

Sistemas Operativos

Ayudantía 7: File Systems

Profesores: Martín Gutiérrez, Víctor Reyes

Ayudantes: Diego Banda, Dante Hortuvia



Contacto



dante.hortuvia@mail_udp.cl



doshuertos



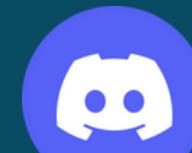
github.com/Doshuertos/Ayudantias_SO_CIT2010

Sección 1

Sección 2



diego.banda@mail_udp.cl



darklouds



github.com/DiegoBan/SO2025-I

Y dele... (Tarea 2)

- Recordar fecha de entrega: 1 de junio (**QUEDAN 4 DIAS!!**)



¿Preguntas/Dudas?

¿Cómo van?

File System

¿Qué es un File System?

Un sistema de archivos es el componente del sistema operativo encargado de gestionar cómo se almacenan, organizan y acceden los datos en un dispositivo de almacenamiento (como discos duros, SSD, USB, etc.). Está compuesto por dos partes principales:

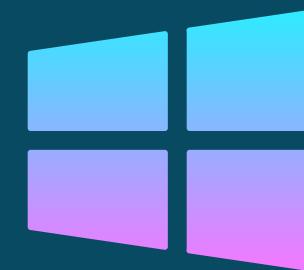
1. Colección de archivos: cada archivo contiene datos relacionados, como documentos, imágenes, programas, entre otros.
2. Estructura de directorios: organiza los archivos en una jerarquía (carpetas y subcarpetas), facilitando su búsqueda, acceso y administración.

Además, el sistema de archivos se encarga de llevar el control de atributos como el nombre del archivo, permisos de acceso, tamaño y fechas de creación/modificación. Algunos ejemplos de sistemas de archivos comunes son FAT32, NTFS, ext4 y APFS.

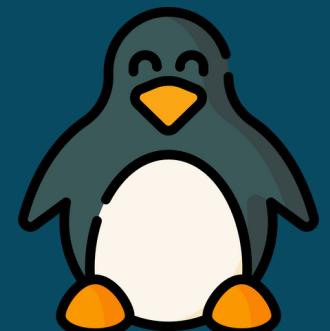
Tipos de archivos

Como se mencionó previamente, existen varios tipos de sistemas de archivos. **FAT** y **FAT32** son comúnmente utilizados en dispositivos de almacenamiento extraíbles, como unidades USB y tarjetas SD, debido a su amplia compatibilidad entre distintos sistemas operativos. **exFAT** (Extended File Allocation Table) es una evolución de **FAT32** que mejora el soporte para archivos de gran tamaño y elimina algunas de sus limitaciones.

NTFS es el sistema de archivos que utiliza Windows



ext3 y **ext4** son los sistemas de archivos se utilizan en Linux, **ext4** es una mejora de **ext3**



APFS (Apple File System), es el sistema de archivos que se usa en Apple, antiguamente se utilizaba **HFS+**



Estructura de los Archivos

En un sistema de archivos (File System), la unidad mínima de almacenamiento es el archivo. Un archivo es una secuencia de bytes que puede representar distintos tipos de información, como texto, imágenes, videos, programas, entre otros.

Estos archivos se almacenan en dispositivos de memoria secundaria (HDD, SSD).

Para organizar los archivos, el sistema operativo utiliza una estructura jerárquica de directorios, similar a un árbol. En esta estructura, cada directorio puede contener archivos y otros subdirectorios, lo que permite una organización lógica y ordenada de la información.

Un directorio es, en esencia, un tipo especial de archivo que contiene referencias (también llamadas entradas) a otros archivos o directorios. Esta jerarquía facilita la navegación, el acceso y la gestión eficiente de los datos dentro del sistema.



Syscalls en File Systems

Para controlar los archivos en un File System, se utilizan syscalls, algunos son:

- **open**: Abre un archivo para su lectura, escritura o ambas. Si el archivo no existe, puede crearse.
- **read/write**: read: extrae datos del archivo al programa/write: envía datos desde el programa al archivo.
- **fsync**: Fuerza a que los datos modificados en un archivo se escriban físicamente en el disco.
- **mv**: No es una syscall directamente, sino un comando de usuario en sistemas tipo Unix que utiliza syscalls, Sirve para mover o renombrar archivos o directorios.
- **stat**: Obtiene información sobre un archivo, como su tamaño, permisos, fecha de modificación, tipo, etc.
- **rm**: es un comando, no una syscall directa, sirve para eliminar archivos o directorios
- **touch**: También es un comando de usuario. Su función principal es crear archivos vacíos o actualizar la fecha de acceso/modificación de un archivo existente.



Syscalls en File Systems

Los directorios no se manejan exactamente igual que los archivos regulares. Por eso, existen llamadas al sistema específicas para trabajar con ellos, lo que permite tener mayor control, organización y seguridad en la estructura jerárquica del sistema de archivos:

`mkdir`: Crea un nuevo directorio.

`opendir,readdir,closedir` : Se utilizan para abrir, leer y cerrar directorios respectivamente.

`rmdir` : Elimina un directorio vacío.



ls -l

El comando ls -l en sistemas Unix/Linux se usa para listar los archivos y directorios en el directorio actual (o uno especificado) en formato detallado.



Como funcionan los permisos en Unix/Linux

En Unix/Linux hay 3 tipos de permisos que controlan el acceso a archivos y directorios:

- r : significa **read**, esta permite **leer** el contenido.
- w : significa **write**, esta permite **modificar o borrar**.
- x : significa **execute**, permite **ejecutar** el archivo o **ingresar** a un directorio.

Si un permiso no está presente, se representa con -

Chmod (Change mode)

El comando chmod (change mode) permite cambiar los permisos usando una notación octal donde:

- read = 4
- write = 2
- execute = 1

Ejemplo a continuacion

Ejemplo: Chmod (Change mode)

```
-rw-r--r--. 1 dhortuvia dhortuvia 193 may 16 22:27 Importante_ejecucion_Docker.txt
```

```
chmod 777 Importante_ejecucion_Docker.txt
```

Donde tenemos que user = 4 + 2 + 1, group = 4 + 2 + 1 y other = 4 + 2 + 1, esto le indicara al SO que todos tienen permisos para leer, escribir y ejecutar.

```
-rwxrwxrwx. 1 dhortuvia dhortuvia 193 may 16 22:27 Importante_ejecucion_Docker.txt
```



chown

El comando chown (change owner) permite cambiar el propietario (usuario) y/o el grupo de un archivo o directorio.

