Sistemas Operativos I

Sistemas de Archivos

"What is or is not implemented in the kernel represents both a great responsibility and a great power"

Ken Thompson

Rafael Ignacio Zurita <rafa@fi.uncoma.edu.ar>
Depto. Ingeniería de Computadoras
Facultad de Informática - UNCo

Advertencia: Estos slides traen ejemplos.

No copiar (ctrl+c) y pegar en un shell o terminal los comandos aquí presentes.

Algunos no funcionarán, porque al copiar y pegar tambien van caracteres "ocultos" (no visibles pero que están en el pdf) que luego interfieren en el shell.

Sucedió en vivo :)

Conviene "escribirlos" manualmente al trabajar.

Sistemas Operativos I - Sistemas de Archivos

Contenido

- Interfaz del sistema para gestión de
  - Archivos
  - Directorios
  - Protección
- Implementación de sistemas de archivos
  - Ejemplo: FAT
  - Ejemplo: UNIX - inodos
  - Encontrando bloques libres
- Integridad (Journaling, fsck)
- Protección (Backup, Journaling)
- Performance (buffer caché)

Sistema de Archivos

Un ***sistema de archivos*** es un software que gestiona objetos de datos permanentes, cuyos valores persisten luego de que el proceso que los creó y utilizó finalizó.

Los datos permanentes son mantenidos en ***archivos***.

Los archivos se guardan en un ***almacenamiento secundario***, como discos electromecánicos o de estado sólido.

Archivos

Los archivos son organizados en ***directorios***
(también llamado carpetas).

Conceptualmente, cada archivo es una secuencia de objetos de datos (usualmente, el dato básico es el byte).

Los archivos son creados y eliminados mediante **operaciones provistas por el sistema de archivos**.

También abrir, leer, escribir y cerrar.

Ejemplo de un conjunto de operaciones sobre archivos

Un principio común es que cada archivo es una secuencia de bytes.

Cualquier estructura de datos de esos bytes es una interpretación de los programas de aplicación.

- open
- read
- write
- close
- seek
- getc
- putc

Ejemplo de un conjunto de operaciones sobre archivos

Un principio común es que cada archivo es una secuencia de bytes.

Cualquier estructura de datos de esos bytes es una interpretación de los programas de aplicación.

- open()
- read()
- write()
- close()
- seek()
- getc()
- putc()

Otros:

truncate()

¿Estas operaciones, le recuerdan a algún otro subsistema de un sistema operativo?

Atributos o metadatos de los archivos

Información o atributos descriptivos o de control del archivo.
No son parte de los datos.

- Nombre
- id del archivo en el sistema de archivos (generalmente es un número)
- Tipo: permite reconocer clases de archivos
- Ubicación: puntero a la ubicación física en el dispositivo donde se encuentra
- Tamaño
- Dueño
- Protección: datos de control
- Fecha y hora: para registrar el momento en que fue creado, modificado, y accedido por última vez

Ejemplo de un conjunto de operaciones sobre directorios

- Estructura de datos que contiene información de archivos y subdirectorios.
- Cada entrada del directorio se corresponde con un archivo o subdirectorio.
- Es una especie de “índice”.

  - Crear un archivo
  - Borrar un archivo
  - Listar el directorio
  - Renombrar un archivo
  - Crear un subdirectorio
  - Borrar un subdirectorio

Protección

El SO se va a encargar de administrar

- ***cuales operaciones*** (read, write, execute, append, delete, list) se pueden realizar sobre un archivo; y
- ***quienes*** son los sujetos o usuarios que las pueden realizar.

En general el propietario o creador del archivo es quien normalmente tiene la capacidad de controlar quién accede y con cuáles operaciones, pero siempre es el SO quien las impone.

Protección

1. Acceso que depende de la identidad del usuario.
    - Cada archivo y directorio tiene asociada una lista de control de acceso (ACL), que especifica nombres de usuario y tipo de acceso permitido.
    - Dificultad: tamaño de la lista y su gestión.

