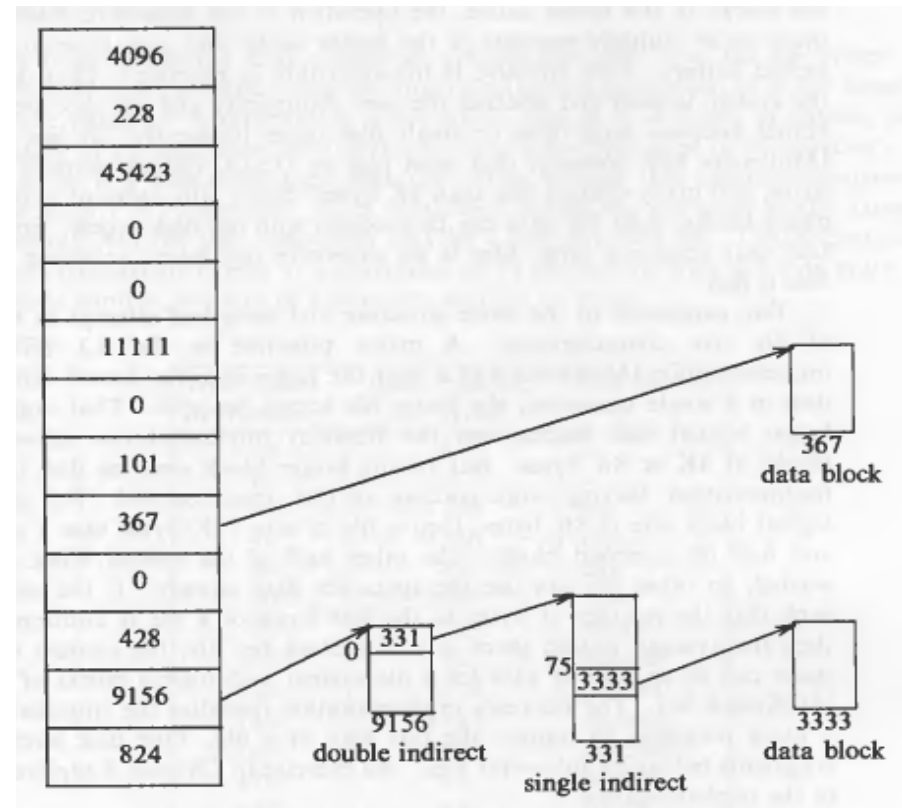


Implementación del sistema de ficheros

Implementación de ficheros

■ Asignación indexada (continuación)

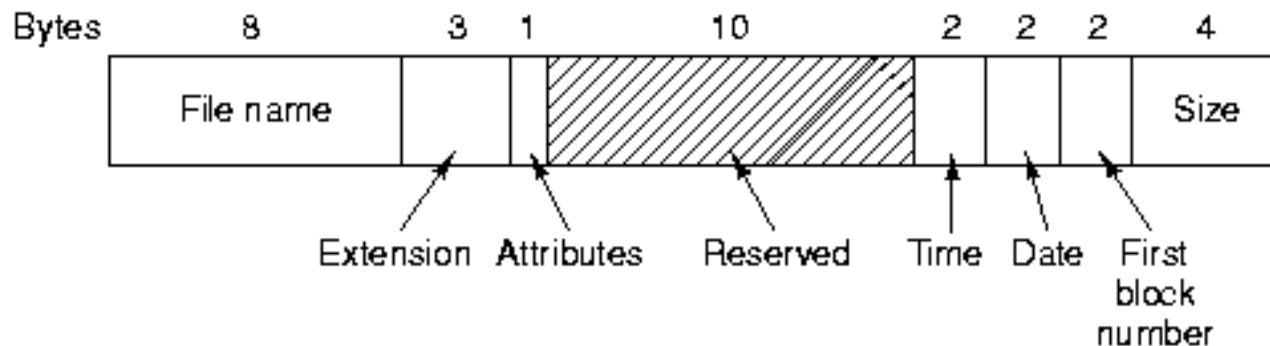
- Utilizada en UNIX
- En el ejemplo
 - 10 bloques directos
 - 1 de indirección simple
 - 1 de indirección doble
 - 1 de indirección triple



