SISTEMAS DE FICHEROS

Universidad San Pablo-CEU
Escuela Politécnica Superior
Rodrigo García Carmona

OBJETIVOS

- Entender cómo funcionan los **discos magnéticos**, sus particularidades, y cómo solucionar los problemas que presentan, en particular mediante el uso de **algoritmos de planificación**, **esquemas RAID** y **optimización del tiempo de rotación**.
- Comprender el **concepto de fichero**, sus características, qué operaciones se pueden realizar sobre ellos, y cómo están implementados, prestando especial atención a su asignación en disco.
- Entender la estructuración de ficheros en forma de **directorios**, el *path name*, y las operaciones que se pueden llevar a cabo sobre los directorios. Conocer cómo están implementados.
- Ser consciente del problema de la **consistencia** en la información almacenada en disco, y aprender cómo asegurarla.

CONTENIDOS

- Discos Magnéticos
- Ficheros
- Directorios
- Consistencia

Bibliografía

- W. Stallings:
  Sistemas Operativos.
  - Capítulo 11, 12.
- A.S. Tanenbaum:
  Modern Operating Systems.
  - Capítulo 4, 5.

DISCOS MAGNÉTICOS

ESTRUCTURA FÍSICA DE UN DISCO MAGNÉTICO

- Disco magnético:
  - Floppy (Diskettes).
  - Disco Duro.
- Un disco magnético está compuesto de uno o más **platos**.
- Un plato puede tener una o dos **caras**.
- Cada cara tiene una **cabeza** que se encarga de leerla.
- Cada cara está dividida en **pistas** circulares.
  - Las pistas siguen la misma geometría para todos los platos.
- Al conjunto de la misma pista en todos los platos se le llama **cilindro**.
- Cada pista está dividida en **sectores** de un tamaño fijo.
  - El número de sectores por pista puede ser constante para todo el disco o depende según unas divisiones denominadas **zonas**.

TIEMPO DE ACCESO EN DISCOS MAGNÉTICOS

- **Tiempo total de acceso:** Búsqueda + Rotación + Transferencia.
  - **Tiempo de búsqueda:**
    - Mover la cabeza hasta la pista.
    - Muy variable: 0-15 milisegundos.
  - **Tiempo de rotación:**
    - Esperar a que el sector se sitúe bajo la cabeza.
    - Variable: 0-5 milisegundos.
  - **Tiempo de transferencia:**
    - Leer el sector.
    - Fijo: 15 microsegundos.
- Son accesos lentos. Es necesario optimizar el acceso a disco:
  - Algoritmos de planificación.
  - Esquemas RAID.
  - Optimización del tiempo de rotación.

ALGORITMOS DE PLANIFICACIÓN (I)

- El objetivo es minimizar el **tiempo de búsqueda**.
- Se aprovecha la geometría del disco.
  - El tiempo de búsqueda se reduce si accedemos a **cilindros cercanos**.
- **FCFS (First-Come First-Served):**
  - Más sencillo de los algoritmos.
  - Se atiende a las peticiones en orden de entrada.
  - No se optimiza el tiempo de búsqueda.
  - Es el más justo.
    - No se puede producir inanición.

ALGORITMOS DE PLANIFICACIÓN (II)

- Las peticiones pendientes se almacenan en una lista.
- **SSF (Shortest Seek First):**
  - Se atiende a la petición cuya pista se encuentre más cerca.
  - Mejora mucho el tiempo de búsqueda.
  - Puede producirse inanición.
    - Tienen preferencia las peticiones cerca de la mitad del disco.
- **Ascensor:**
  - Se intenta que la cabeza se mueva en un solo sentido…
  - …hasta llegar al límite, momento en que cambia de sentido.
  - Casi tan bueno como SSF para el tiempo de búsqueda.
  - Espera máxima: dos veces el total de cilindros.
  - **Variación:** Movimiento en un único sentido.
    - Cuando se llega al cilindro más exterior se atiende la petición del cilindro más interior.
    - Espera máxima: el total de cilindros.