2. Utilizar una clasificación de usuarios (ejem. Unix / Linux):
    a) owner access
    b) group access
    c) public access

    crear grupos y usuarios : /etc/passwd /etc/group
    man chmod

Principios de diseño

Métodos de asignación de bloques a archivos

Se refiere a las estrategias por las que se le asignan bloques de disco a los archivos:

- Contigua
- Enlazada
- Indexada

Sistemas Operativos I - Sistemas de Archivos

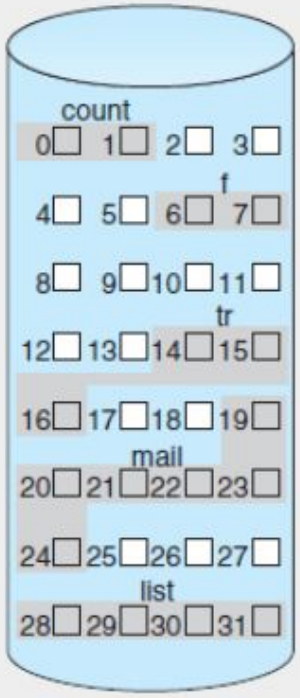
Principios de diseño
Métodos de asignación contigua

Un archivo ocupa una cierta cantidad de ***bloques contiguos***.

Soporta tanto el acceso secuencial como el directo.

Inconvenientes:

- Es complejo implementar el crecimiento de un archivo.
- Produce mala utilización del espacio en disco, debido a fragmentación externa.



Directorio

archivo	inicio	longitud
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

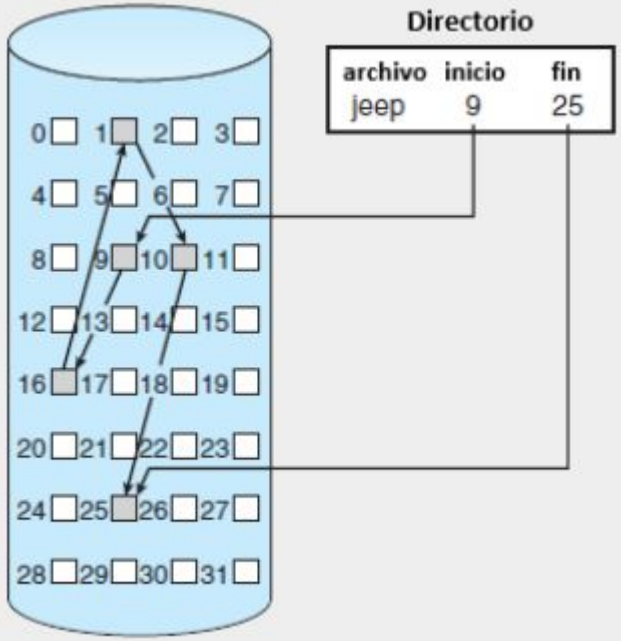
Principios de diseño

Métodos de asignación con lista enlazada

Un archivo ocupa una cierta cantidad de
bloques posiblemente dispersos.
Soporta tanto el acceso secuencial como el directo.
Cada bloque tiene un puntero al siguiente bloque.

Inconvenientes:

- Utiliza unos pocos bytes del bloque para punteros.
- Muchas lecturas a disco para alcanzar un bloque o byte.