Implementación del sistema de ficheros

Implementación de directorios

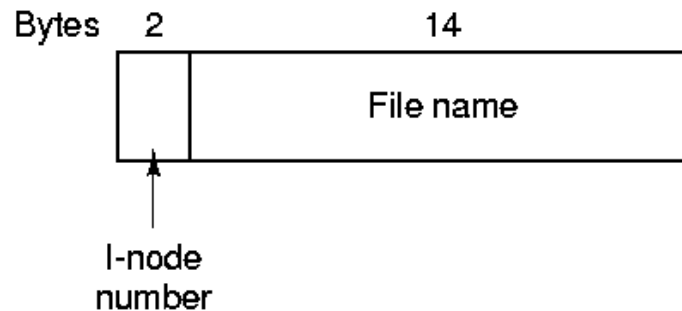
- Para que un fichero pueda ser manipulado
 - Se tiene que obtener su BDF a partir de su nombre
 - La relación entre BDF y nombres de fichero es mantenida por las entradas de los directorios
 - El BDF del directorio raíz en MP (siempre accesible)
- Implementación de directorios tipo MS-DOS
 - El contenido del BDF reside en la entrada del directorio



Implementación del sistema de ficheros

Implementación de directorios

- Implementación de directorios tipo UNIX
 - El BDF reside en una parte concreta del sistema de ficheros (lista de i-nodos en disco)
 - Un BDF se llama i-nodo
 - Se referencia por su número (identificador del fichero)





Implementación del sistema de ficheros

Implementación de directorios

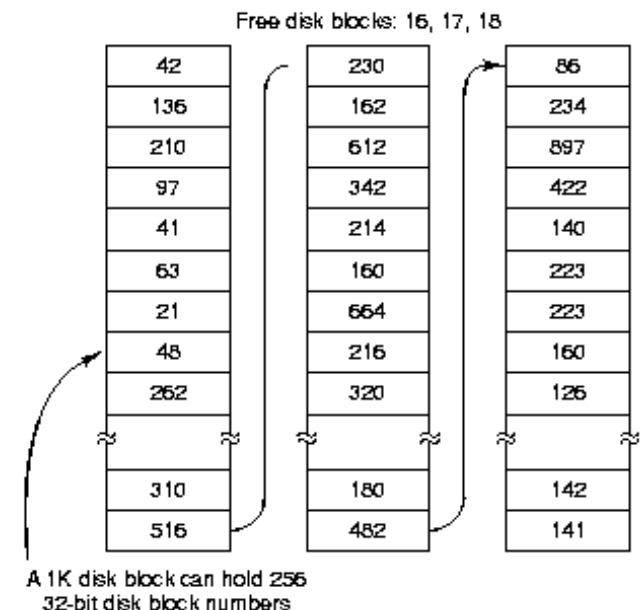
- Implementación de directorios tipo UNIX (*continuación*)
 - Ejemplo de contenido de directorio

Byte Offset in Directory	Inode Number (2 bytes)	File Names
0	83	.
16	2	..
32	1798	init
48	1276	fsck
64	85	clri
80	1268	motd
96	1799	mount
112	88	mknod
128	2114	passwd
144	1717	umount
160	1851	checklist
176	92	fsdb1b
192	84	config
208	1432	getty
224	0	crash
240	95	mkfs
256	188	inittab

Implementación del sistema de ficheros

Gestión del espacio libre

- Gestión de bloques libres
 - Es necesario conocer qué bloques están libres para asignar a ficheros nuevos o que crecen
 - Lista enlazada
 - Se usan los bloques libres para almacenar n° de bloques libres y se enlazan formando una lista (un n° se usa como puntero al siguiente elto. de la lista)
 - En memoria sólo necesario el primer elemento de la lista



