Sistema de archivos

Conceptos

- En general un archivo es una secuencia de bits cuyo significado el creador y usuario define

- **SISTEMA DE ARCHIVOS**: conjunto de módulos del SO que se encargan de la gestión de la información (archivos)

Organización del sistema de archivos

Un sistema de archivos presenta dos problemas de diseño muy distintos:

- Definir que aspecto debe presentar el sistema de archivos a los usuarios (atributos, operaciones, estructura de directorios, etc...)

- Definir los algoritmos y estructuras de datos que permiten mapear el sistema de ficheros lógico sobre los equipos físicos

Finalidad del Sistema de Archivo

El disco duro externo, o interno de tu ordenador, un pendrive USB y la tarjeta SD. Todos ellos son unidades de almacenamiento, cuando se formatean se crea la infraestructura en la que se almacenan los datos.

El sistema de archivos, es un componente del sistema operativo que se encarga de administrar la memoria de cada unidad además asigna a los archivos el espacio que necesiten, los ordena, permitir el acceso a ellos y administrar el espacio libre de las unidades de almacenamiento. Al ordenar y registrar la posición exacta en la que se ha escrito un fichero dentro de la unidad, el sistema operativo accede rápidamente a ellos y sabe en dónde empieza y termina cada uno.

Cada sistema de archivo gestiona los datos de formas diferentes. Cada sistema de archivos posee sus propias ventajas y limitaciones, por lo que es importante el conocerlos para elegir adecuadamente el que mejor se ajuste a cada necesidad de los usuarios.

Sistema de archivos FAT32

FAT (FILE ALLOCATION TABLE). Se establecido en 1996, es uno de los viejos formatos del mundo entre los sistemas de archivo, robusto pero obsoleto, siendo muy versátil gracias a su enorme compatibilidad con prácticamente todos los dispositivos y sistemas operativos, razón por la que la mayoría de unidades USB nuevas estarán formateadas con él.

Su mayor y más popular limitación es que sólo permite guardar archivos de hasta 4 GB, por lo que si se desea guardar un único archivo que ocupe más de 4 GB la única opción será formatear con otro sistema de archivos. Su lado positivo es que es perfectamente compatible con Windows, MacOS y GNU/Linux, además de funcionar sin problemas en los viejos USB 2.0.

Sistema de archivos exFAT

En esencia el sistema exFAT es como una actualización al FAT32 introducida por Microsoft en Windows Vista con la intención de acabar con los problemas que provoca la limitante de 4 GB de su predecesor.

En cuestión de compatibilidad puedes usarlo en Windows, MacOS o GNU/Linux, aunque sólo en las versiones más recientes como a partir de Windows XP SP3 u OS X 10.6.5 Snow Leopard. Es un sistema de archivos muy recomendado para unidades externas como un USB o tarjeta SD donde vayas a guardar archivos de más de 4 GB y no se quiera tener problemas de compatibilidad.

Sistema de archivos NTFS

NTFS (NEW TECHNOLOGY FILE SYSTEM). Otra alternativa al sistema FAT32 promovida por Microsoft, de hecho es el sistema de archivos que Windows utiliza por defecto. Sin los límites del tamaño máximo de archivo del FAT32, el NTFS se convierte en una muy buena opción para discos duros y otras unidades externas (solo usuarios de Windows).

Su mayor desventaja es no ser totalmente compatible con todos los sistemas operativos. Por ejemplo, de forma nativa MacOS leé las unidades formateadas con él, pero no puede escribir en ellas. Esto quiere decir que si tienes un disco duro con NTFS no se podrá guardar nada de un sistema Mac a no ser que se formateé con otro sistema de archivos.

Sistema de archivos HFS+

De la misma manera que el NTFS es uno de los actuales sistema de archivo de referencia en Windows, Apple creó el sistema HFS+ a su medida.

En los sistemas GNU/Linux se pueden trabajar con él sin problemas, mientras en Windows sólo se podrá leer el contenido de los discos formateados con HFS+, pero no escribir en ellos.

Sistema de archivos Ext2, Ext3 y Ext4

EXT# (# EXTENDED FILESYSTEM). Es un sistema de archivos principalmente utilizado en distribuciones Linux con registro por dia (journaling). Está siendo reemplazado por su sucesor, ext4 (FOURTH EXTENDED FILE SYSTEM), aunque todavía se continua utilizando. El principal inconveniente es que sólo puede ser utilizado con la familia de sistemas operativos propios de GNU/LINUX.