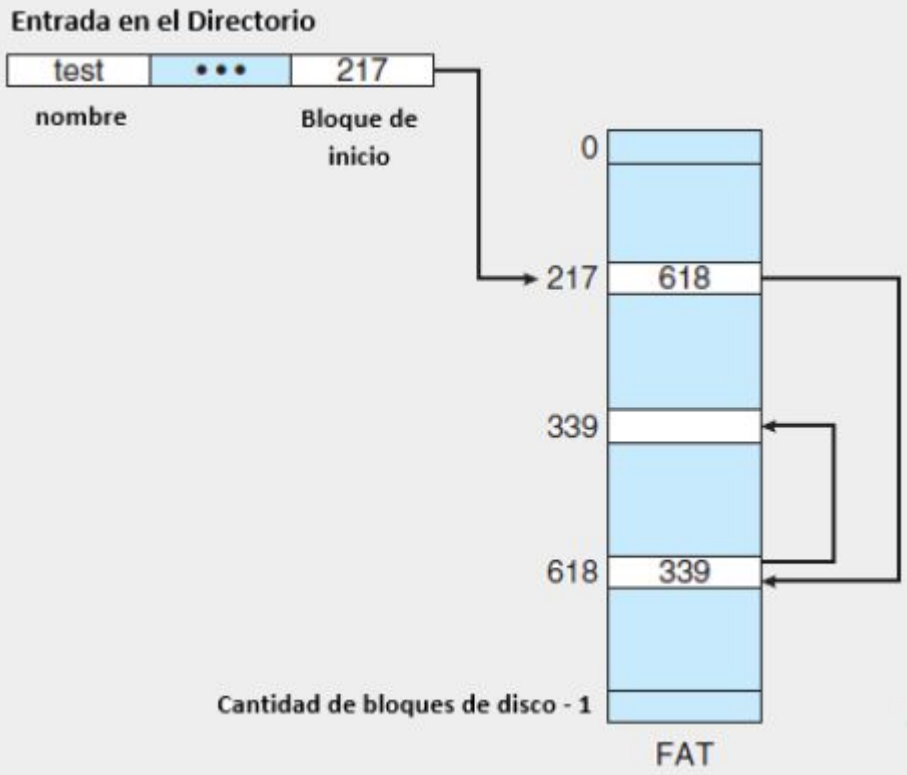
Sistemas Operativos I - Sistemas de Archivos

Principios de diseño

Método de tabla de asignación de archivos

Es una versión mejorada de lista enlazada

Punteros de la lista enlazada
en una tabla en memoria



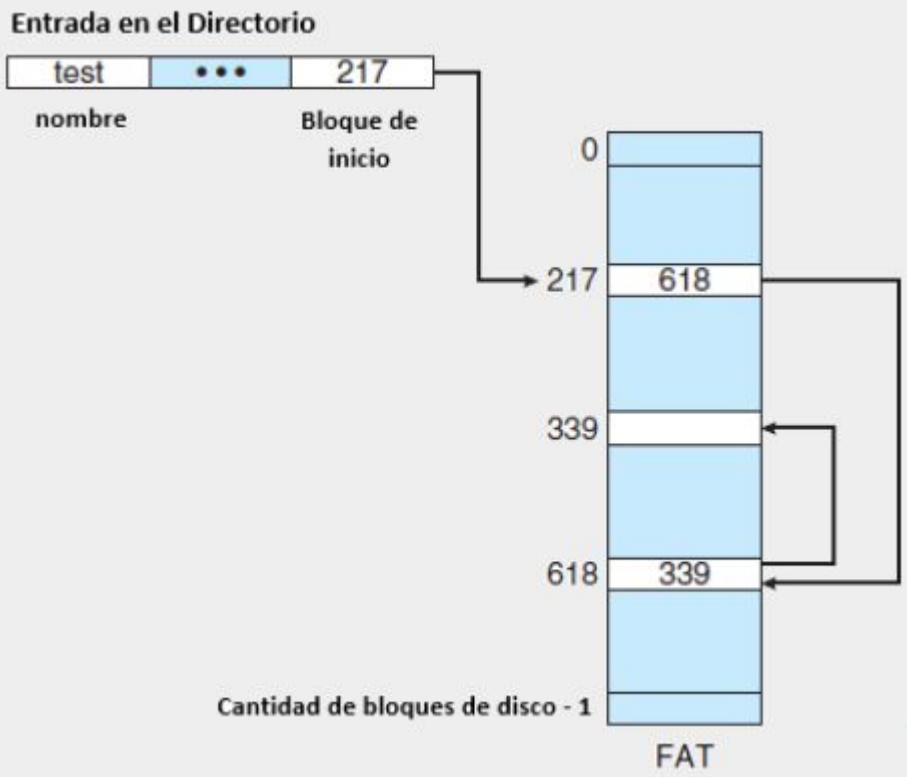
History CHANNEL:

cp/m, DOS, Windows 95, Windows 98
FAT12, FAT16, FAT32
Nuestros pen drives!

