

1. Interfaz de los sistemas de archivos

Un archivo es más que una colección de información relacionada y almacenada en un dispositivo de almacenamiento secundario de acuerdo con este "almacenador" con direcciones lógicas contiguas, por tanto su espacio de direcciones lógicas es contiguo.

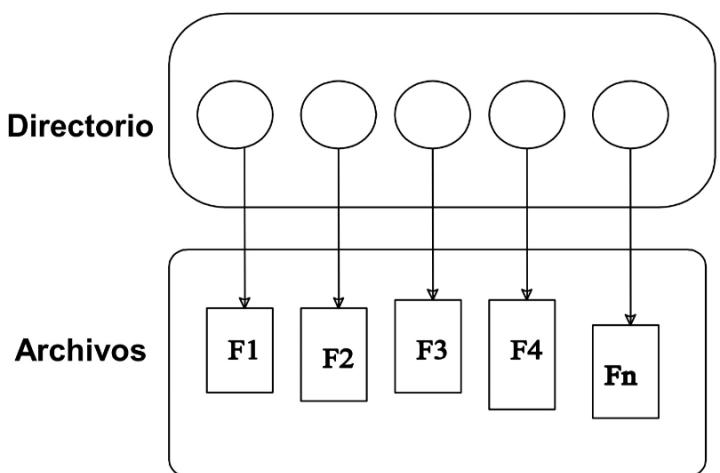
Dispone de una estructura interna lógica con secuencia de bytes que depende del tipo de archivo.

Hay como tipos los archivos regulares, directorios o dispositivo y formas de acceso secundario y alternativas entre otras,

Cada archivo dispone de unos atributos: número, tipo, localización, tamaño, protección, temporales.

1.1. Directorios

Un directorio es un archivo que contiene una colección de bytes con información de sus distintos archivos, residiendo en almacenamiento secundario.



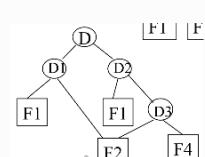
Su organización debe ser efectiva, independiente según el usuario, que use el dispositivo y agrupar los archivos de forma lógica de acuerdo a sus propiedades.

Agrupación en árbol

Sigue cumpliendo la mayoría de las expectativas que se usa, si no que sea en el grafo.

Agrupación en grafo

Permite que se hagan referencias duplicadas por directorios debido a la compartimentación de subdirectorios y archivos. Es más flexible y complejo.



1.2. Protección

Consiste en proporcionar un acceso controlado a los archivos, es decir, qué se puede hacer y qué no.

La primera solución y principal es hacer que el acceso dependa del uso, de esta manera, son las dotas de acceso aquellas que tienen importancia. Además, todo se puede solucionar con privilegios según la clase de usuario (User-group-classes). Lectura

1.3. Semánticas de consistencia

Especifican cómo las modificaciones de datos por un usuario se observan por los demás.

- Unix: siempre y permite que varios usuarios el puntero "base".
- De Gestión: cuando pone cuando un archivo se cierra, sus cambios se observan en sesiones posteriores. git hub.
- Inmutables: Si se declara compartido \Rightarrow no se puede modificar.

1.4. Funciones básicas del Sistema de Archivos

- Tener conocimiento de todos los archivos del sistema.
- Controlar la coexistencia y fusionar la protección.
- Gestionar el espacio
- Traducir de lógicos a físicos

2. Diseño software del sistema de archivos

Problemas:

1) ¿Cuáles debe ver el usuario el sistema de archivos?

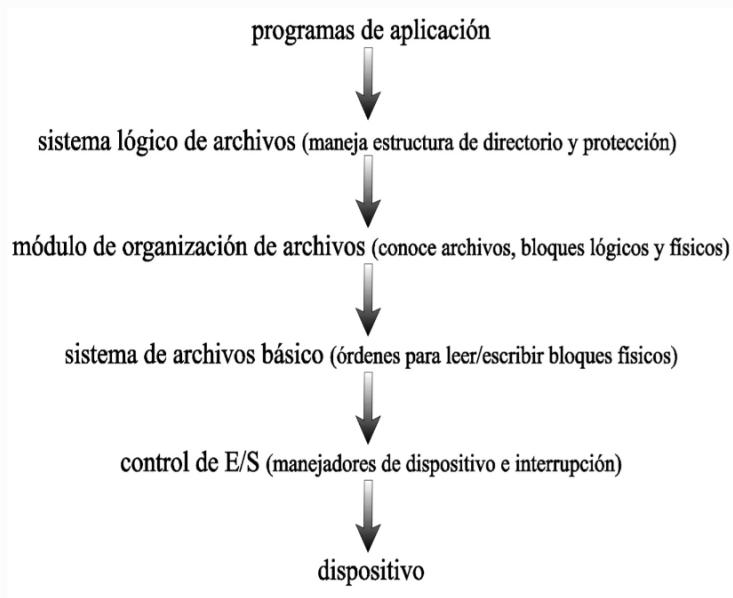
- i) Definir el archivo y sus atributos
- ii) Operaciones
- iii) Estructura de las direcciones