Métodos de acceso

- **Archivo**: secuencia de registros lógicos de longitud fija
- ¿De qué manera se accede a la información almacenada en los archivos?
  - Algunos SO ofrecen un solo método de acceso mientras que otros ofrecen diferentes métodos de acceso
- ACCESO SECUENCIAL
  - Se basa en un modelo de archivo de cinta
- ACCESO DIRECTO o RELATIVO
  - Se basa en el modelo de archivo de disco
- ACCESO INDEXADO
  - Requiere de estructuras adicionales: tablas de indices

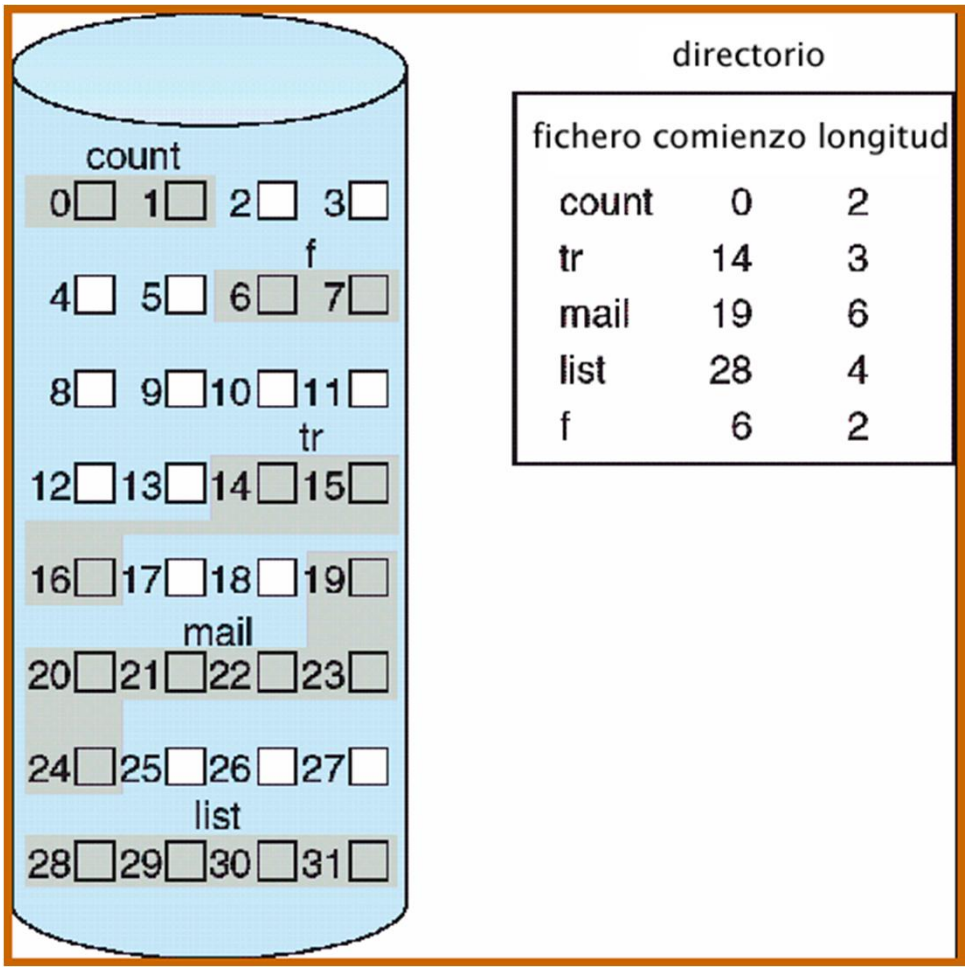
Métodos de asignación de espacio

- OBJETIVO: asignar espacio a ficheros de modo que el espacio en disco se aproveche de forma eficaz y se pueda acceder rápidamente a los archivos

  - *Asignación Contigua*
  - *Asignación Enlazada*
  - *Asignación Indexada*

Asignación contigua

- Cada fichero ocupa un conjunto de bloques contiguos en el disco (optimiza movimiento de las cabezas del disco)
- Entrada de directorio para cada fichero
  - Dirección del bloque inicial
  - Longitud del área asignada al archivo.