- Si solo se le pasa un **nuevo_usuario**, solo cambia el dueño del archivo y se conserva el grupo quedando [nuevo_usuario]:[grupo]
- Si solo se le pasa un **nuevo_usuario:**, cambia al dueño y el grupo al nuevo usuario quedando [nuevo_usuario]:[nuevo_usuario]
- Si se le pasa un **nuevo_usuario:nuevo_grupo**, cambia al dueño y el grupo al nuevo usuario y al nuevo grupo quedando [nuevo_usuario]:[nuevo_grupo]
- Si se le pasa un **:nuevo_grupo**, se conserva al dueño y el grupo es cambiado quedando [usuario]:[nuevo_grupo]

**Su sintaxis es : chown [usuario]:
[grupo] nombre_del_archivo**

```
-rw-r--r--. 1 dhortuvia dhortuvia 193 may 16 22:27  
Importante_ejecucion_Docker.txt
```

sudo chown dbanda Importante_ejecucion_Docker.txt

```
-rw-r--r--. 1 dbanda dhortuvia 193 may 16 22:27  
Importante_ejecucion_Docker.txt
```

Como se organiza un File system

Así como los procesos tienen un PID, los archivos tienen un i-nodo, un identificador único que almacena metadatos como:

- Tamaño del archivo
- Propietario
- Permisos
- Fechas de creación/modificación
- Punteros a los bloques de datos

Para visualizar el id del i-nodo en Linux utilizar ls -li y deberia aprecer algo asi :

```
189752 drwxr-xr-x. 1 dhortuvia dhortuvia 118 may 14 17:46 Ayudantia_so
```

Como se organiza un File system

El sistema de archivos se organiza como un árbol, con la raíz / como punto de partida.

Ruta Absoluta: parte desde la raiz, es la ubicacion exacta con respecto a la raiz , NO se puede repetir

Ej : home/user/escritorio/ayudantia_so/ayudantia_2/pauta.pdf

Ruta relativa: parte desde el directorio actual, su ubicacion del archivo con respecto a otro, Si se puede repetir

Ej : ../ayudantia_5/Codigo_ejemplo.c

Links

Los links (enlaces) son referencias que permiten acceder a archivos o directorios desde diferentes ubicaciones dentro del sistema de archivos, sin duplicar su contenido.

Existen dos tipos principales de enlaces:

1. Soft Link (Symbolic Link)

- Es una referencia al nombre del archivo o directorio.
- **No** está vinculado al i-nodo, solo al path.
- Similar a un acceso directo.
- Si se elimina el archivo original, el link queda roto (dead link).
- Puede apuntar a archivos o directorios, incluso en otro sistema de archivos.

2. Hard Link

- Es una referencia directa al i-nodo del archivo.
- Es como una "copia" del archivo, pero sin duplicar datos.
- Ambos nombres apuntan al mismo contenido.
- El archivo no se elimina hasta que todos los hard links desaparezcan (usa un contador de referencias).
- **No** se pueden hacer hard links a directorios ni entre distintos sistemas de archivos.
- **No** se puede crear hard links a carpetas, pipes, enlaces simbólicos, Sockets y dispositivos en /dev

Links

Tengo un archivo de Solemne_1.ppt y quiero crearle un link

Para crear un Soft Link se debe utilizar :

```
ln -s Solemne_1.ppt ruta_solemne_1.ppt
```

Como se visualizaría en la consola:

```
lrwxrwxrwx 1 dhortuvia dhortuvia may 27 20:27 link_Solemne.ppt -> Solemne_1.ppt
```

Para crear un Hard Link se debe utilizar :

```
ln Solemne_1.ppt Clon_solemne_1.ppt
```

Como se visualizaría en la consola:

```
423809 -rw-r--r-- 2 dhortuvia dhortuvia 4096 may 27 Solemne_1.ppt
```

```
423809 -rw-r--r-- 2 dhortuvia dhortuvia 4096 may 27 Solemne_copia.ppt
```

Links

- El soft link se identifica con una "l" (letra ele) al principio en el listado de ls -l, mientras que el hard link aparece con un "-", como cualquier archivo normal.
- Tanto el hard link como el archivo original tienen un contador de enlaces (link count). Si ves 2, significa que existen dos referencias (nombres) al mismo contenido. Si se creara un tercer hard link, el contador subiría a 3.
- El soft link indica explícitamente hacia dónde apunta (aparece con → destino).
- Si se elimina el archivo original (Solemne_1.ppt), el soft link (link_Solemne.ppt) se convierte en un dead link (enlace roto).
- Un soft link es simplemente un archivo especial que contiene una ruta, mientras que un hard link comparte directamente el contenido con el archivo original.
- Aunque en la terminal ambos se vean como archivos normales, el hard link no es una "copia", sino otra entrada al mismo i-nodo.

Implementación de un File system

- Master Boot Record (MBR):

Es el primer sector del disco (conocido como Sector 0), y se carga al iniciar el PC. Contiene el Boot Loader, que se encarga de iniciar el sistema operativo.

- Partition Table:

Está ubicada dentro del MBR y contiene información sobre las particiones del disco: su ubicación, tamaño, tipo de sistema de archivos, etc.

- Boot Control Block:

Presente en cada partición, permite cargar el sistema operativo desde esa partición específica. Suele ser parte del primer bloque de la partición.

- Volume Control Block:

Almacena información sobre el formato del sistema de archivos de la partición: tipo de FS (ext4, NTFS, etc.), número total de bloques, tamaño de bloque, entre otros datos clave.

- File Control Block (FCB):

Contiene los metadatos de cada archivo, como el nombre, tamaño, permisos, ubicación en disco, fechas de acceso/modificación, etc.

- Root Directory:

Es el directorio raíz de la partición, donde comienza la estructura jerárquica del sistema de archivos.