Sistemas Operativos I - Sistemas de Archivos

Principios de diseño
Método de tabla de asignación de archivos

Es una versión mejorada de lista enlazada



History CHANNEL:

cp/m, DOS, Windows 95, Windows 98
FAT12, FAT16, FAT32
Nuestros pen drives!

¿Se acordarán de nuestros sistemas en 20(

Sistemas Operativos I - Sistemas de Archivos

Para indagar mas:

FAT

https://social.technet.microsoft.com/wiki/contents/articles/6771.the-fat-file-system.aspx

https://people.cs.umass.edu/~liberato/courses/2017-spring-compsci365/lecture-notes/11-fats-and-directory-entries/

Sistemas Operativos I - Sistemas de Archivos

Ejemplo de tabla con punteros. FAT : File Allocation Table

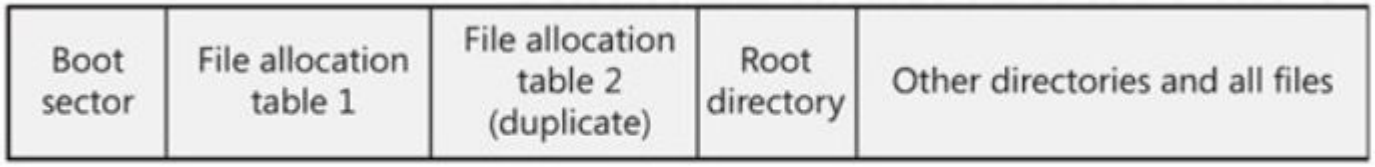


Figure 1: FAT file system volume organization.

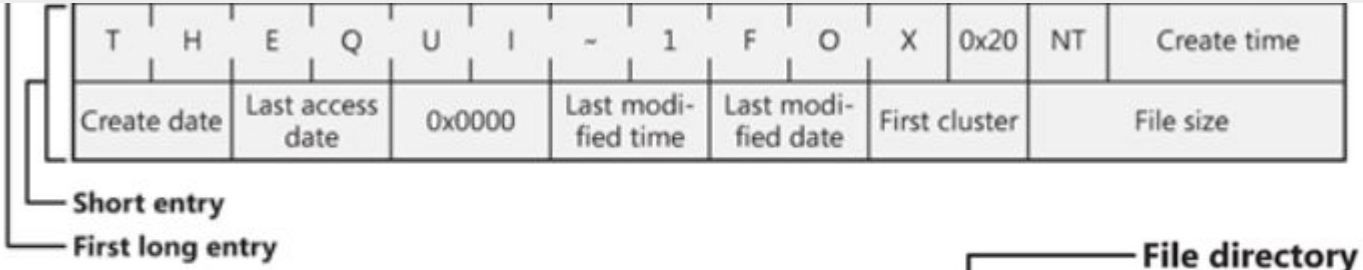
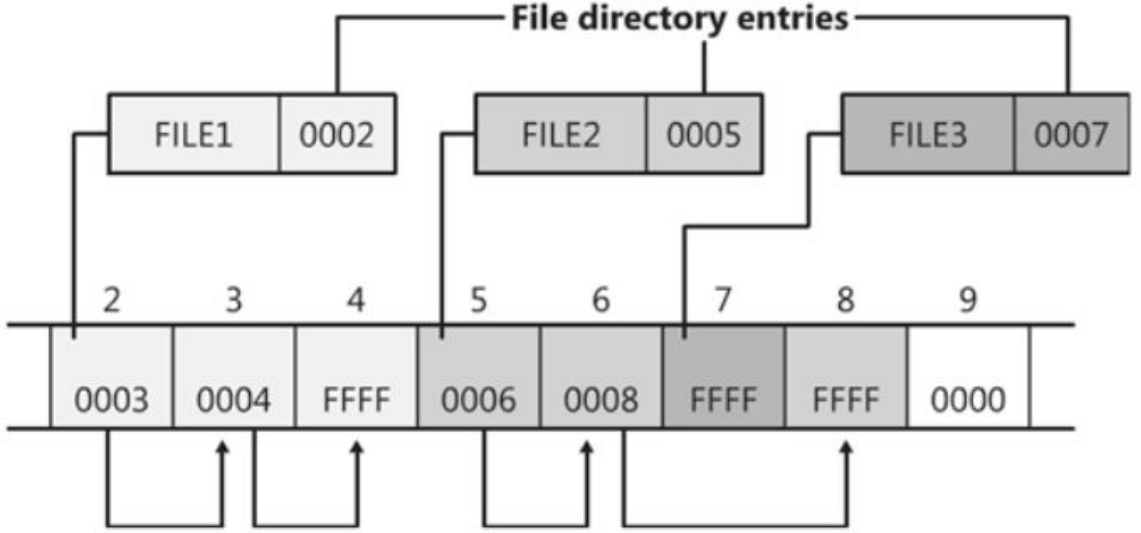