Asignación contigua

- Permite manejar acceso tanto secuencial como directo
- Dificultades
  - Encontrar espacio para la creación de un fichero
    - Algoritmos mas utilizados
      - Primer ajuste (*First Fit*)
      - Mejor ajuste (*Best Fit*)
      - Desde un punto de vista de aprovechamiento del espacio no existen diferencias pero el primero suele ser mas rápido
    - Problema ambos algoritmos: *Fragmentación Externa*
      - Solución: Compactación (pero es una solución costosa)

Asignación contigua

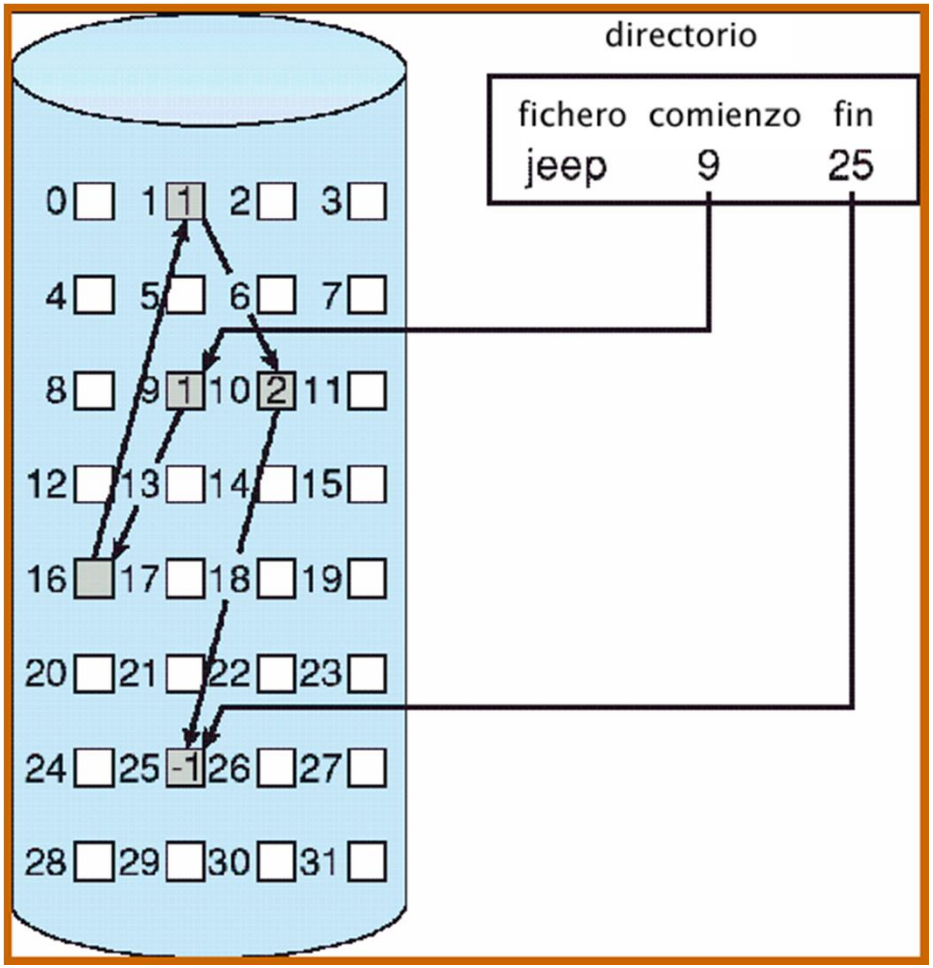
- Otro problema importante:
  - Determinar cuanto espacio se necesita para un fichero
  - En unas ocasiones es sencillo (cuando el archivo es copia de otro)
  - Muy complicado
  - Incluso cuando se conoce la cantidad total de espacio requerido, la pre-asignación puede ser ineficiente
    - Crecimiento lento (fragmentación interna)

Asignación contigua con extensiones (*extents*)

- Para evitar estas desventajas algunos SO permiten fragmentar el archivo (NTFS, XFS…):
  - Inicialmente se asigna un trozo contiguo de espacio
  - Cuando se requiere más espacio, se añade otro trozo de espacio contiguo → *extensiones*
  - Ubicación de los bloques de un archivo: bloque inicial, nº bloques, enlace al primer bloque de la siguiente extensión
  - Sigue existiendo fragmentación interna (si las extensiones son grandes) y externa (como consecuencia de la asignación y liberación de extensiones de diferentes tamaños)

Asignación enlazada

- Cada fichero es una lista enlazada de bloques de disco
- Entrada de directorio contiene:
  - Puntero al primer y último bloque del archivo