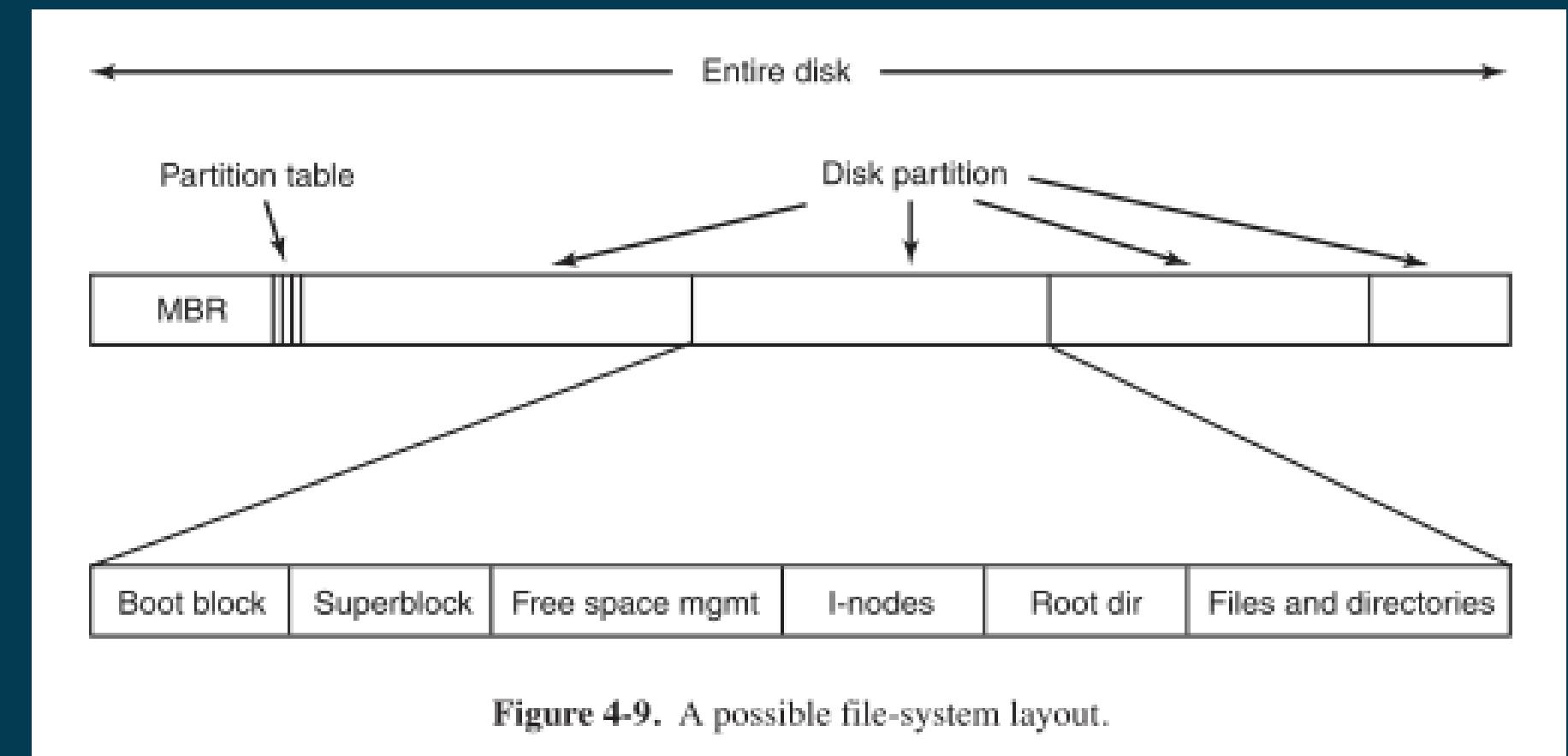


Figure 4-9. A possible file-system layout.

VFS (Virtual File System)

- Es una capa de abstracción que permite al sistema operativo interactuar con distintos tipos de sistemas de archivos de forma uniforme.
- Gracias a VFS, cualquier sistema de archivos que lo implemente (como ext4, FAT32, NTFS, etc.) puede integrarse al sistema operativo sin necesidad de cambiar la forma en que los programas acceden a los archivos.
- Proporciona una interfaz común y unificada, facilitando la interoperabilidad entre sistemas de archivos heterogéneos.
- VFS utiliza estructuras llamadas v-nodes (virtual nodes), que representan archivos o directorios de forma abstracta.
- Cada v-node apunta a funciones específicas del sistema de archivos real, que implementan operaciones como open, read, write, etc.
- Importante: Los v-nodes no son equivalentes a los i-nodos. Mientras que los i-nodos contienen información física de archivos en sistemas como ext4, los v-nodes son estructuras virtuales gestionadas por VFS para abstraer esa información.

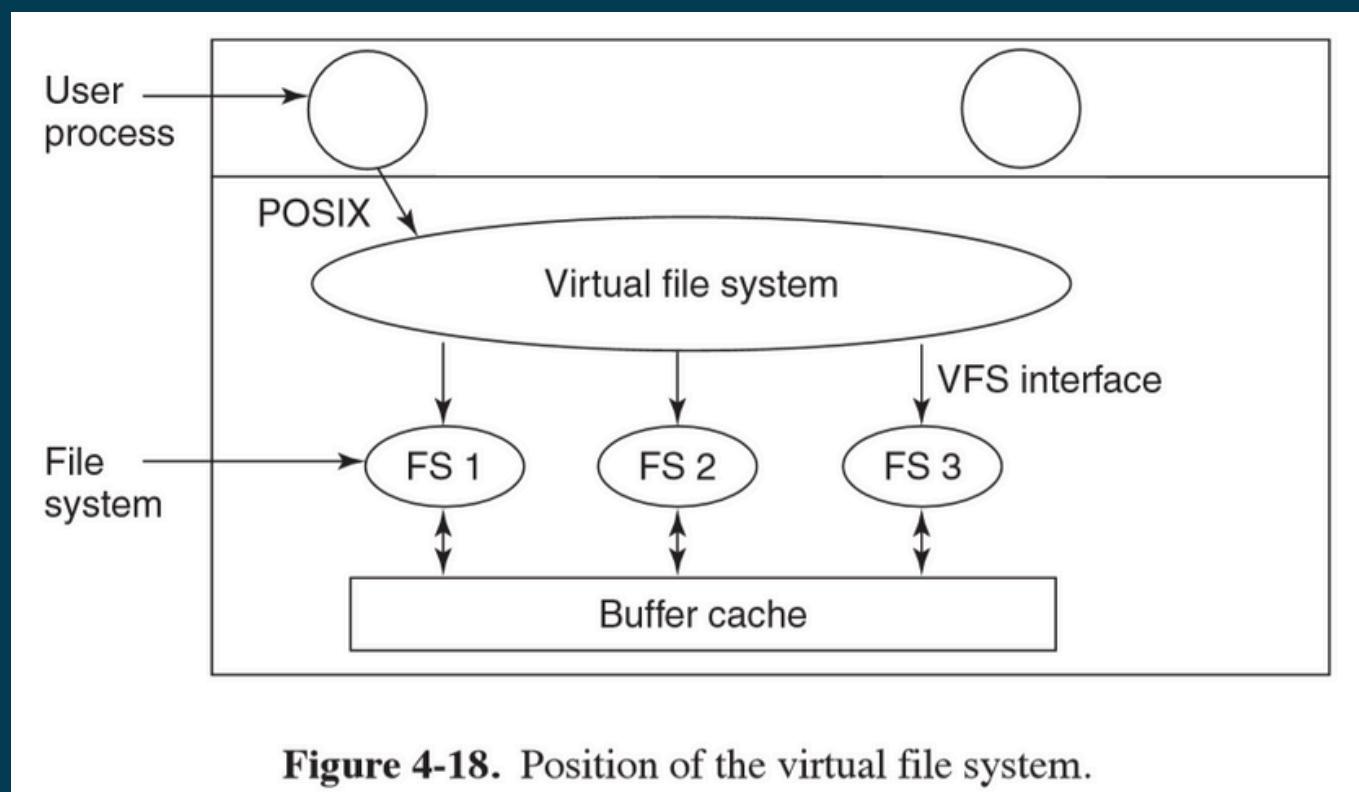


Figure 4-18. Position of the virtual file system.