Implementación del sistema de ficheros

Gestión del espacio libre

- Gestión de bloques libres
 - Mapa de bits
 - Partición de N bloques usa un mapa de N bits
 - 1=libre, 0=ocupado
 - Usa mucho espacio si hay pocos bloques libres
 - Fácil encontrar M bloques consecutivos
 - El mapa podría ser paginado

1001101101101100
0110110111101111
1010110110110110
0110110110111011
1110111011101111
1101101010001111
0000111011010111
1011101101101111
1100100011101111
⋮
0111011101110111
1101111101110111

A bit map



Implementación del sistema de ficheros

Consistencia del sistema de ficheros

- Muchas operaciones sobre ficheros y estructuras de datos que los soportan tienen lugar en memoria principal y se graban en el disco cuando el SO lo considera oportuno
 - Si el sistema “cae” antes de que toda la información haya sido escrita en disco, el sistema de ficheros puede haber quedado en un estado inconsistente
 - El problema es especialmente importante si la información no escrita se corresponde con BDF, directorios o la estructura de bloques libres
- Los SO disponen de una utilidad que comprueba la consistencia del sistema de ficheros
 - *fsck*, *scandisk*, etc.



Implementación del sistema de ficheros

Consistencia del sistema de ficheros

- Comprobaciones de consistencia (*fsck*)
 - De bloques
 - El programa construye dos tablas, cada una con un contador para cada bloque, inicialmente a 0
 - La primera tabla registra el nº de veces que un bloque aparece en un fichero
 - La segunda, ídem en la estructura de bloques libres
 - **Cada bloque un 1 en la 1º o en la 2º tabla → Sistema de ficheros consistente 😊**

BLOCK NUMBER															
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	0	1	0	1	1	1	1	0	0	1	1	1	0	0
Blocks in use															
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1
Free blocks															

Implementación del sistema de ficheros

Consistencia del sistema de ficheros

- Comprobaciones de consistencia (*fsck*) (*continuación*)
 - De bloques (*continuación*)
 - Algún bloque sin 1 en ambas tablas → Bloques perdidos ☹

BLOCK number

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

Blocks in use

0	0	0	0	1	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Free blocks

- Valor >1 en tabla bloques libres → Bloque repetido en estructura de libres ☹

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

Blocks in use

0	0	1	0	2	0	0	0	0	1	1	0	0	0	1	1
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Free blocks



Implementación del sistema de ficheros

Consistencia del sistema de ficheros

- Comprobaciones de consistencia (*fsck*) (*continuación*)
 - De bloques (*continuación*)
 - **Valor >1 en tabla bloques usados → Bloque repetido en varios ficheros ☹**
 - Se sacan tantas copias como sean necesarias del bloque, una para cada fichero

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	1	0	1	0	2	1	1	1	0	0	1	1	1	0	0	Blocks in use
0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	1	Free blocks



Implementación del sistema de ficheros

Consistencia del sistema de ficheros

- Comprobaciones de consistencia (*fsck*) (*continuación*)
 - De directorios
 - Usa la propiedad *nº de nombres* en el BDF
 - Crea un contador por cada BDF y lo inicializa a cero. Recorre la jerarquía de directorios incrementando el contador con cada entrada que referencia al mismo BDF
 - Al final, compara el contador con el número de nombres del fichero almacenado en su BDF
 - **Si son iguales, sistema de ficheros consistente** 😊
 - **Si el nº de nombres es mayor que el contador** ☹:
 - Reajustar nº de nombres en el BDF
 - **Si el nº de nombres es menor que el contador:** *un fichero podría llegar a eliminarse cuando todavía quedase algún nombre del mismo en algún directorio* ☹
 - Reajustar nº de nombres en el BDF