Figure 3: a FAT directory entry.

Explicación de cómo funciona en vivo!



Entrada de directorio. Tipo :

0x20 Archivo
0x10 Directorio

FAT16: 128KB en RAM

Maximo tamaño de bloque 32KB.

Maximo tamaño de archivo: 2GB

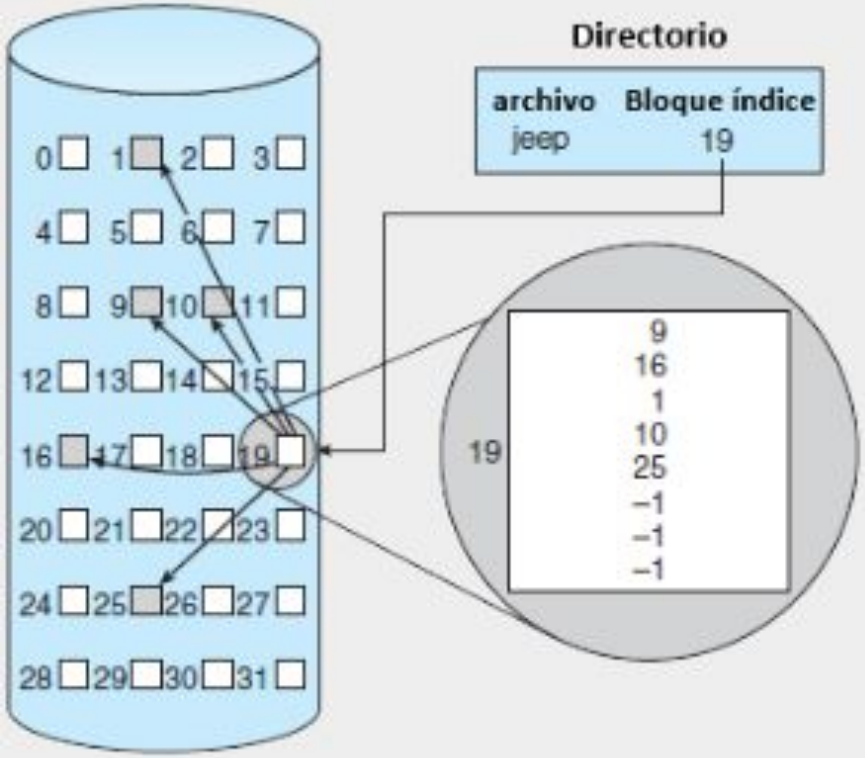
Principios de diseño

Método de asignación de bloques con índices

Todos los punteros a bloques de disco de un archivo se almacenan en un bloque de índices.

Cada archivo tiene su propio bloque de índices.

En la entrada del directorio hay un puntero al bloque de índices.



Principios de diseño

Método de asignacion de bloques indexada.