Lectura de un Archivo con VFS

1. Inicio desde el proceso:

Una aplicación solicita la lectura de un archivo mediante una syscall (por ejemplo, read()).

2. Uso del File Descriptor:

El sistema busca en la tabla de procesos el file descriptor, que identifica el archivo abierto por ese proceso.

3. Acceso al v-node:

El file descriptor apunta a un v-node en el VFS, que representa el archivo de forma abstracta.

Este v-node contiene:

- Información del archivo.
- Punteros a funciones del sistema de archivos real (read, write, etc.).

4. Invocación de syscall específica:

El VFS llama a la función correspondiente del sistema de archivos real (por ejemplo, read()).

5. Lectura desde el FS real:

La función read() del sistema de archivos accede al dispositivo de almacenamiento y retorna los datos al proceso.

Ejemplo:

Leer un archivo desde un pendrive con FAT32 montado en un sistema Mac.

Gracias a VFS, se accede al archivo de forma transparente, sin importar el tipo de FS.

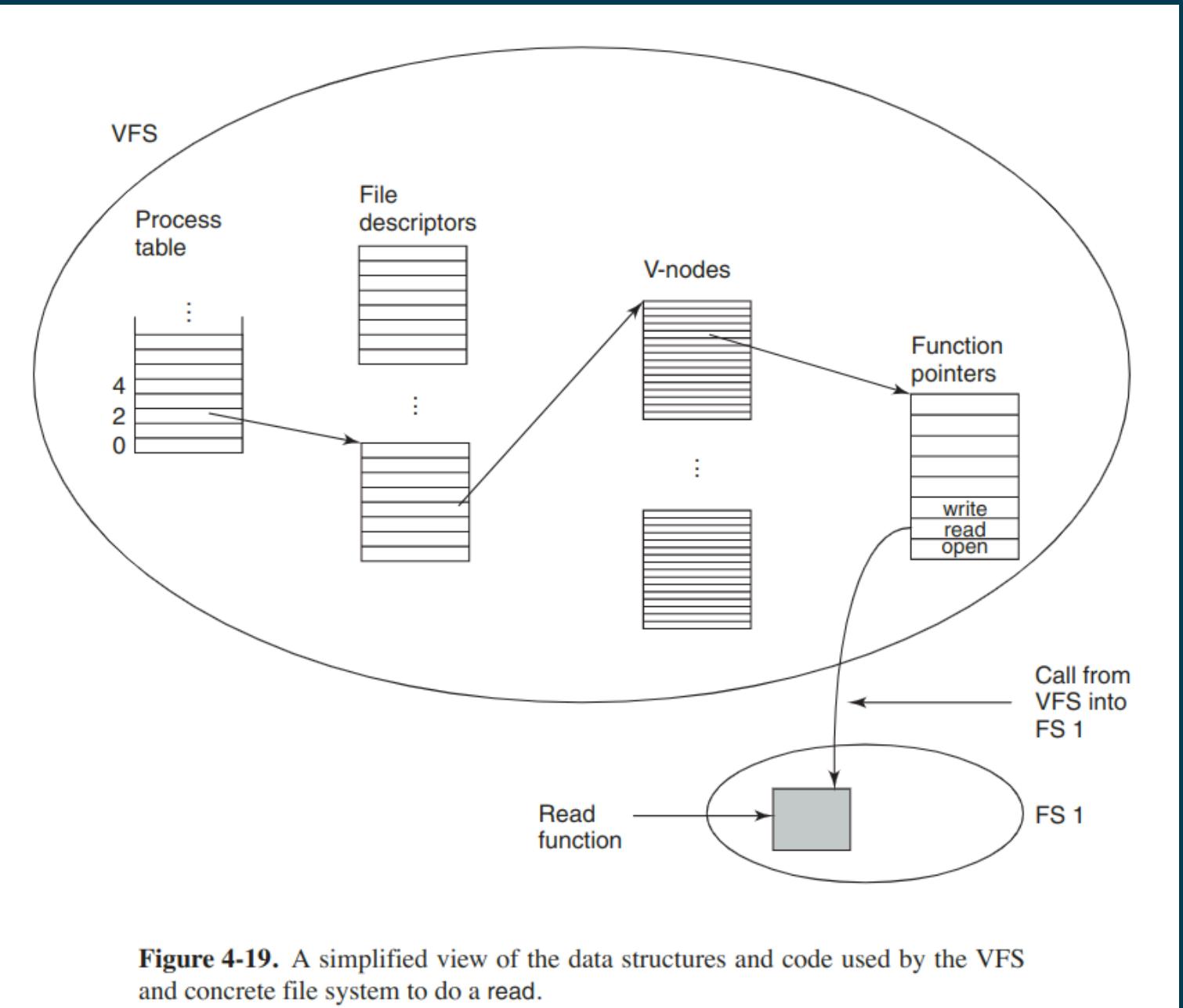


Figure 4-19. A simplified view of the data structures and code used by the VFS and concrete file system to do a read.

File Systems en Memoria Principal

- Un File System en memoria se monta directamente sobre la RAM, en lugar de un disco físico.
- Se utiliza cuando se requiere velocidad extrema de acceso, ya que la RAM es mucho más rápida que cualquier tipo de almacenamiento permanente.
- Ejemplos comunes:
 - /tmp en muchas distribuciones de Linux (temporal).
 - Sistemas embebidos o live-CDs que cargan todo el FS en memoria.
- Características:
 - Extremadamente rápido.
 - Los datos no persisten tras un reinicio (son volátiles).
 - Ideal para archivos temporales, cachés o testing.
- Tipos de FS en memoria:
 - tmpfs: muy usado en Linux; puede intercambiar a disco si falta RAM.
 - ramfs: siempre permanece en RAM, sin swap.