2) ¿Cómo se corresponden el sistema de archivos lógico y físico?

- i) Algoritmos
- ii) SD

2.1 Estructura

Depende de una organización en niveles donde el sistema maneja bloque lógico de archivos abiertos que contiene los descriptores de archivos y como es usa el bloqueo controlado archivo, que contiene la información de un archivo abierto.



2.2. Métodos de asignación

Contigo

Cada archivo ocupa un conjunto de bloques contiguos en disco.

Ventajas

- Sencillo: (inicio, longitud)
- Buen acceso secuencial y directo

Desventajas

- No conserva el tamaño de archivo en el momento de su creación
- Pierde espacio → fragmentación externa
- No crecimiento de archivos → apagando el disco

Además, depende de la siguiente traducción:

- si los bloques de disco son de tamaño k:

$$\text{logí}/u \rightarrow c(\text{cylinder}), r(\text{sector})$$

$$\text{Bloque a acceder} = c + \text{dirección de sector}$$

Desplazamiento: r

Estructura

Un archivo es una lista enlazada de bloques, dentro que pueden estar dispersos entre ellos.
 En realidad es totalmente autónoma.

Ventajas

- Evita fragmentación externa
- Creamos directamente (si hay bloques libres)
- Solo almacenar puntero al primer bloque

Desventajas

- Acceso directo ineficiente
- Espacio requerido para almacenar los punteros → Registers (solución)

Si tenemos direcciones de uB ⇒

$$\text{Logico} / k_u = C, r$$

$$\text{Bloque} = C$$

$$\text{Desplazamiento} = r + u$$

Bloques FAT
Físicos

| |
|-----|
| 0 |
| 1 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| * |
| 7 |
| 2 |
| 8 |
| 9 |
| ... |

← A

Como variación, se usa la Tabla de asignación de direcciones (FAT), consiste en tener una tabla del disco al comienzo de la partición para almacenar en cada entrada (una por bloque en disco) la dirección del bloque a leer.

Es simple y eficiente mientras esté en cache; sin embargo, una pérdida de punteros causaría un error, por tanto, se mantiene esta seguridad.

Indexado

Disponemos de un bloque indexado donde almacenamos los punteros a los distintos bloques. El directorio contiene la localización de cada bloque índice y cada archivo tiene su propio índice. El último bloque está en la última entrada.

Ventajas

- Permiten acceso directo
- Evita fragmentación interna

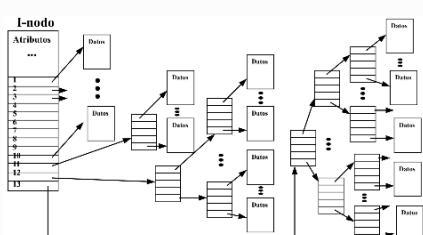
Desventajas

- Dependencia de espacio en los bloques libres
- Tamaño de BI

i) Estructuras

ii) Multivínculo: para evitar multivínculos a disco se mantienen algunos bloques en memoria principal.

iii) Estructura combinada.



2.3. Gestión de espacio libre

- Mantener una lista de los bloques libres: lista de espacio libre.
- Faz uso necesario su implementación

Mapa o vector de bits

Se usa un bit para decir si el bloque es libre o ocupado. Es fácil de usar para encontrar bloques libres y tener archivos en bloques contiguos pero es ineficiente si uno mantiene su memoria principal.

Lista encadenada

Se crea una lista encadenada de bloques libres por la que se sigue para encontrar el primer bloque libre. Se demanda espacio pero es ineficiente.

Lista encadenada con agrupación

Consiste en una lista encadenada de una direcciones de bloques libres por lo que obtener muchas direcciones de bloques libres es rápido.