OPTIMIZACIÓN DEL TIEMPO DE ROTACIÓN

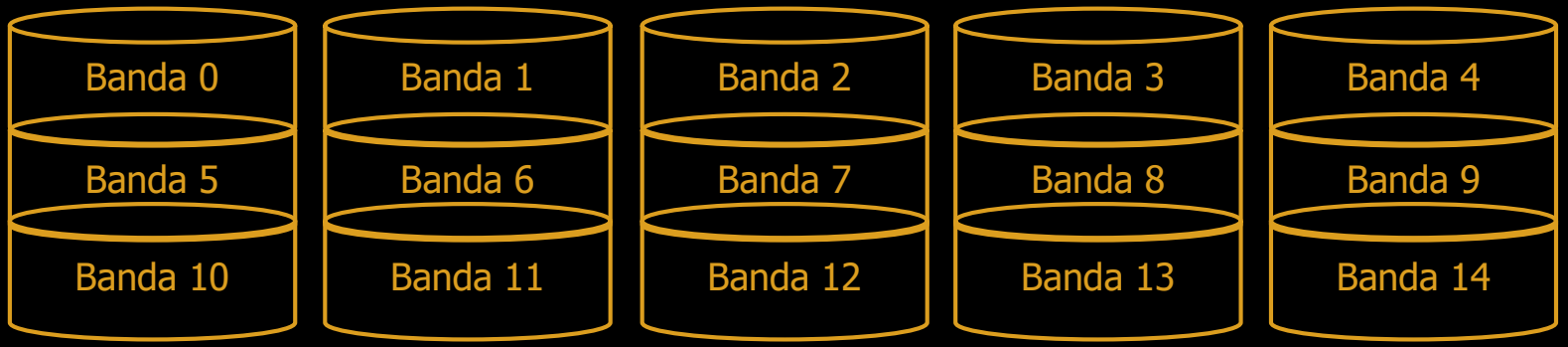
- También se puede optimizar el tiempo de rotación.
- Diferentes estrategias:
  - Formateo escalonado:
    - Se marca como primer sector de cada pista de manera escalonada.
    - Se aprovecha la relación entre tiempo de rotación y búsqueda.
    - Apto para peticiones grandes.
  - Planificación:
    - Se encolan las peticiones para la misma pista en orden.
  - Cache de disco:
    - Se almacenan varios sectores seguidos y se almacenan.
    - Transparente para el Sistema Operativo.
    - Cache de lectura y cache de escritura.

RAID

- La velocidad de los discos duros no avanza a la misma velocidad que el resto de componentes informáticos.
  - Problema mecánico.
  - No obedece la ley de Moore.
- Para hacer frente a este problema se propone el uso de paralelismo.
- **RAID:** Redundant Array of Inexpensive/Independent Disks.
  - Varios discos duros conectados a la misma controladora.
  - A nivel lógico funcionan como un único disco.
  - Además añaden tolerancia a fallos.
  - 6 esquemas de RAID.
- **Frente a SLED:** Single Large Expensive Disk.

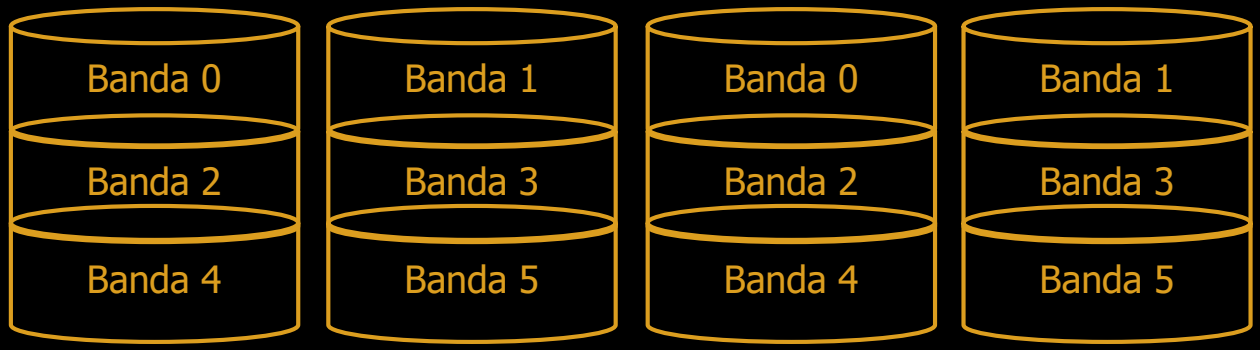
RAID NIVEL 0

- Dividido en bandas (*strips*) de *k* sectores cada una.
- Distribución física: una banda por disco.
- **Entrada/Salida en paralelo:**
  - Gran mejora de rendimiento.
  - Adecuado para peticiones grandes.
  - Soporta peticiones en paralelo.
- **Menor fiabilidad:**
  - El tiempo medio de fallo se divide por el número de discos.