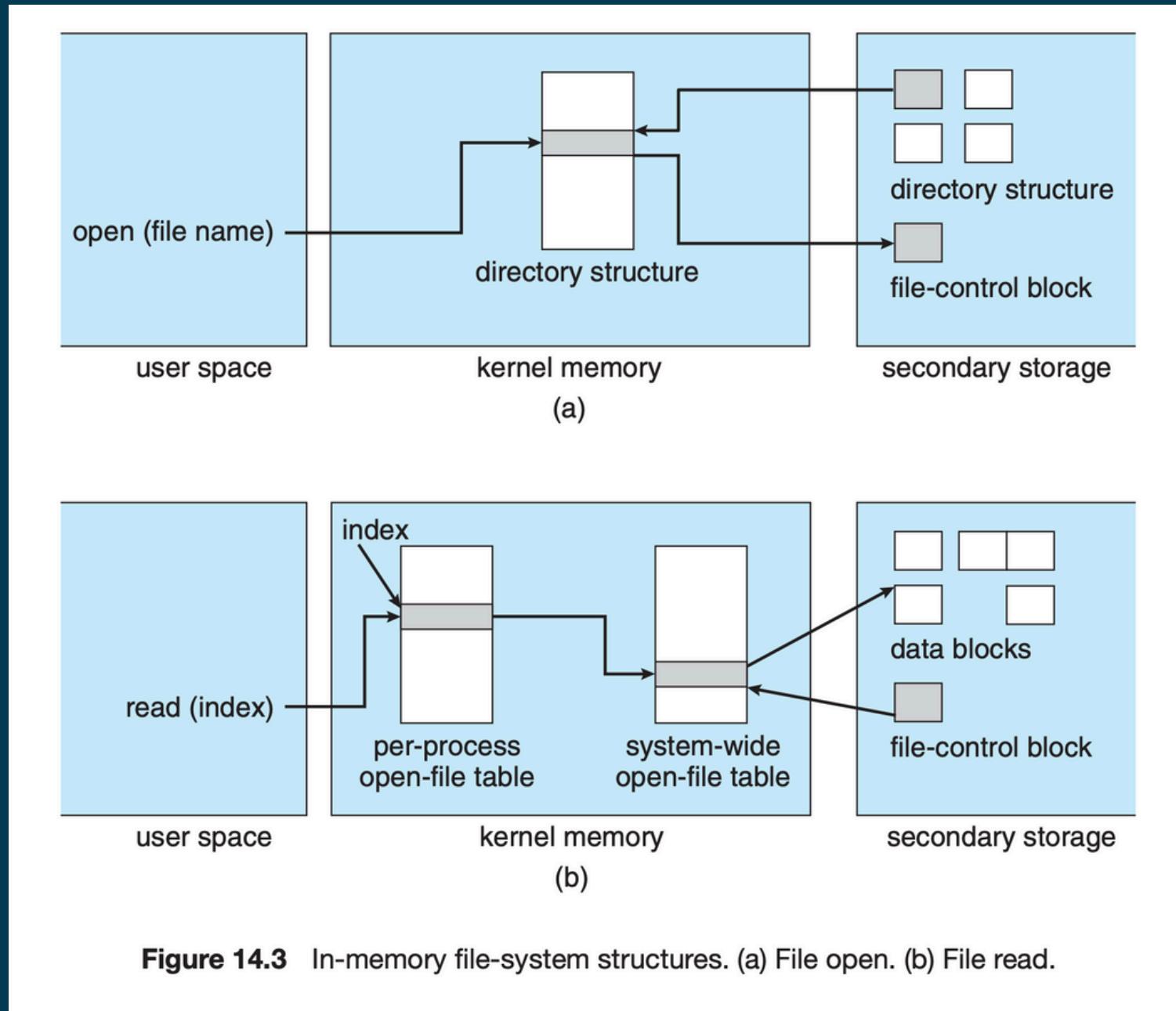


Figure 14.3 In-memory file-system structures. (a) File open. (b) File read.

Fragmentacion

Al igual que en la memoria, los sistemas de archivos también enfrentan problemas de fragmentación, lo que puede afectar el rendimiento.

Fragmentación Interna:

- Ocurre cuando se asigna un bloque de tamaño fijo a un archivo, pero este no ocupa todo el espacio disponible en dicho bloque.

Resultado: Espacio desaprovechado dentro del bloque.

Fragmentación Externa:

- Sucede cuando hay suficiente espacio libre en el disco en total, pero está disperso en bloques no contiguos, impidiendo guardar archivos grandes de forma continua.

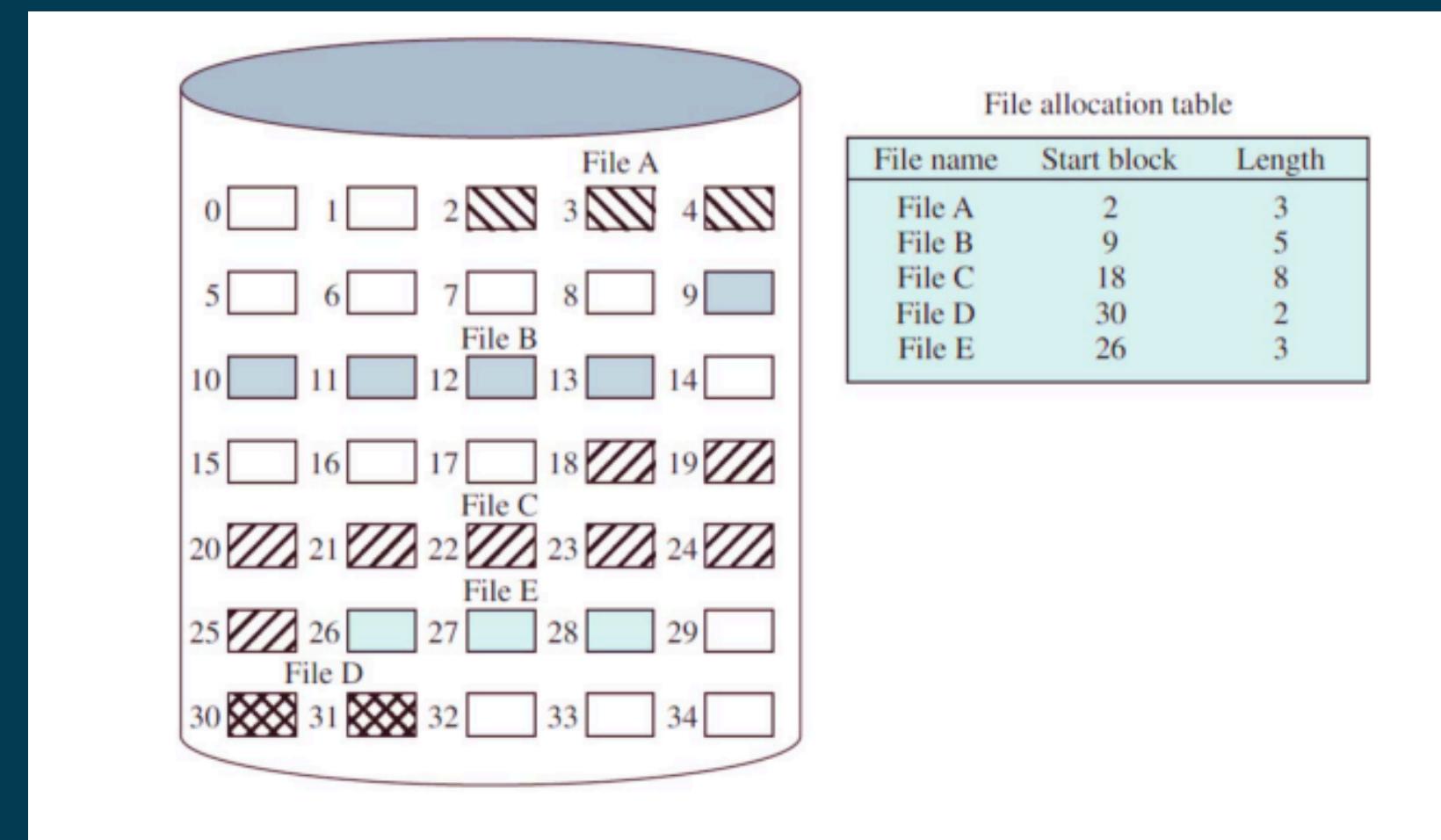
Resultado: Mayor tiempo de lectura/escritura y desorden.

