

Programación y Administración de Sistemas

6. Organización de sistemas de ficheros y discos

Pedro Antonio Gutiérrez

Asignatura “Programación y Administración de Sistemas”
2º Curso Grado en Ingeniería Informática
Escuela Politécnica Superior
(Universidad de Córdoba)
pagutierrez@uco.es

9 de abril de 2018



Objetivos del aprendizaje I

- Distinguir entre discos duros rígidos y discos duros SSD.
- Enumerar las partes de las que está compuesto un disco duro rígido.
- Explicar qué es el sistema de archivos de un sistema operativo y cómo funciona.
- Analizar la estructura del sistema de archivos: bloques de carga, bloques de datos, metainformación, superbloques, descriptores físicos, mapas de bits, listas de recursos libres...
- Dividir el servidor de archivos de un sistema operativo en los componentes que lo forman: sistema de archivos virtual, módulo de organización de archivos, servidor de bloques y manejadores de dispositivos, analizando la función de cada uno.

- Establecer la distintas opciones para implementar la asignación de bloques (con las distintas alternativas ofrecidas por los diferentes sistemas de archivos).
- Enumerar mecanismos para gestión del espacio libre.
- Enumerar y explicar mecanismos para el incremento de prestaciones mediante el uso de caché.

6.1. Introducción.

6.1.1. Discos rígidos y discos SSD.

6.1.2. Estructura de un disco rígido.

6.2. Estructura del sistema de archivos.

6.3. Servidor de archivos.

6.3.1. Sistema de archivos virtual.

6.3.2. Módulo de organización de archivos.

6.3.3. Servidor de bloques.

6.3.4. Manejador de dispositivos.

6.4. Asignación de bloques.

6.4.1. Lista enlazada.

6.4.2. Tabla de asignación de archivos (FAT).

6.4.3. Índices e índices multinivel.

6.5. Gestión del espacio libre.

6.6. Incremento de prestaciones.

6.6.1. Caché de nombres.

6.6.2. Caché de bloques.

- Cuestionarios objetivos.
- Pruebas de respuesta libre.
- Tareas de administración.

Introducción

- La función principal de un disco duro es almacenar la información del PC cuando no se encuentra conectado a la corriente eléctrica.
- También puede servir de extensión para la memoria RAM, gracias al mecanismo de **memoria virtual** (**intercambio**).
- En la actualidad, existen dos tecnologías que conviven en los discos duros: la de los **SSD** y la de los **discos rígidos**.
- Los discos rígidos funcionan de forma parecida a un tocadiscos, mientras que los discos SSD (*Solid State Disk* o, mejor, *Solid State Drive*) utilizan una memoria formada por semiconductores para almacenar la información (similar a *pendrives* o tarjetas de memoria).



Discos rígidos

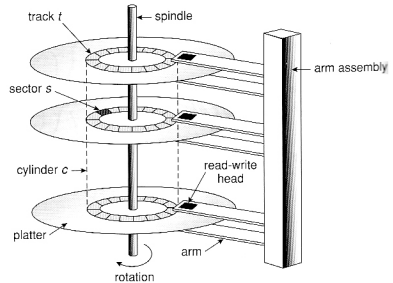
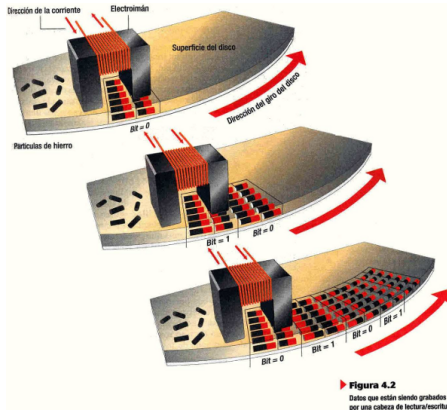


Figure 12.1 Moving-head disk mechanism.



Discos rígidos: escritura



Archivos

- **Archivo**: unidad de almacenamiento lógico no volátil que agrupa un conjunto de información relacionada entre si bajo un mismo nombre.
 - Un archivo debe poseer un nombre que permita acceder al mismo de forma unívoca.
 - Este nombre incluye una extensión (.txt, .zip...) que identifica el tipo de archivo.
 - El acceso a un archivo puede ser **secuencial** (para acceder a una posición hay que acceder antes a las anteriores) o **directo/aleatorio** (se puede acceder a cualquier posición).