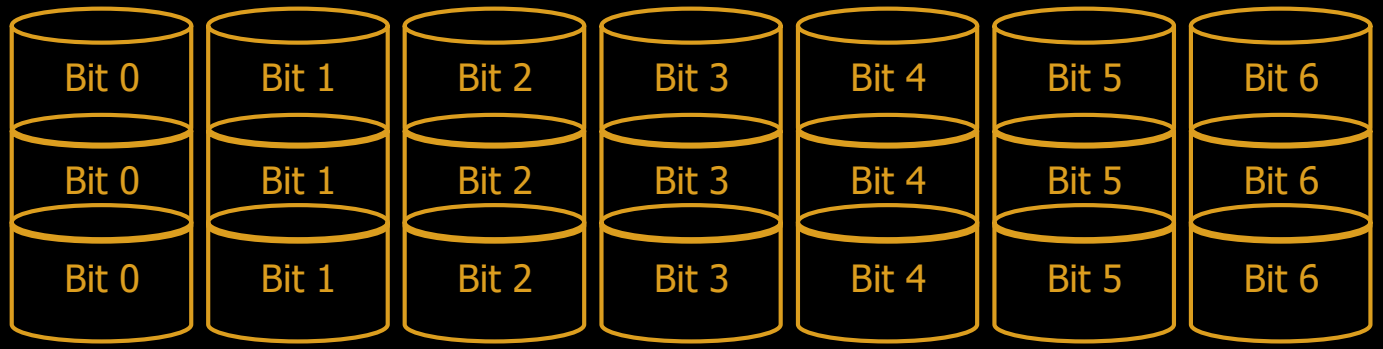
RAID NIVEL 1

- Distribución física: bandas duplicadas en un segundo disco.
- **Lectura:** Se puede leer de cualquiera de las dos copias.
  - Hasta dos veces mejor.
  - Soporta peticiones en paralelo.
- **Escritura:** Se escribe en las dos copias a la vez.
  - Igual de eficiente que sin RAID.
- **Tolerancia a fallos:** Aguanta el fallo de un disco duro.
- **Espacio en disco:** Se usa sólo la mitad.