Como se soluciona?

Asignación Contigua

- Se asigna una secuencia continua de bloques a un archivo.
- Acceso rápido y eficiente a datos secuenciales, ya que los bloques están físicamente uno al lado del otro.
 - Se mantiene una tabla con:
 - Bloque de inicio del archivo.
 - Tamaño total (en bloques).
 - Problema: Fragmentación externa.
 - Con el tiempo, puede ser difícil encontrar espacio contiguo libre.

Mediante la asignación de bloques



- Solución: Compactación del disco.
 - Reorganiza archivos para liberar espacio contiguo.
 - Menos costoso que compactar la RAM.

Asignación Enlazada

Cada archivo se almacena como una lista de bloques que pueden estar dispersos en el disco.

Estructura:

- Existe una tabla que guarda el bloque de inicio y el bloque final.
- Cada bloque contiene:
 - Los datos del archivo.
 - Un puntero al siguiente bloque en la lista.

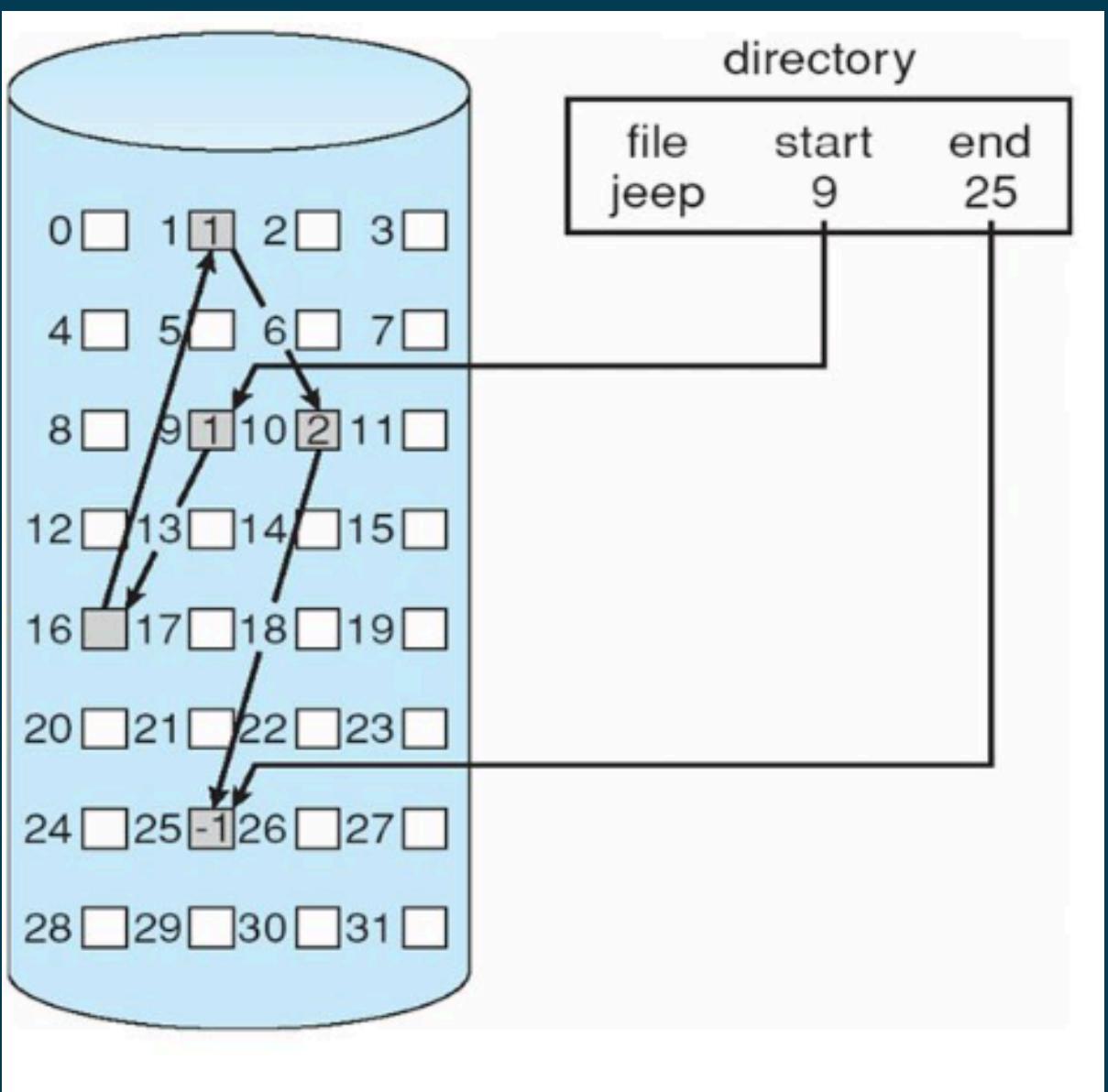
Ventajas:

- Los archivos pueden crecer de forma dinámica.
- No requiere bloques contiguos, lo que ayuda a evitar fragmentación externa.

Desventajas:

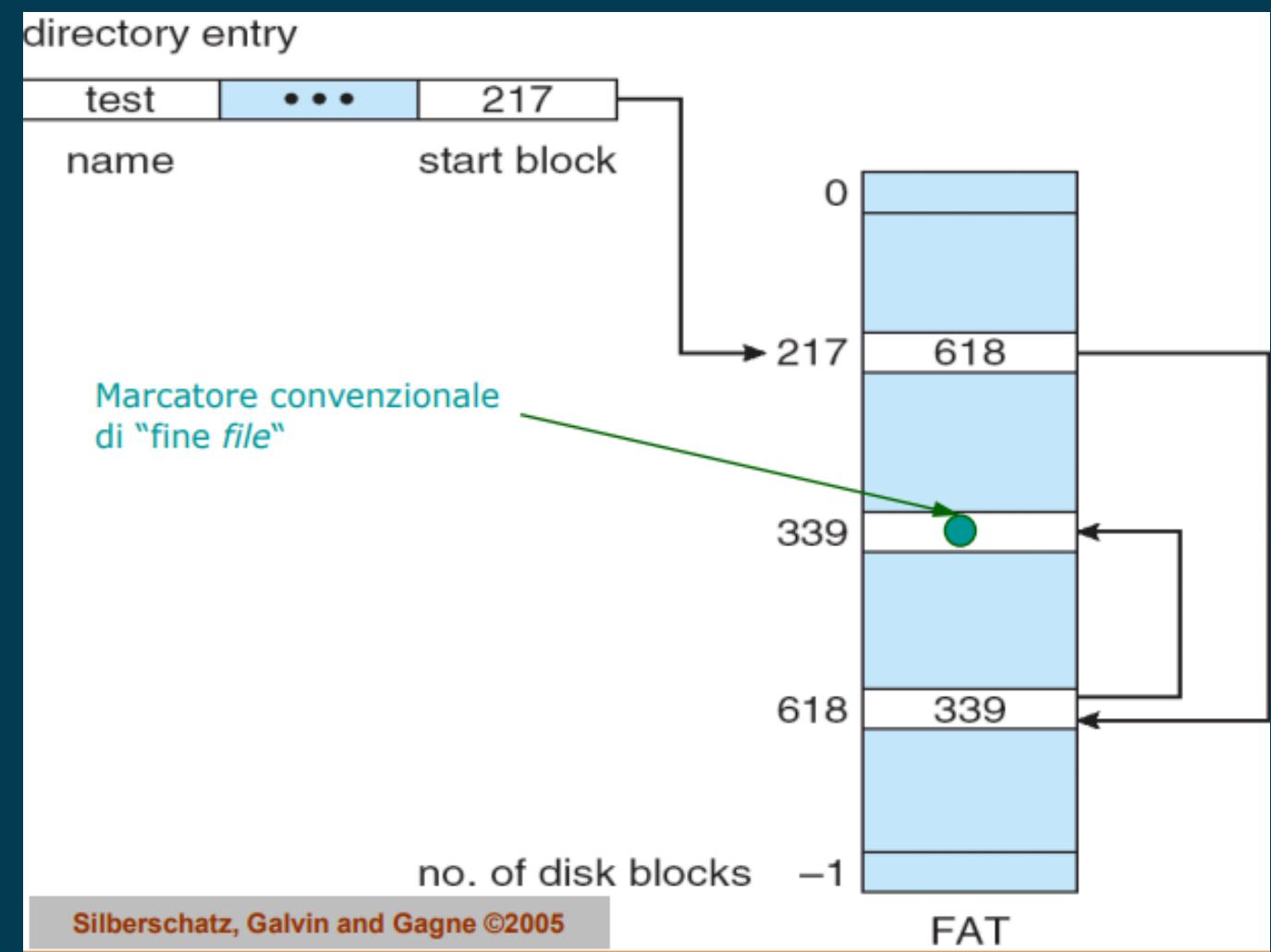
- No permite acceso aleatorio eficiente.
Para acceder a un bloque intermedio se debe recorrer la lista desde el inicio.
- Si un bloque se daña, se pierde el acceso a los bloques posteriores.

Es ideal para archivos que se acceden de forma secuencial.



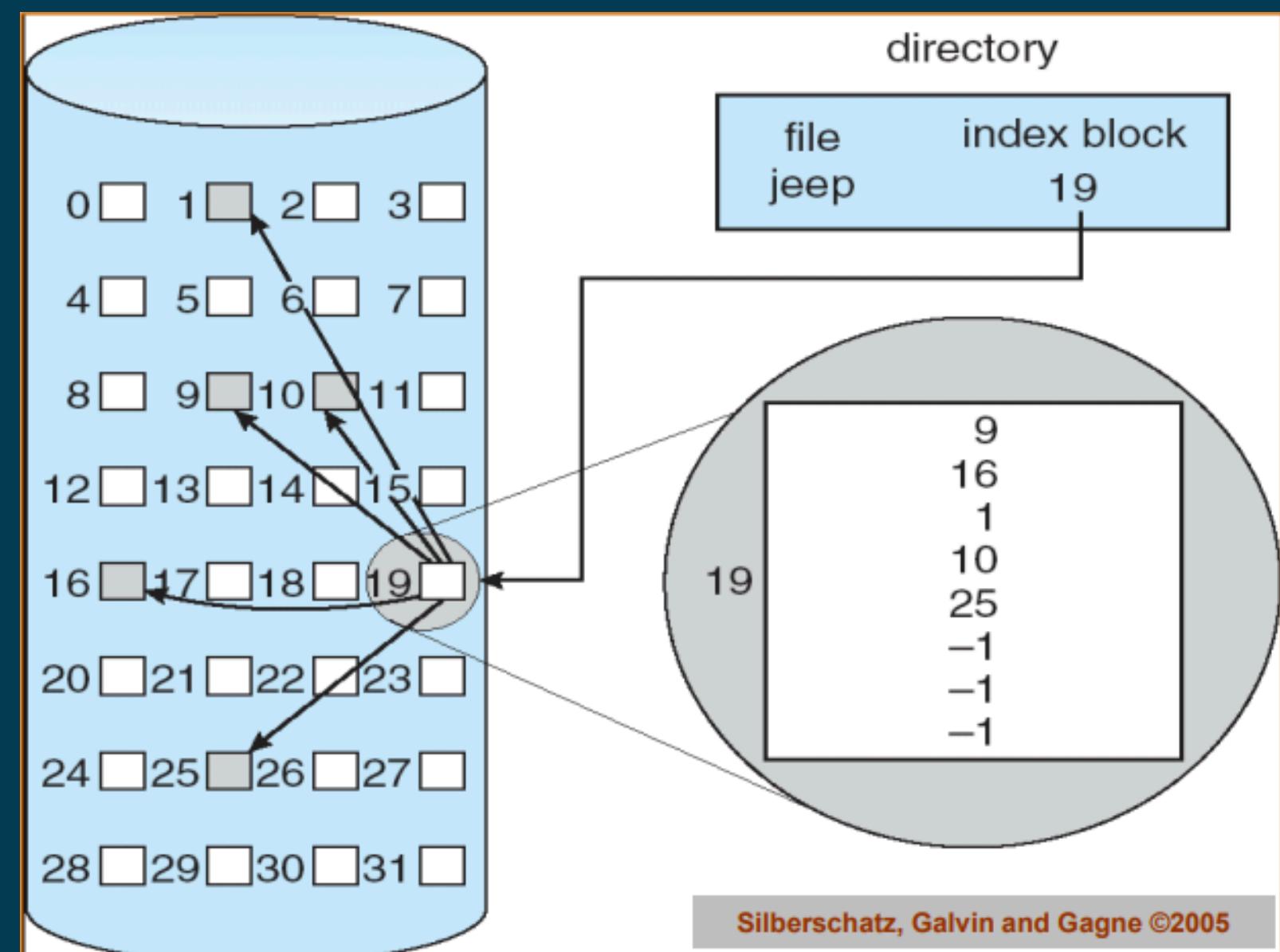
File Allocation Table (FAT)

- Es una variante mejorada de la asignación enlazada para gestionar archivos.
- Utiliza una tabla que registra el bloque inicial de cada archivo y los enlaces a los bloques siguientes, mejorando el acceso secuencial.
- El último bloque de un archivo contiene un valor especial (por ejemplo, -1) que indica el End Of File (EOF).
- Tiene limitaciones en el tamaño máximo de archivo según la versión:
- Por ejemplo, en FAT32, el tamaño máximo es de 4 GB (2^{32} bytes).