Sistema de Archivos

Sistema de Archivos/Ficheros (SA/SF)

- Organiza la información de los dispositivos de almacenamiento secundario (disco duro, disco extraíble, DVDs, CDRom...).
- El dispositivo se divide manera lógica para que quede organizado de una forma inteligible para el SO.
- La división se hace a múltiples niveles:
 - Particiones o volúmenes.
 - Bloques.
 - Agrupaciones.

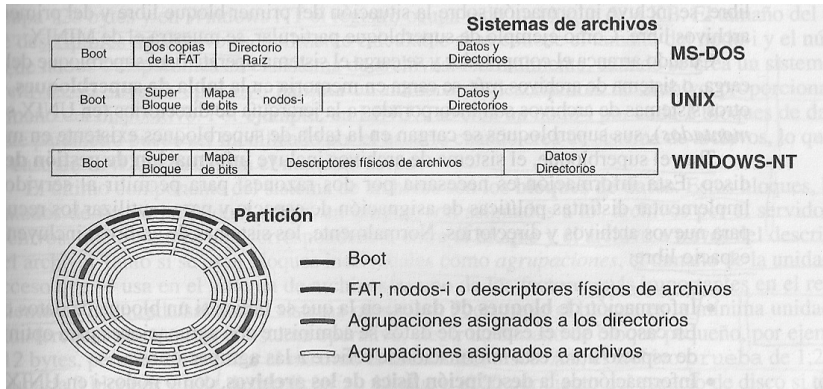


Sistema de Archivos

- **Partición:** porción de un disco a la que se le dota de una identidad propia y que se manipula como un entidad lógica independiente.
 - Las particiones deben **formatearse** para que se creen las estructuras necesarias que permiten al SO manipular el disco.
- **Bloque:** agrupación lógica de sectores físicos del disco, la cual supone la unidad de transferencia mínima que usa el SA.
 - El tamaño de bloque es un parámetro decisivo que afecta a la eficiencia del acceso a disco y a la fragmentación del mismo.
 - Tamaño de bloque **pequeño**: Mayor número de operaciones de Entrada/Salida (E/S) para acceder al archivo. Menor fragmentación.
 - Tamaño de bloque **grande**: Menor número de operaciones E/S para acceder al archivo. Mayor fragmentación.
- **Agrupación:** conjunto de bloques gestionado como una unidad lógica de almacenamiento.



Estructura del Sistema de Archivos



Estructura del Sistema de Archivos

- El bloque de carga (*boot* o *Volume Boot Record*) contiene código ejecutado al arrancar el ordenador por el iniciador ROM utilizando esa partición.
 - El MBR apunta al VBR de la partición activa¹.
 - Se suele incluir en todas las particiones (aunque no contengan el SO) para así mantener una estructura uniforme.
 - Se añade un *número mágico*, el cuál será comprobado por el iniciador ROM para demostrar que el bloque de carga es válido.
- *Metainformación*: super-bloques, FAT, nodos-i, mapas de bits, descriptores físicos...
 - Describe el SA y la distribución de sus componentes.
 - Es necesaria para poder acceder a los datos.

¹<https://whereismydata.wordpress.com/2008/08/23/file-systems-mbr-and-volume-boot-record-basic/>



Estructura del Sistema de Archivos

- **Superbloque:** características del SA, posición de los distintos elementos, tamaño...
 - Se mantiene una serie de información común para todos los SAs y una entrada característica para cada tipo de SA.
 - Al arrancar la máquina, los superbloques de todos los SAs que son cargados se mantienen en memoria.
- **Descriptores físicos de archivos:** nodos-i, registros de Windows-NT...
 - Describen cada uno de los archivos almacenados.
 - Tienen una estructura y tamaño muy dependiente del SO.
 - El número de descriptores debe ser proporcional al tamaño total del disco.
 - Incluyen: tamaño, apuntadores a los bloques del archivo, permisos, propietarios...