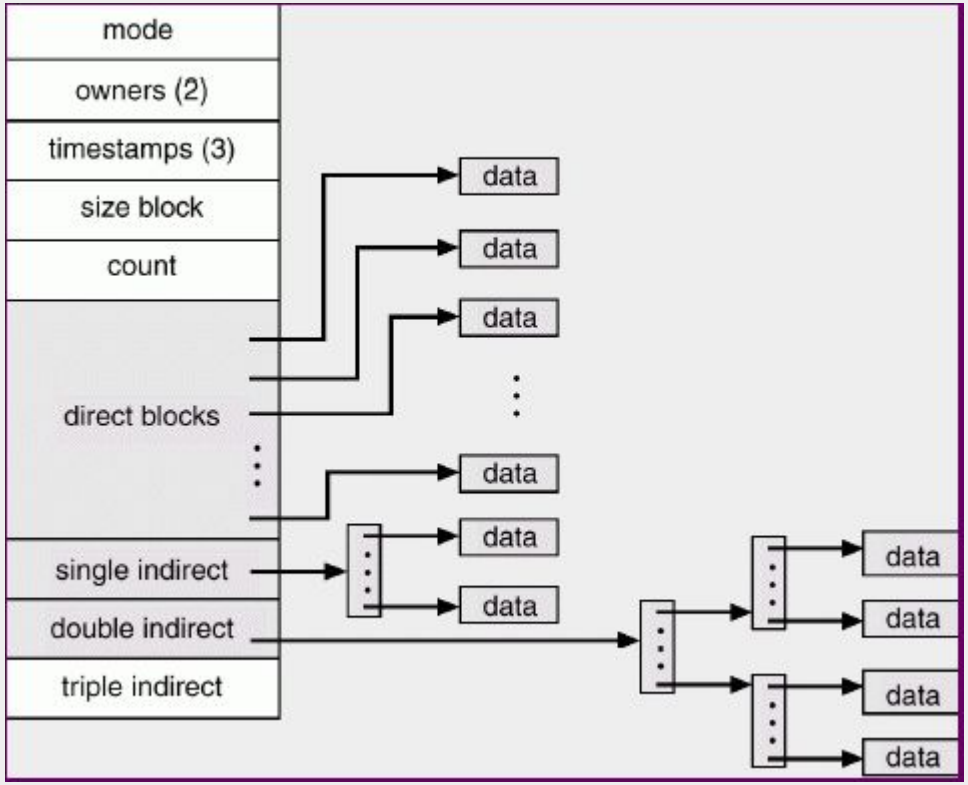
Ejemplo: UNIX/Linux i-nodes

Todos los punteros a bloques de disco de un archivo se almacenan en un bloque de índices.

Cada archivo tiene su propio bloque de índices

En la entrada del directorio hay un puntero al bloque de índices.

¿Ventajas?