Asignación Indexada

- Cada archivo tiene un bloque índice que contiene punteros a todos los bloques de datos que lo componen.
- Permite acceso aleatorio eficiente a los datos, sin los problemas de fragmentación externa que afectan a otros métodos.
- El último bloque del archivo contiene un marcador de End Of File (EOF).
- Desventaja:
Requiere espacio adicional para almacenar el bloque índice de cada archivo, lo que puede aumentar el uso total de espacio en disco.



Tipos de Asignación Indexada

- Esquema enlazado

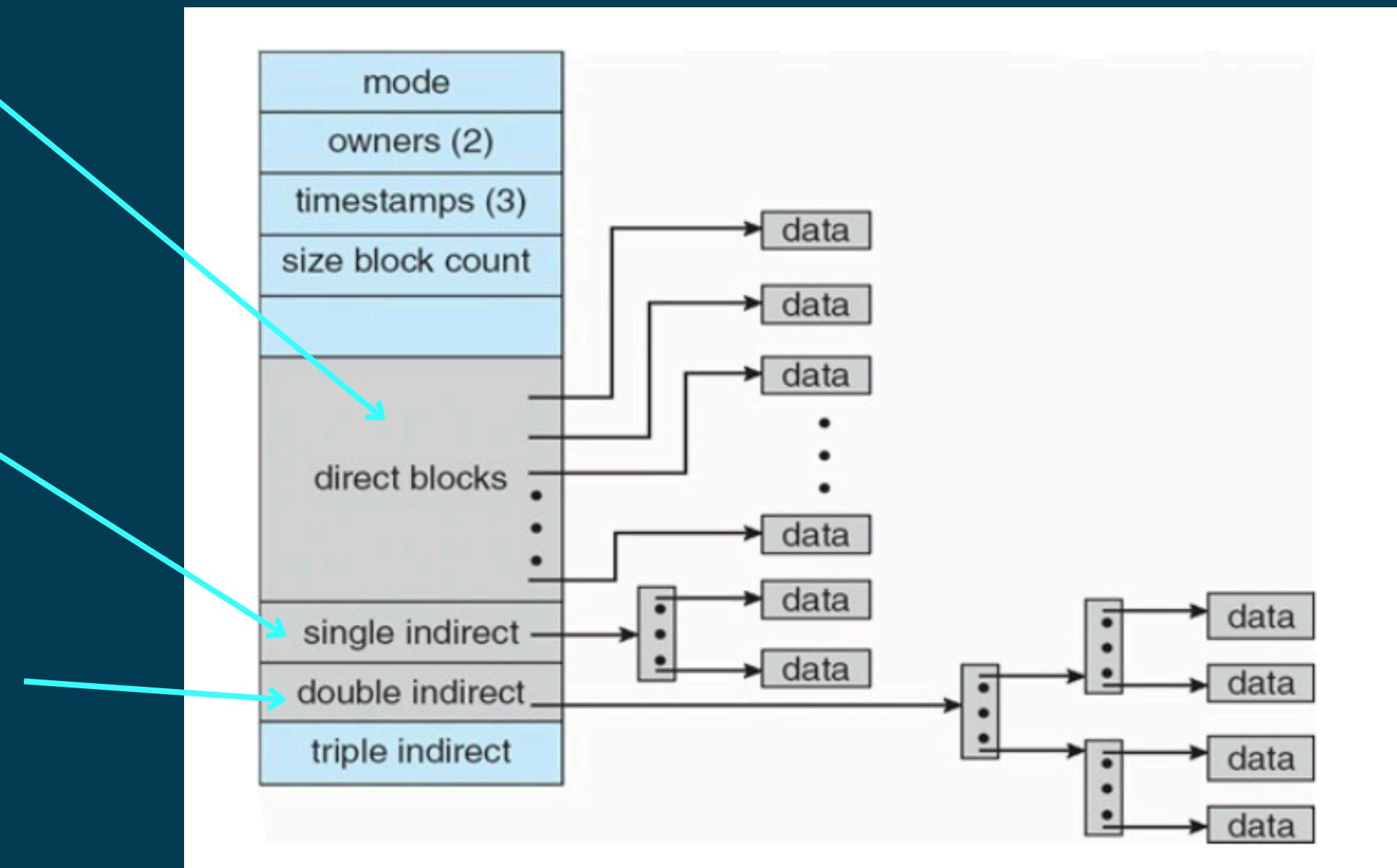
Utiliza una lista enlazada de bloques libres, donde cada bloque apunta al siguiente bloque libre disponible.

- Esquema multnivel

Emplea múltiples niveles de tablas de índices para administrar bloques libres de diferentes tamaños, optimizando la búsqueda y asignación.

- Esquema combinado

Integra listas enlazadas y tablas de índices para aprovechar las ventajas de ambos métodos, mejorando la flexibilidad y eficiencia en la gestión de bloques libres.



Como podemos encontrar espacio libre?

Bit Vector / Bitmap

- Es una serie de bits donde cada bit representa un bloque del disco.
- Un bit en 0 indica que el bloque está asignado.
- Un bit en 1 indica que el bloque está libre.
- Es fácil de implementar pero no escala bien en discos grandes debido a su tamaño fijo y recorrido secuencial.

Lista Enlazada

- Utiliza punteros para enlazar los bloques libres entre sí.
- Puede ser menos eficiente por la necesidad de mantener una estructura adicional.
- Sin embargo, es más escalable y flexible para manejar espacios libres dispersos.

Ejercicio 1: Links

Dada la siguiente secuencia de comandos:

```
touch brisket // crea un archivo brisket  
ln -s brisket macaroni-cheese // crea un soft-link macaroni-cheese hacia brisket  
ln brisket ribs // crea un hard-link ribs hacia brisket  
ln -s macaroni-cheese coleslaw // crea un soft-link coleslaw hacia macaroni-cheese  
ln brisket pulled-pork // crea un hard-link pulled-pork hacia brisket
```

Responda para cada caso que pasa si se ejecuta de manera secuencial los siguientes comandos:



	¿Qué links funcionan?	¿Cuántos archivos existen?
rm macaroni-cheese		
rm brisket		
touch brisket		
rm pulled-pork		

Ejercicio 2: File System

Supongamos que tenemos un disco con 20 bloques, enumerados del 0 al 19, y que necesitamos almacenar tres archivos en este disco:

Archivo A: 5 bloques | Archivo B: 3 bloques | Archivo C: 6 bloques

Se considerarán bloques completos para almacenar los bloques de los archivos y para marcar los índices (si es que es necesario) de dichos archivos. Determine la asignación de bloques para administrar el almacenamiento (de estos archivos) en el disco considerando:

- Asignación Contigua.
- Asignación Enlazada bajo esquema de punteros a bloques intercalados.
- Asignación Indexada.

*No olvide que si necesita marcar un índice, esto considera un bloque en el disco