Estructura del Sistema de Archivos

- **Gestión del espacio libre**: distintos mecanismos permiten gestionar el espacio libre.
 - Se pueden utilizar **mapas de bits** o **listas de recursos libres**.
 - Gestión de dos tipos de recursos:
 - Mapas de **bloques**: indican qué bloques (o agrupaciones) están libres.
 - Mapas de **descriptores de archivos**: indican qué descriptores de archivos (nodos-i, registros...) están libres.
- **Bloques de datos**: es dónde se almacena realmente la información.



Servidor de Archivos

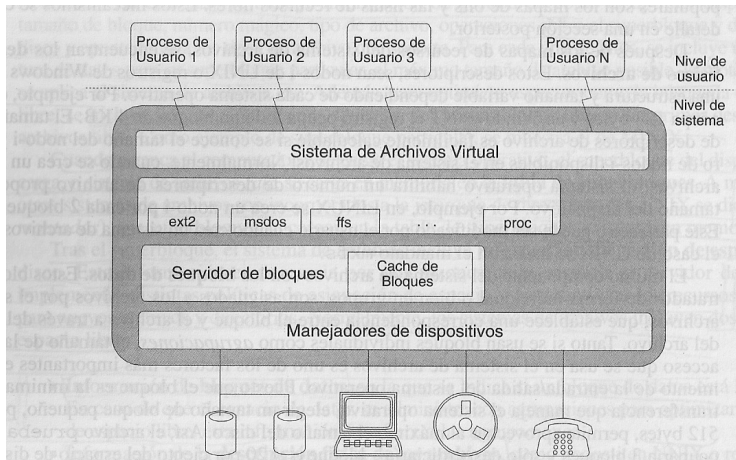
- **Servidor de Archivos:** es el componente del SO que se encargará de gestionar el acceso a archivos.
- Se sigue una filosofía de organización en capas.
 - Los niveles inferiores proporcionan servicios a los niveles superiores, y en cada nivel se aumenta la abstracción de las operaciones.

Capas del servidor de archivos

- 1 Sistema de archivos virtual (+ abstracto).
- 2 Módulo de organización de archivos.
- 3 Servidor de bloques.
- 4 Manejadores de dispositivos (- abstracto).



Servidor de Archivos



Servidor de Archivos

1 Sistema de archivos virtual:

- Proporciona la interfaz para las llamadas de E/S que deseen realizar los procesos de usuario, interactuando con el módulo de organización de archivos.
- Cumple las funciones de manejo de directorios, gestión de nombres, servicios de seguridad, integración de archivos de distintos dispositivos/particiones...
- Por ello, es necesario utilizar una estructura adicional (nodos virtuales o **nodos-v** en UNIX), que incluye las características comunes a todos los sistemas de archivos y un enlace al descriptor de archivo particular (nodo-i o registro).



Servidor de Archivos

1 Sistema de archivos virtual:

- Hay operaciones genéricas que se pueden realizar en cualquier SA (caché de nombres, gestión de nodos virtuales...).
- Otras operaciones deben ser implementadas independientemente para cada tipo de SA.
- Los **nodos virtuales** contienen la siguiente información:
 - Atributos del archivo.
 - Puntero al nodo-i real.
 - Punteros a funciones que realizan las operaciones genéricas de cualquier SA.
 - Punteros a funciones que realizan las operaciones propias del SA concreto.



Servidor de Archivos

② Módulo de organización de archivos:

- Se implementa por separado para cada tipo de SA.
- Relaciona la imagen lógica de un archivo con su imagen física, traduciendo direcciones lógicas (**contiguas**) del archivo a las direcciones físicas (normalmente **dispersas**) del dispositivo.
- Se prestan los servicios de gestión de espacio libre y manejo de descriptores de archivos físicos (no virtuales).
- Este nivel se basa en la información de los nodos-i y utiliza los servicios del servidor de bloques para realizar las operaciones correspondientes.



Servidor de Archivos

3 Servidor de bloques:

- Este nivel emite los mandatos genéricos para leer y escribir bloques en los manejadores de dispositivo (E/S de bloques).
- Se traducirán en llamadas al manejador específico del SA.
- En este nivel se realiza la caché de bloques.