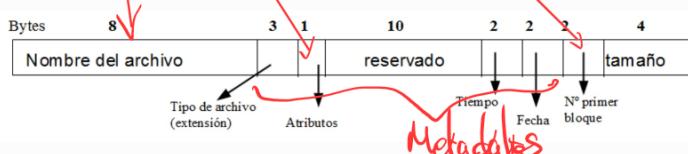
Resumen

Encadenada de la lista seguido el nº de bloques libres que tiene.

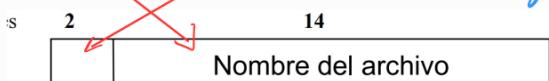
2.4. Implementación de directorios

Hay varias formas:

- i) Máximo archivo - atributos - Dirección de los datos



- ii) Máximo archivo - Puntero a la estructura de información



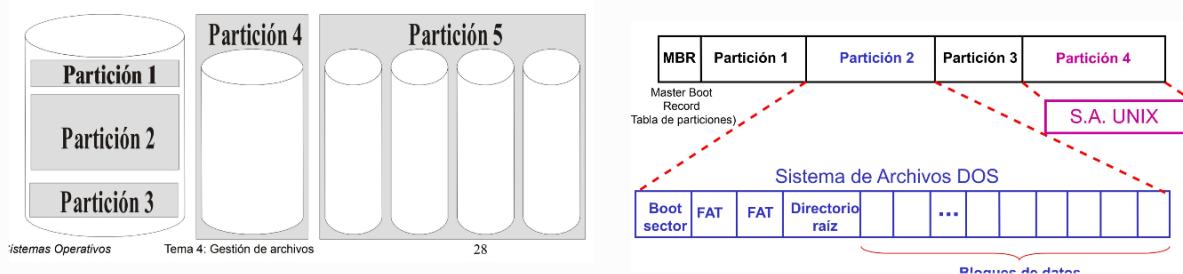
Para abrir un archivo, el sistema busca su entrada en el directorio, extrae sus atributos y la localización de sus bloques y los almacena en una tabla en la memoria que solo se manipula esa tabla.

Para archivar compartidos:

- **Archivos simbólicos**: es una nueva entrada en el directorio, se indica su tipo y se almacena su mismo directorio; se usa cuando hay un grande de directorios o discos y en entornos distribuidos.
- **Archivos absolutos**: se crea una nueva entrada en el directorio y se copia la dirección de la estructura de datos con la información del archivo. Para controlar el borrado se usa un controlador de entradas.

2.5 Distribución del sistema de archivos

Se almacenan en **diseños** que se pueden partitionar:



El **formato del disco** puede ser **físico** o **lógico**. En el primer caso, pone estructura por pista, un sector es una cabeza y el código de errores. En el segundo caso, escribe la información que el ordenador necesita para llegar a los contenidos del disco.

2.6 Recuperación

Como todo tiene una segunda **copia**, ya sea en un archivo, se debe garantizar que no haya perdido datos.

Técnicas:

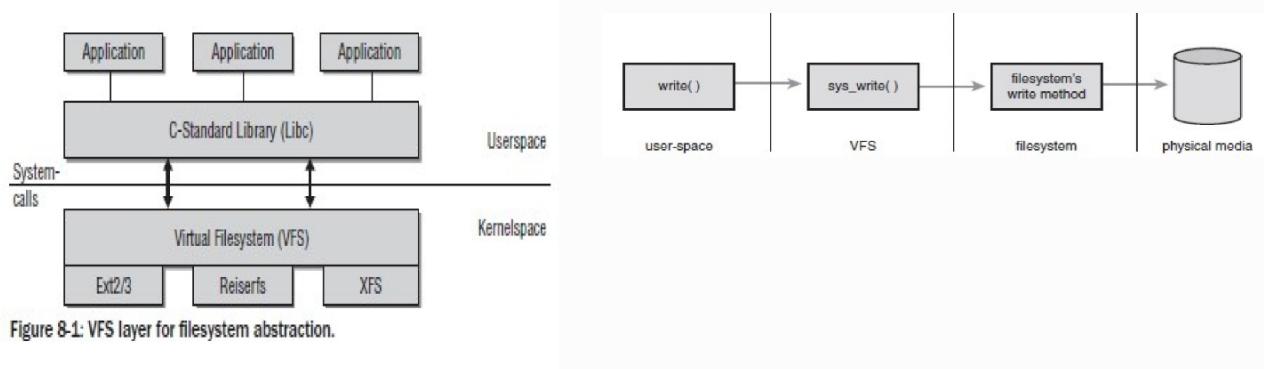
1. **Comparación de contenido**: Comparar lo que hay en un contenedor que hay en el disco (más fácil en listas jerárquicas que en bloques de texto)
2. Realizar **backups** (favorito del profes)

3. Linux

En este sistema la **representación interna** del archivo es un **node** que incluye **pathname**, sin embargo, puede tener muchos **archivos**.