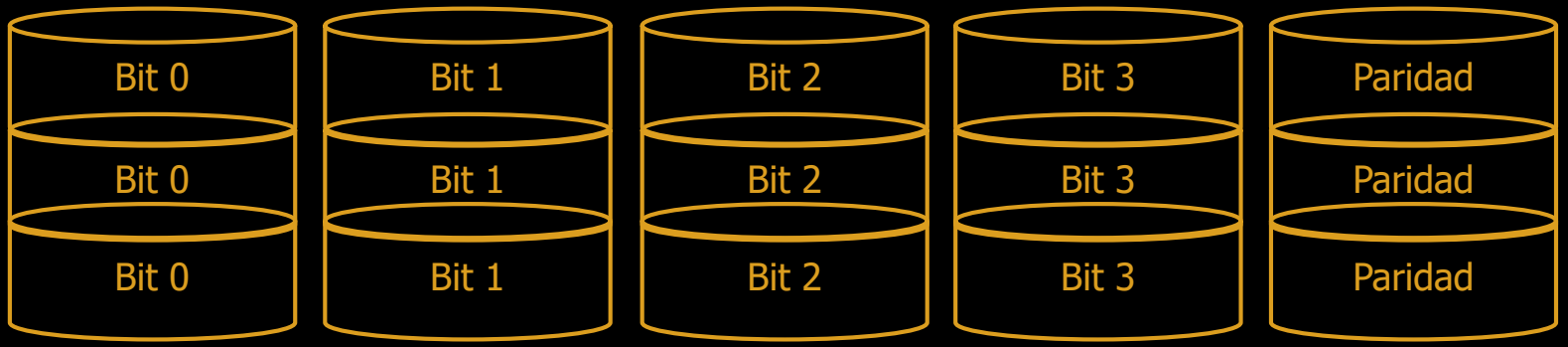
RAID NIVEL 2

- Usa código Hamming para corrección de errores:
  - Ejemplo: 4 bits de información (0, 3, 5, y 6), 3 bits corrección (1,2, y 4).
  - Ejemplo: 32 bits de información, 6 bits de paridad.
- Distribución física: una bit en cada disco, alternativamente.
- **Entrada/Salida en paralelo:** Gran mejora de rendimiento.
  - Pero no puede manejar peticiones en paralelo.
- **Tolerancia a fallos:** Si un disco falla se corrige el error.
- **Dificultad técnica:** Sincronización de discos.
- **Tamaño:** Sólo tiene sentido con cantidades de discos muy grandes.



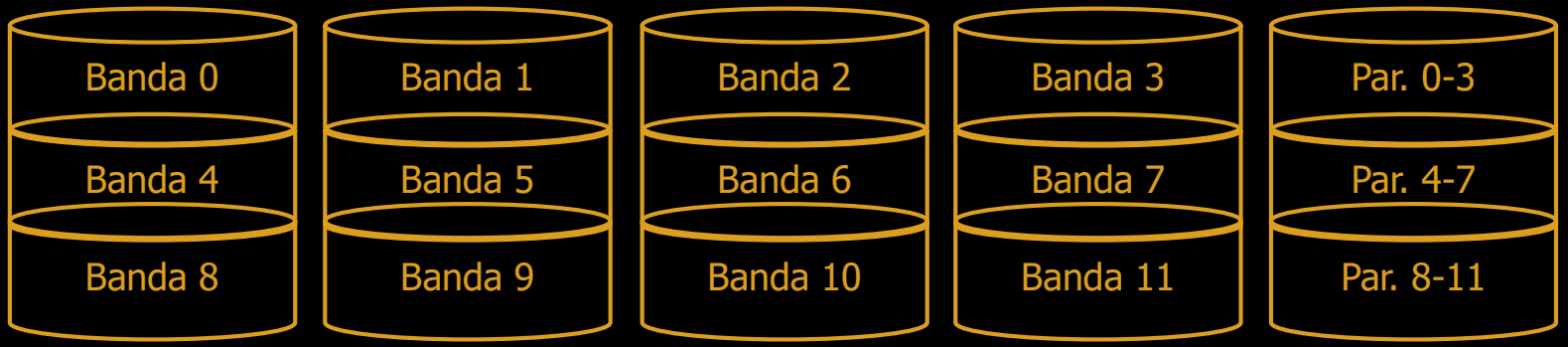
RAID NIVEL 3

- Versión simplificada de RAID 2.
- Un único bit de paridad:
  - Permite detectar, pero no corregir, un error.
  - Pero como sabemos **dónde** se ha producir el error, podemos corregirlo.
- **Entrada/Salida en paralelo:** Gran mejora de rendimiento.
  - Pero no puede manejar peticiones en paralelo.
- **Tolerancia a fallos:** Si un disco falla se corrige el error.
- **Dificultad técnica:** Sincronización de discos.