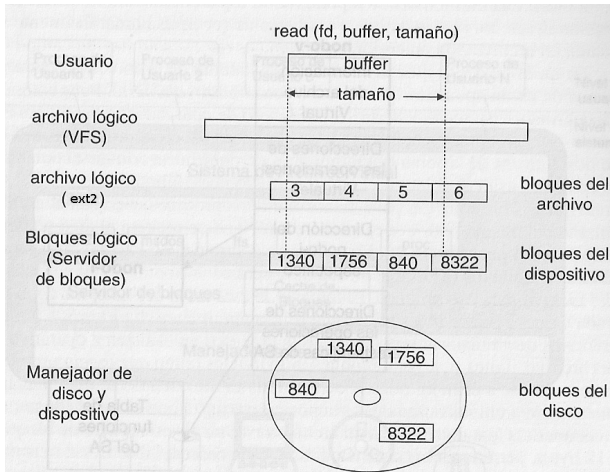
Servidor de Archivos

4 Manejadores de dispositivos:

- Son específicos para cada *hardware*.
- Traducen órdenes de E/S de alto nivel a un formato que pueda entender el dispositivo (dependiente del *hardware*).

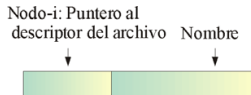


Servidor de Archivos

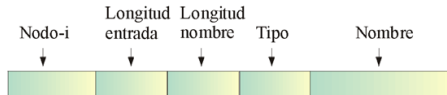


Directorios

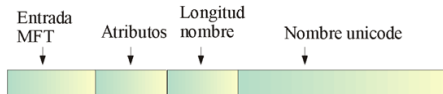
- Un directorio es un fichero con un formato determinado.
- El contenido de un directorio es una serie de entradas (**registros**), una por cada fichero contenido en él.
- Cada registro tiene, al menos, el nombre del fichero y el puntero al descriptor físico correspondiente.



Directorio de UNIX SV



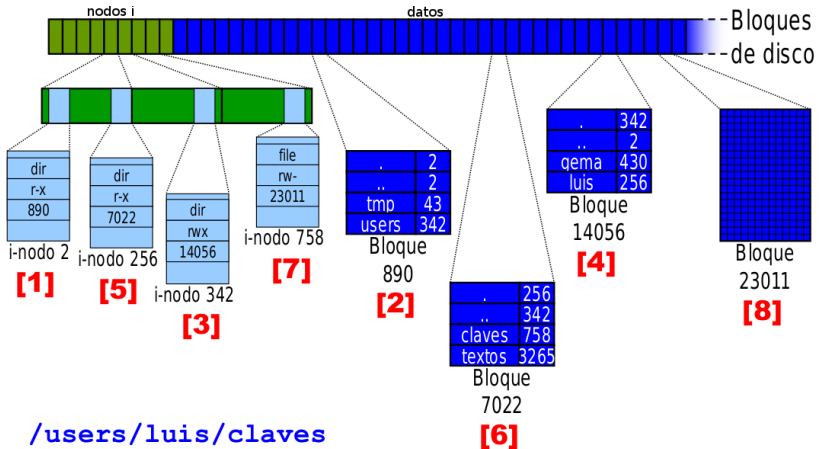
Directorio de LINUX Ext2



Directorio de Windows NTFS



Directorios



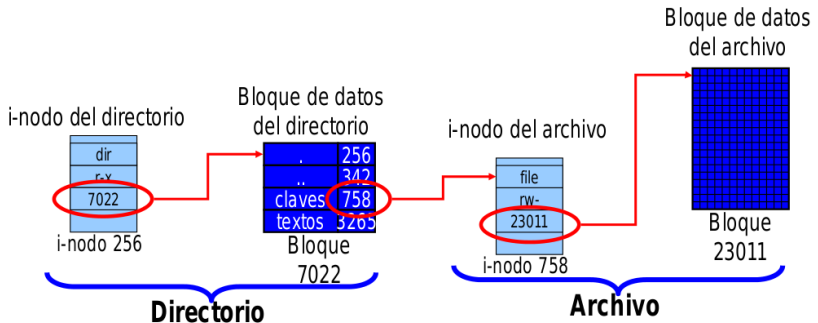
Directorios

- La ruta `/users/luís/claves` se interpreta de forma recursiva:
 - 1 Traer a memoria bloque del i-nodo 2 (i-nodo raíz, conocido).
 - 2 Se busca dentro `users` y se obtiene el i-nodo 342.
 - 3 Traer a memoria bloque del i-nodo 342.
 - 4 Se busca dentro `luís` y se obtiene el i-nodo 256.
 - 5 Traer a memoria bloque del i-nodo 256.
 - 6 Se busca dentro `claves` y se obtiene el i-nodo 758.
 - 7 Al leer el i-nodo 758, se detecta que es un fichero y ya se tienen dónde están los datos del archivo.
 - 8 Leer los bloques del fichero.
- ¿Cuándo parar?
 - No se tienen permisos.
 - Se ha encontrado el i-nodo del archivo.
 - No se encuentra el siguiente elemento de la ruta.