Sistemas Operativos I - Sistemas de Archivos

Ejemplo: UNIX/Linux i-nodes

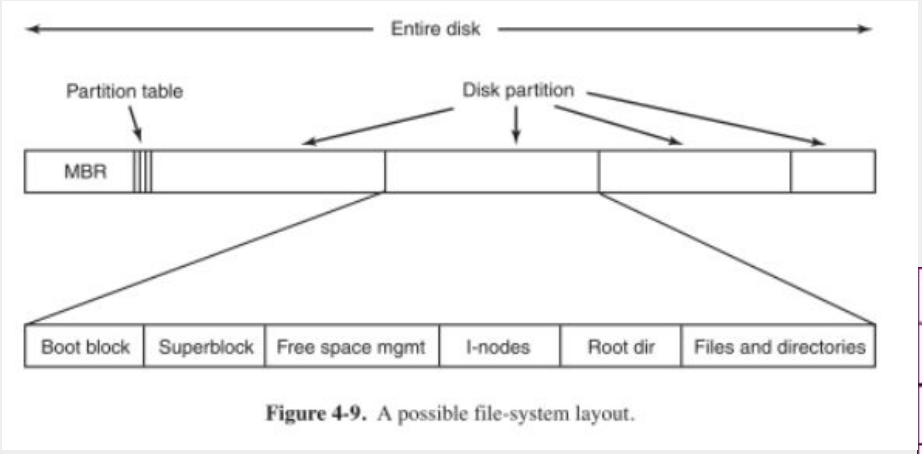
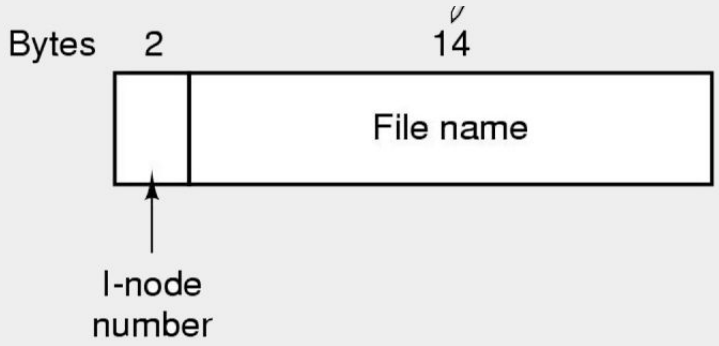
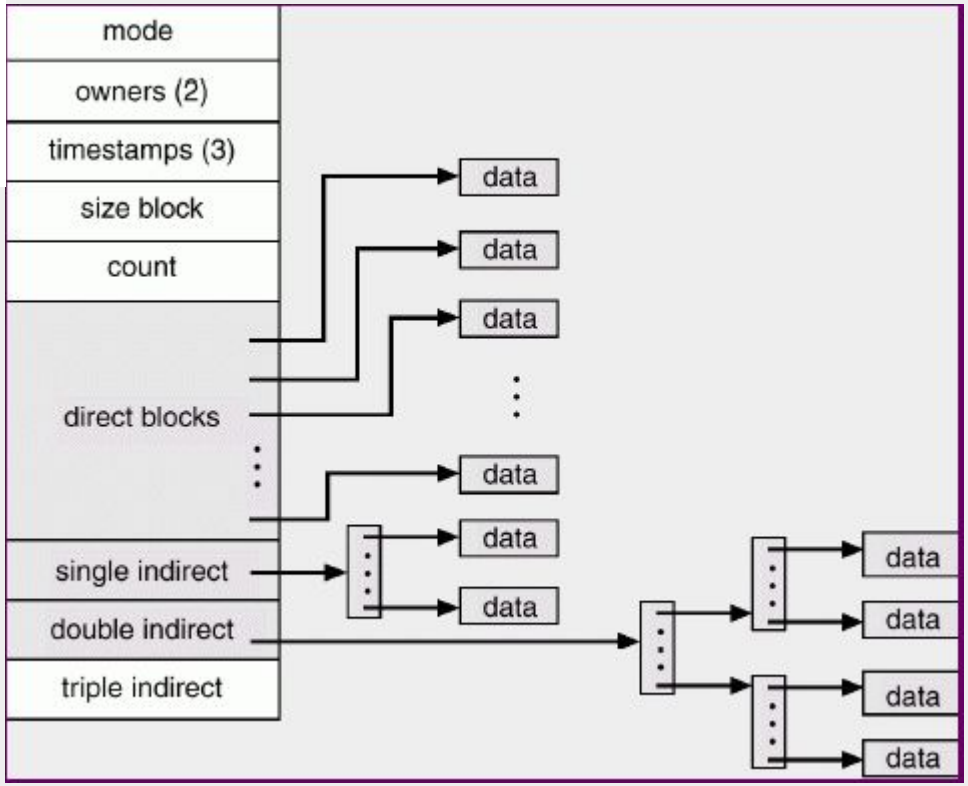


Figure 4-9. A possible file-system layout.





- Listado de bloques libres (gestión del espacio libre)
- Listado de inodos utilizados
- Tamaño de cada inodo
- Cantidad total de inodos
- Entrada de directorio

Explicación de cómo funciona en vivo!