Asignación enlazada: Pros y Contras

- Se solucionan los problemas de la asignación contigua
  - No se produce fragmentación externa
  - No es necesario declarar de antemano el tamaño del archivo
- Desventajas:
  - Solo eficiente para archivos de acceso secuencial
  - Espacio que ocupan los punteros (un archivo requerirá algo mas de espacio del que requeriría en otro caso)

Asignación enlazada por *clusters*

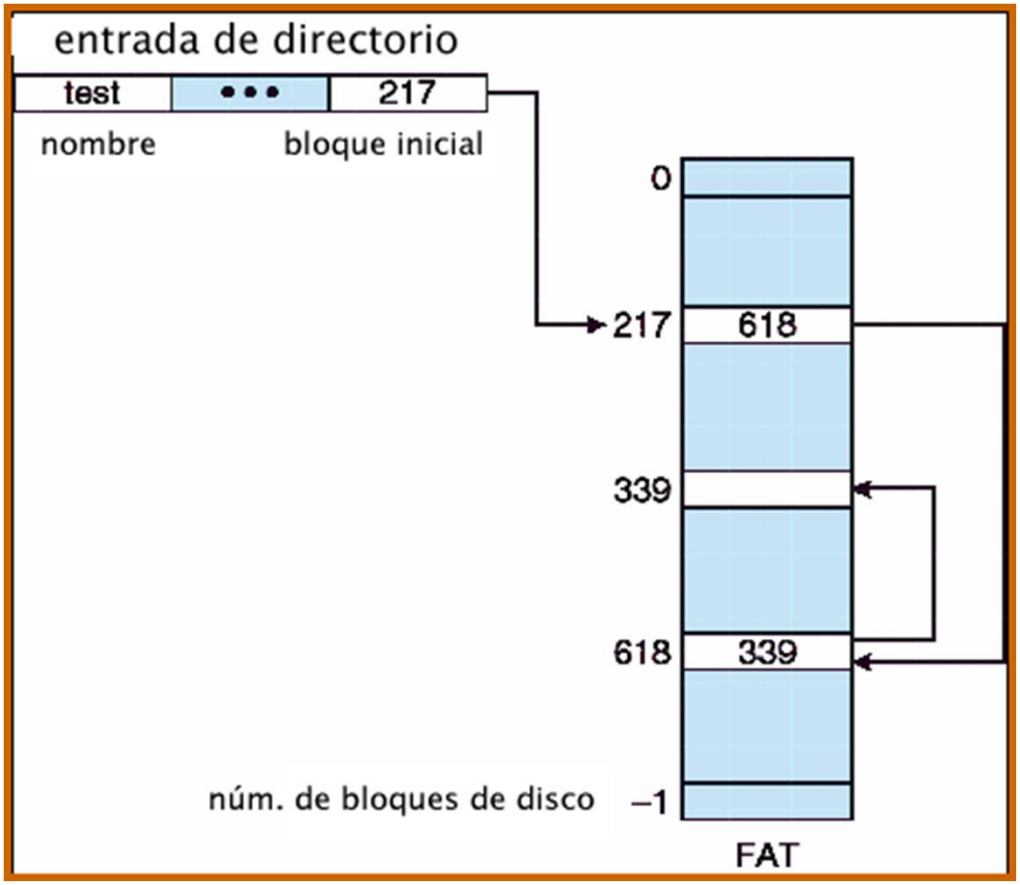
- Solución usual:
  - Agrupar los bloques en *grupos o clusters*
  - Esto permite:
    - Correspondencia entre bloques lógicos y físicos siga siendo sencilla
    - Mejora el rendimiento del disco (menos búsqueda de disco)
    - Reduce el espacio necesario para la asignación de bloques y la administración de la lista de espacio libre
  - Costo de esta estrategia:
    - Aumento de la fragmentación interna: se desperdicia mas espacio cuando un cluster está parcialmente lleno que cuando lo esta un bloque

Asignación enlazada

- Confiabilidad (¿Que sucedería si un puntero se perdiera o deteriorara ?)
  - Soluciones
    - Listas doblemente enlazadas
    - Almacenar el nombre del archivo y el numero de bloque relativo en cada bloque
  - Problema: gasta extra espacio

Asignación enlazada con FAT

- Variación del método de asignación enlazada
- FAT (Tabla de asignación de archivos, **File Allocation Table**)
- Se aparta una sección del disco al principio de cada partición para guardar en ella la tabla
- Empleado en MS-DOS y las tarjetas de memoria flash



Asignación Enlazada con FAT

- Si no está en caché puede dar lugar a un número significativo de movimientos de la cabeza del disco
- La cabeza debe moverse al principio de la partición para leer la FAT y encontrar el bloque en cuestión y luego moverse a la posición del bloque en sí.
- En el peor caso ocurrirán ambos movimientos para cada uno de los bloques
- Beneficio: mejora el tiempo de acceso aleatorio ya que se puede encontrar la posición de cualquier bloque leyendo la información en la FAT.