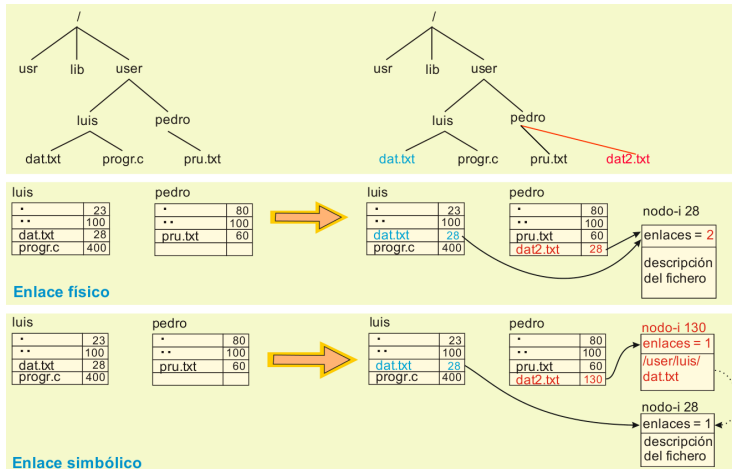


Directorios

- La llamada `open()` termina con la lectura del i-nodo.
- La verificación de permisos se hace con los datos del i-nodo.
- Un directorio no es un i-nodo:



Enlaces



Asignación de bloques

- **Asignación**: cómo se hace la correspondencia entre los bloques físicos del disco y los bloques lógicos del archivo.
- **Mecanismos de asignación**:
 - Asignación de **bloques contiguos**:
 - Todos los bloques del archivo se encuentran contiguos en el disco.
 - ☺ Muy sencillo de implementar.
 - ☺ Accesos secuencial y directo muy rápidos.
 - ☹ Necesario saber el tamaño del archivo al crearlo.
 - ☹ Fragmentación del disco.
 - ☹ Para añadir datos al archivo, puede que haya que moverlo.
 - Por todo ello, no se utiliza.



Asignación de bloques

- **Mecanismos de asignación:**
 - Asignación de **bloques no contiguos**:
 - Los bloques del archivo se encuentran en cualquier posición del disco.
 - ☺ Se produce menos **fragmentación** → el primer bloque asignado es el primero que hay libre.
 - ☹ Es necesario traducir el número de bloque lógico al número de bloque en el dispositivo.
 - Es la opción utilizada en la mayoría de SOs.
 - Para tener constancia de qué bloques no contiguos pertenecen a cada archivo, se utilizan **listas enlazadas** o **índices** (que pueden ser multinivel).
 - ISO9660: Inicio y tamaño (fichero contiguo).
 - SF MS-DOS: FAT (fichero enlazado).
 - SF UNIX: i-nodo (fichero indexado).
 - NTFS: Registro Windows (fichero indexado).



Asignación de bloques

Lista enlazada

- Cada bloque tiene un apuntador al siguiente bloque que seguiría en el archivo.
- El descriptor del archivo solo debe incluir la referencia al primer bloque.
 - ☺ El acceso secuencial es muy rápido.
 - ☹ El acceso aleatorio a un bloque concreto de un archivo es muy costoso.
 - ☹ Cada bloque incluye un apuntador que aumenta su tamaño (y complica el cálculo de espacio libre).
 - ☹ La pérdida de un bloque supone perder el archivo completo.