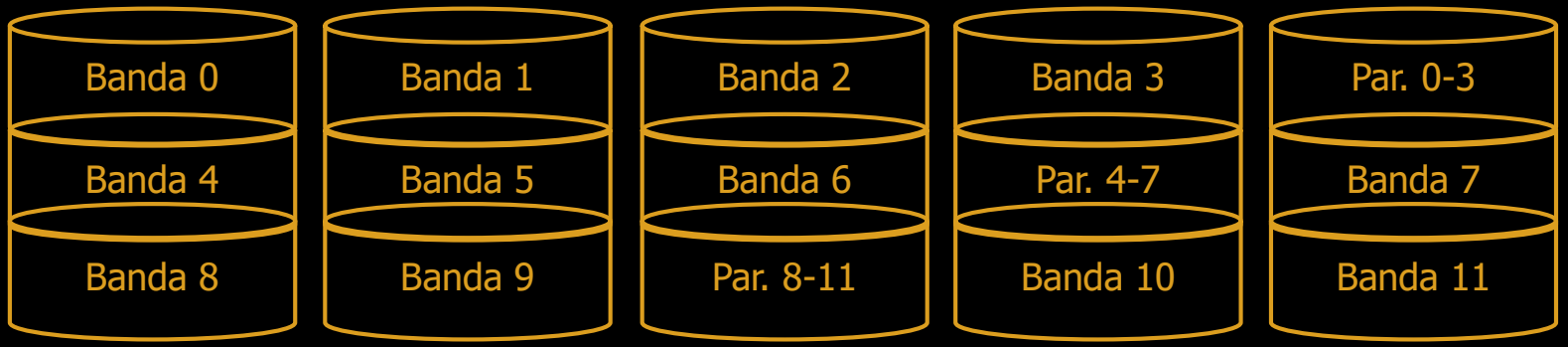
RAID NIVEL 4

- Similar a RAID 3 pero con bandas en lugar de bits.
- Un disco almacena la banda de paridad.
- No es necesario sincronizar los discos
- **Tolerancia a fallos:** Si un disco falla se corrige el error.
- **Soporta peticiones en paralelo:**
  - Cada escritura implica otra en el disco de paridad.
  - Es complicado recalcular la paridad en estas circunstancias.
  - El disco de paridad se convierte en un cuello de botella.



RAID NIVEL 5

- Similar a RAID 4.
- La banda de paridad se almacena para grupo de bandas en un disco distinto.
- **Se elimina el cuello de botella de RAID 4:**
  - Ahora no siempre el mismo disco actúa como paridad.
  - Conserva el resto de ventajas.
- **Problema:** Es más complejo reconstruir un disco si se produce un fallo.



FICHEROS

FICHEROS

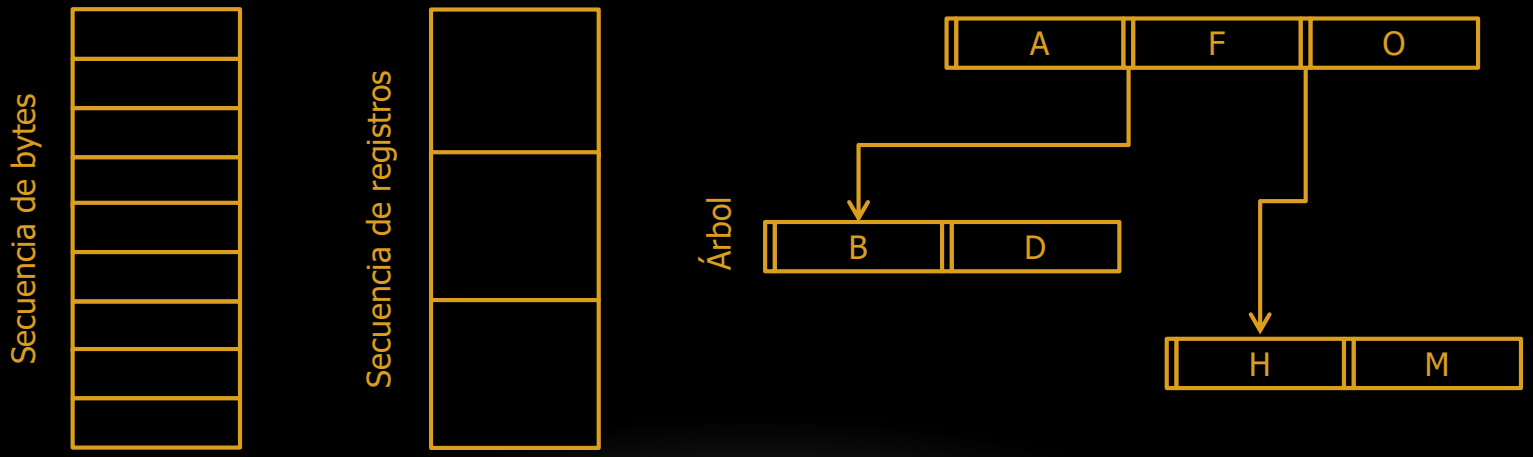
- **Ficheros:** unidades **lógicas** de almacenamiento de información.
  - Compuestos de uno o más bloques.
  - Un bloque puede ser, o no, equivalente a un sector en un disco magnético.
- Los ficheros almacenan información **persistente**.
  - Deben existir más allá de la vida de un proceso.
- Están gestionados por un componente del sistema operativo llamado **sistema de ficheros**.
  - Abstrae del manejo del disco físico.
  - Determina las características de los ficheros.
  - Define las operaciones que se pueden realizar sobre los ficheros.
  - Establece quién puede realizar dichas operaciones.