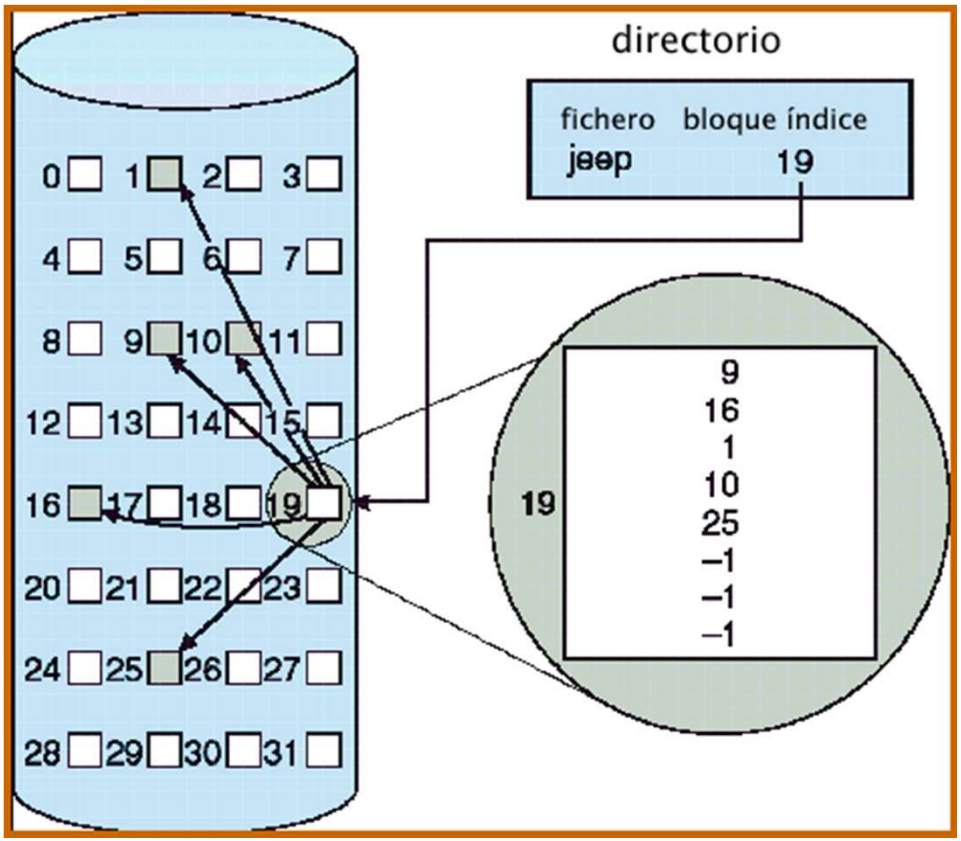
Asignación indexada

- La asignación enlazada resolvía los problemas de la asignación contigua
  - fragmentación externa y declaración anticipada del tamaño de los archivos

- Problema: si no se usa FAT no se puede implementar un acceso directo EFICIENTE (punteros dispersos junto con los bloques)

Asignación indexada

- SOLUCION:
  Reunir todos los punteros en el mismo lugar → **bloque de índices**
- La i-ésima entrada del bloque índice apunta al i-ésimo bloque del archivo



Asignación indexada: Pros y Contras

- Soporta acceso directo sin sufrir fragmentación externa
- Desventaja:
  - Desperdicia espacio (peor que en el caso de la asignación encadenada, p. Ej. Fichero que ocupe 1 o 2 bloques)
- ¿ Qué tamaño debería tener el bloque de índices?

Asignación indexada: tamaño del bloque de índices

- Cada archivo debe tener un bloque índice → conviene que el bloque sea lo mas pequeño posible (normalmente ocupa un bloque de disco)
- Pero, si es demasiado pequeño no podrá tener suficientes punteros para un archivo grande. Soluciones:
  - Esquema Enlazado
  - Índice multinivel
  - Esquema combinado (BSD UNIX)