Asignación de bloques

Tabla de asignación de archivos

- Es una variación del [método lista enlazada](#).
- Los apuntadores se almacenan en una tabla independiente de los bloques (*File Allocation Table*, FAT).
- La tabla posee una entrada por cada bloque del SA.
- La FAT ocupará un espacio prefijado en la partición.
- Descriptor fichero → incluye su primera posición en la tabla.
- Acceso aleatorio al archivo: recorriendo la tabla.
- La tabla se aloja en caché para mejorar las prestaciones y se mantiene una copia doble en el disco para mayor fiabilidad.
- ☹ ¡La FAT puede llegar a ocupar mucho! → agrupaciones.



Asignación de bloques

FAT

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

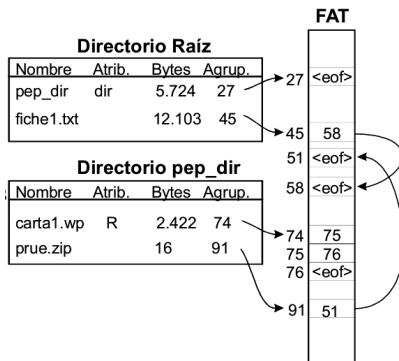
archivo B: 5 → 9 → 12

archivo C: 10 → 3 → 13



Asignación de bloques

- FAT de 12 bits: 4K agrupaciones.
- FAT de 16 bits: 64K agrupaciones.
- FAT de 32 bits: 2^{28} agrupaciones (solo usa 28 bits).
Tamaño de fichero en directorios \rightarrow 32 bits.
Tamaño máximo
 $\rightarrow 2^{32} - 1 = 4GB - 1$



Asignación de bloques

Índices

- Los punteros a los bloques están juntos y contiguos en una localización concreta → **Bloques índice**.
- Cada archivo tiene un bloque índice.
- Para buscar el *i-ésimo* bloque de un fichero, buscamos la *i-ésima* entrada en su bloque índice
 - ☺ Buen acceso directo.
 - ☺ Se evita la fragmentación.
 - ☹ ¿Tamaño del bloque índice? → debe fijarse un número de entradas y hay que reservar espacio para todas ellas.
 - ☹ Limitamos el tamaño máximo de los archivos.



Asignación de bloques

Índice multinivel

- Consiste en introducir n niveles de apuntadores, de manera que los apuntadores del descriptor apuntan a otros.
- Índice multinivel de nivel 1: el bloque índice apunta a otros bloques índices que finalmente apunta a un bloque de datos del fichero.
 - ☺ Evita tener que prefijar el tamaño del bloque índice (podemos poner apuntadores a NULL).
 - ☺ El bloque índice tendrá un número pequeño de entradas.
 - ☹ Cada nivel, supone un acceso a disco adicional.
 - ☹ Para archivos pequeños, se desaprovechan muchos bloques índice.



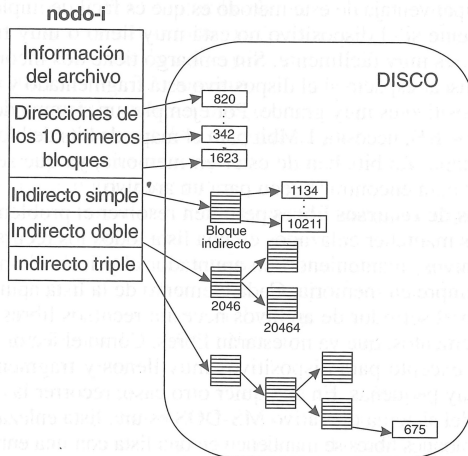
Asignación de bloques

Solución UNIX → Esquema híbrido

- Por cada nodo-*i* incluir:
 - Punteros directos a los 10 primeros bloques (para archivos pequeños).
 - Puntero a un bloque índice de primer nivel (donde encontraremos punteros a bloques).
 - Puntero a un bloque índice de segundo nivel (donde encontraremos punteros a punteros a bloques).
 - etc.
- Bloques del disco: bloques de datos o bloques índice.
- En ext4 y en NTFS existen los extents (bloques índice especiales que marcan una zona contigua del disco “numeroBloqueInicial, numeroBloques”).