- **Crear archivo** → asignar **node**
- **Reforestar archivo** → se actualizan permutos y se libera el node que se cierre.

Hoy en día Linux usa ext4.



Hay tres clases generales de sistemas de archivos:

- **Basados en disco:** forma física de almacenar archivos en medios no volátiles
- **Virtual:** generadas por el kernel y constituyen una forma sencilla para permitir la comunicación entre los programas y los usuarios; no se guarda almacenamiento hardware.
- **de Red:** a través de red.

Para un proceso, un archivo se identifica por su descriptor de archivo que lo engaña el kernel y es válido sólo dentro de un proceso; cada archivo tiene asociado un inode que contiene:

- URGID
- Tipo
- Permisos
- Tiempo de creación
- Contador de entradas
- Direcciones de los datos
- Tamaño

} Metadatos

Estructura VFS

Es un **virtual filesystem** para objetos, consta de directorios y **que necesitan gestionar**.

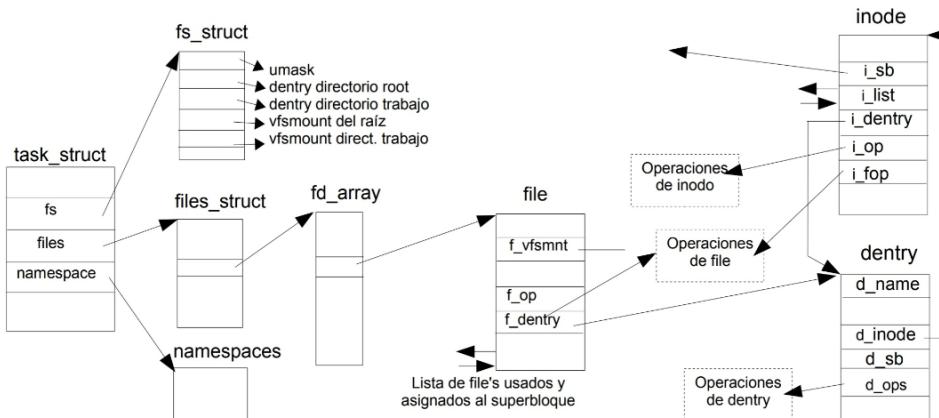
Los archivos y directorios son representados con **ED en C**.

Típos de objetos principales:

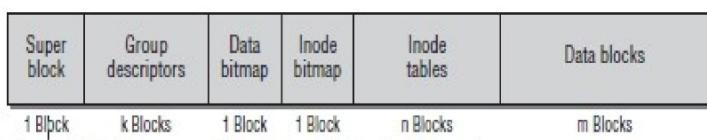
- **superblock:** se monta
- **inode:** Archivo
- **directory:** Entrada a un directorio
- **file:** Archivo abierto y su estructura por proceso

Cada objeto tiene un **vector** con sus **operaciones object_operations**.

Estructuras de datos VFS



Ext2



Donde el disco duro en un conjunto de bloques de igual tamaño que almacenan los datos de los archivos que los utilizan.

El elemento central es block group

Sistema de archivos Ext2 (y II)

- Cada SA consta de un gran número de grupos de bloques secuenciales
- Boot sector (Boot block): zona del disco duro cuyo contenido se carga automáticamente por la BIOS y se ejecuta cuando el sistema arranca
- Cuando se usa un SA (se monta), el superbloque, sus datos, se almacenan en MP.



Figure 9-3: Boot sector and block groups on a hard disk.

Descripción de un Grupo de Bloques

- Superbloque (Superblock):** estructura central para almacenar meta-information del SA.
- Descriptores de grupo (Group descriptors):** contienen información que refleja el estado de los grupos de bloques individuales del SA., por ejemplo, el número de bloques libres e inodos libres.
- Mapa de bits de bloques de datos y de inodos (Data bitmap, Inode bitmap):** contienen un bit por bloque de datos y por inodo respectivamente para indicar si están libres o no.
- Tabla de inodos (Inode tables):** contiene todos los inodos del grupo de bloques. Cada inodo mantiene los metadatos asociados con un archivo o directorio del SA.