NOMBRES DE FICHEROS

- **Nombre:** “título” del fichero.
  - Útil para dar una idea de su función y/o características.
  - **No es** un identificador único. Esa función la cumple el inode.
- El formato del nombre depende del sistema de ficheros:
  - Habitualmente compuesto de base y extensión, separadas por un “.”.
  - La extensión sirve para indicar de qué tipo de fichero se trata.
    - El sistema operativo puede usarlo para determinar con qué programa se debe acceder al fichero.
  - Suele tener un tamaño máximo.
  - Tiene caracteres prohibidos.

ESTRUCTURA INTERNA DE FICHEROS

- Los ficheros pueden estar estructurados como:
  - **Una secuencia de bytes:**
    Como la memoria principal. El más común.
  - **Una secuencia de registros:**
    Cada registro con su estructura propia. Usado en el pasado.
  - **Un árbol:**
    Pares llave-valor. Las llaves están ordenadas. Ideal para búsquedas.



TIPOS DE FICHEROS

- La mayoría de sistemas operativos distinguen varios tipos de ficheros:
  - **Archivos normales (*regular*):** Contienen información. Dos subtipos:
    - **ASCII:** Información en forma de líneas de texto.
    - **Binarios:** Código ejecutable o información no textual.
  - **Directorios:** Los veremos más adelante.
  - **Archivos especiales:** Para modelar Entrada/Salida. Dos subtipos:
    - **De flujo de caracteres.**
    - **De bloque.**
- Los ficheros **binarios** contienen información muy variada, que puede ser o no ejecutable.
  - Tienen una estructura interna estándar según el sistema operativo.
  - Si esta estructura incluye un **número mágico**, esto indica que es un fichero ejecutable.
  - El resto de la estructura indica cómo debe cargarse o manejarse el programa.

CARACTERÍSTICAS DE FICHEROS

- Los ficheros pueden dividirse en dos tipos:
  - **Secuencial:** Deben ser accedidos en orden secuencial. Para cintas.
  - **Aleatorio:** Pueden ser accedidos en el orden que se desee.
- Las características de cada fichero se almacenan en sus **metadatos** o **atributos**:
  - Protección: Quién puede acceder al fichero y de qué forma.
  - Creador y/o dueño.
  - Grupo: Conjunto de usuarios que pueden manejar el fichero.
  - Tamaños actual y máximo.
  - Tiempos de creación, último acceso y última escritura.
  - **Flags:** Indican de forma binaria ciertas características:
    - Oculto.
    - De sistema.
    - ASCII/Binario.
    - Temporal.
    - Sólo lectura.

OPERACIONES SOBRE FICHEROS

- Los ficheros admiten una serie de operaciones sobre ellos. Las más comunes son:
  - **Creación:** El archivo se crea vacío.
  - **Borrado:** Comprueba que no haya más accesos simultáneos.
  - **Apertura:** Necesario para poder operar con el archivo. Lectura/Escritura.
  - **Cierre:** Cuando se ha acabado de operar con el archivo.
  - **Lectura:** A partir de la posición actual.
  - **Búsqueda (*seek*):** Cambio de la posición actual.
  - **Escritura:** Se sobrescribe información.
  - **Adjuntar (*append*):** Se añade información sin sobrescribir.
  - **Leer/modificar atributos.**
  - **Renombrar:** A veces se sustituye por copia y borrado.
- Se implementan mediante llamadas a sistema.

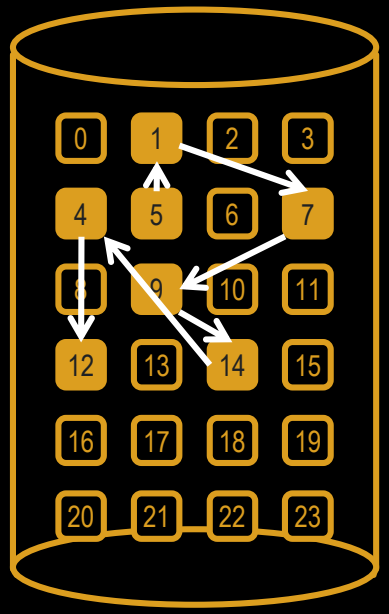
ASIGNACIÓN DE FICHEROS: ASIGNACIÓN CONTINUA

- Traducción de la información lógica de los ficheros en información física en el disco. **Mapeo de bloques a sectores.** Varios esquemas:
- **Asignación contigua:**
  - Los archivos ocupan una serie de sectores contiguos en el disco.
  - **Ventajas:**
    - Simple de implementar.
    - Alta velocidad de lectura y escritura.
  - **Desventajas:**
    - Se produce fragmentación. Es necesario desfragmentar.
    - Es necesario saber con antelación el tamaño máximo del fichero.
  - Se usaba con unidades de cinta. Ha vuelto a la vida gracias a los discos ópticos.