Asignación de bloques: indexado multinivel



Gestión del espacio libre

- **Gestión del espacio libre**: se necesita para asignar espacio a los archivos nuevos o a los que se les desea añadir datos.
- Se mantienen **mapas de recursos**, implementados como mapas de bits o listas de recursos libres.
 - **Mapas de bits**
 - Se incluye un bit por recurso (descriptor de archivo, bloque o agrupación), que será 1 si el recurso está libre y 0 en caso contrario.
 - ☺ Muy sencillo de implementar y de usar.
 - ☺ Disco poco fragmentado → bloques libres al final, búsqueda muy eficiente.
 - ☹ Disco fragmentado → búsqueda más costosa.
 - ☹ Espacio adicional requerido por el mapa.
 - FAT: la propia tabla actúa como mapa de recursos.



Gestión del espacio libre

- Gestión del espacio libre.
 - Listas de recursos libres
 - Mantener una lista de apuntadores a los recursos libres.
 - Al ocupar un recurso lo borramos de la lista.
 - ☺ Muy eficiente para discos muy llenos y fragmentados.
 - ☹ Disco con mucho espacio libre → Ineficiente debido a que hay que cargar la lista.
 - Solución → Incluir número de bloques libres consecutivos en la lista.



Gestión del espacio libre: FAT + lista enlazada

x	x	EOF	13	2	9	8	FREE	4	12	3	FREE	EOF	EOF	FREE	BAD	FREE	FREE	FREE	
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	

Bloques libres → 7 → 11 → 14 → 16 → 17 → 18 (A)

Bloques libres → 7,1 → 11,1 → 14,1 → 16,3 (B)



Incremento de prestaciones

- Acceso a **memoria** → orden de nanosegundos.
- Acceso a **disco** → orden de milisegundos.
 - **Almacenamiento intermedio de los datos**
 - Mantener una caché de datos en Memoria Principal (MP).
 - Aprovecha la proximidad **espacial** y **temporal** en las referencias a los datos accedidos.
 - **Caché de nombres**: lista con {nombre,nodo-i}.
Si se vuelve a acceder al archivo, no hay que hacer toda la búsqueda del nodo-i.
 - **Caché de bloques**: colección de bloques leídos o escritos recientemente.
Si se vuelve a acceder a ese bloque, no hay que cargarlo de nuevo.



Incremento de prestaciones

- **Caché de bloques:**
 - Si el bloque está en MP, se escribirá o leerá en MP.
 - Posteriormente, se moverán los bloques de MP al dispositivo.
 - Si la caché está llena, hay que eliminar algún bloque:
 - **Políticas de reemplazo:** *First In First Out (FIFO)*, *Most Recently Used (MRU)*, *Least Recently Used (LRU)*...
 - Lo más común es **LRU**: aprovecha que los bloques no utilizados durante mucho tiempo, posiblemente no volverán a ser utilizados.
- ☹ Peligroso si hay un fallo del SA.



Incremento de prestaciones

- **Caché de bloques:**

- Bloques sucios (cambiados en caché pero no en el disco).
Distintas políticas a la hora de mantener la **coherencia**:
 - **Escritura inmediata** (*write-through*) → Siempre actualizado.
 - **Escritura diferida** (*write-back*) → Actualizamos cuando el bloque salga de la caché.
 - **Escritura periódica** (*delayed-write*) → Establecer un tiempo periódico para las actualizaciones.
Compromiso entre rendimiento y fiabilidad.
Reduce la extensión de los posibles daños por caídas.
- Se puede distinguir entre bloques **especiales** (directorios, nodos-i o bloques índice) y bloques de **datos**.
Bloques especiales → *write-through*.
- No se debe quitar un disco del sistema sin antes volcar los datos de la cache (comando **sync**).



Referencias



Fernando Pérez-Costoya, Jesús Carretero-Pérez y Félix García-Carballeira.

Problemas de Sistemas Operativos. De la base al diseño.
Tema 8. Archivos y directorios. Sección 8.4. Sistemas de Archivos.

Mc Graw Hill, Segunda Edición, 2003.



Evi Nemeth, Garth Snyder, Trent R. Hein y Ben Whaley

Unix and Linux system administration handbook.

Capítulo 8. *Storage and disk*.

Prentice Hall. Cuarta edición. 2010.



Programación y Administración de Sistemas

6. Organización de sistemas de ficheros y discos

Pedro Antonio Gutiérrez

Asignatura “Programación y Administración de Sistemas”
2º Curso Grado en Ingeniería Informática
Escuela Politécnica Superior
(Universidad de Córdoba)
pagutierrez@uco.es

9 de abril de 2018