Sistemas Operativos I - Sistemas de Archivos

Transparencia en espacio de usuario (UNIX)

- un único árbol de directorios
- soporte a diferentes tipos de sistemas de archivos
- acceso a dispositivos utilizando las mismas primitivas
- diferentes sistemas de archivos montados y accedidos vía directorios

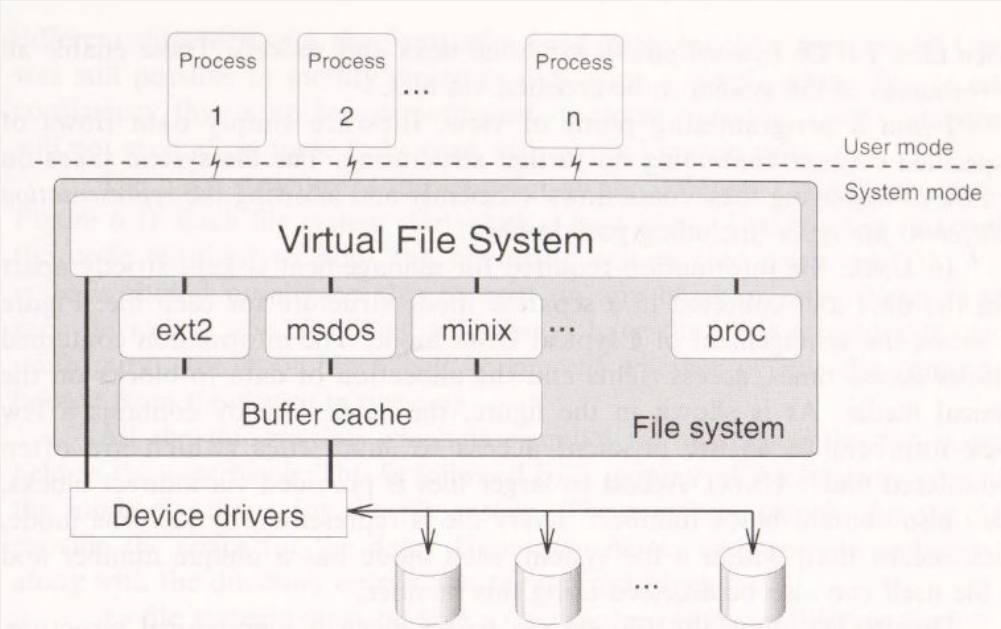


Figure 6.1 The layers in the file system.

Confiabilidad, recuperación y rendimiento

- backups (incremental dumps)
- fsck y logFS

- bloques contiguos
- buffer caché
- read ahead
- logFS (escrituras)

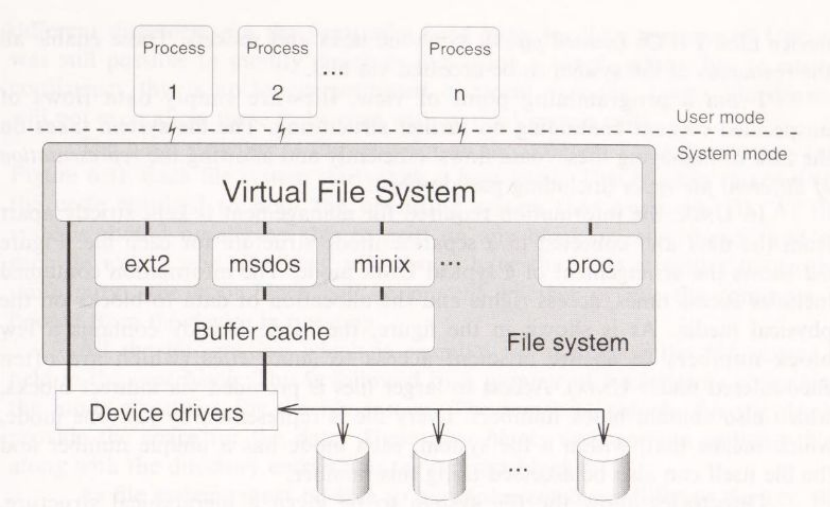


Figure 6.1 The layers in the file system.

Sistemas Operativos I - Sistemas de Archivos

CP/M is clearly not the last word in advanced file systems, but it is simple, fast, and can be implemented by a competent programmer in less than a week. For many embedded applications, it may be all that is needed.

Andrew TANENBAUM