Inicio: 5
Longitud: 6

ASIGNACIÓN DE FICHEROS: LISTA ENLAZADA

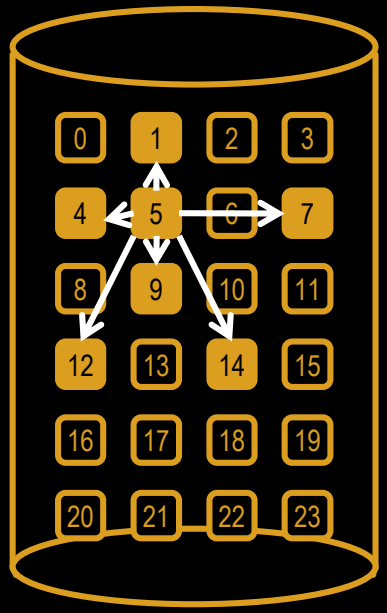
- **Lista enlazada:**
  - Cada sector contiene un enlace al siguiente sector.
  - **Ventajas:**
    - No se produce fragmentación.
    - El archivo puede crecer fácilmente.
  - **Desventajas:**
    - El acceso es más lento.
    - Los sectores no contienen datos potencia de 2. Se invierte espacio en el enlace.
  - Se pueden eliminar esta última desventaja poniendo los enlaces entre sectores en memoria: FAT (File Allocation Table)



Inicio: 5
Sectores: 5,1,7,9,14,4,12

ASIGNACIÓN DE FICHEROS: I-NODES

- **I-nodes (index nodes):**
  - Cada fichero tiene una estructura de datos (i-node) con una lista de las características y ubicación de cada sector.
  - **Ventajas:**
    - No se produce fragmentación.
    - El acceso es más rápido que con una lista enlazada.
    - Los sectores contienen datos con potencia de 2.
  - **Desventajas:**
    - El acceso es más lento que con asignación contigua.
    - Tamaño máximo de fichero limitado por i-node.
  - Se puede eliminar esta última desventaja usando i-nodes multinivel.



Índice: 5
Sectores: 1,7,9,14,4,12

DIRECTORIOS

DIRECTORIOS

- Los **directorios** son el mecanismo que utiliza el sistema de ficheros para organizar los archivos.
- Existen dos estructuras de directorios:
  - **Sistemas de directorio único:**
    - Sólo hay un directorio, el raíz.
    - No pueden repetirse nombres de ficheros.
    - Es sencillo de implementar.
    - Es fácil localizar archivos.
  - **Sistemas de directorios jerárquicos:**
    - Hay varios directorios, organizados en forma de árbol.
    - Permite agrupar ficheros por temática, usuario, función o acceso.
    - Pueden repetirse nombres de ficheros.
    - Usado en la gran mayoría de sistemas modernos.

PATH NAMES

- Los ***path names***, o nombres de camino, indican en qué posición se encuentra un fichero dentro de una estructura de directorios.
- Están compuestos de los nombres de los directorios que hay que recorrer, desde la raíz del árbol, para llegar al que contiene el fichero.
- Estos nombres se separan mediante un carácter específico del sistema:
  - "/" en UNIX.
  - "\" en Windows.
- El directorio raíz indica mediante el separador o un grupo de caracteres especial.
- Los *path names* pueden ser:
  - Relativos: usan como base la posición "actual" dentro del sistema de ficheros.
    - Ejemplo: *./SO2/notas/suspensos.txt*
  - Absolutos: usan como base la raíz del sistema de ficheros.
    - Ejemplo: */home/rodrigo/SO2/notas/suspensos.txt*
- Caracteres especiales:
  - "." : Directorio actual.
  - "..": Directorio padre del actual.

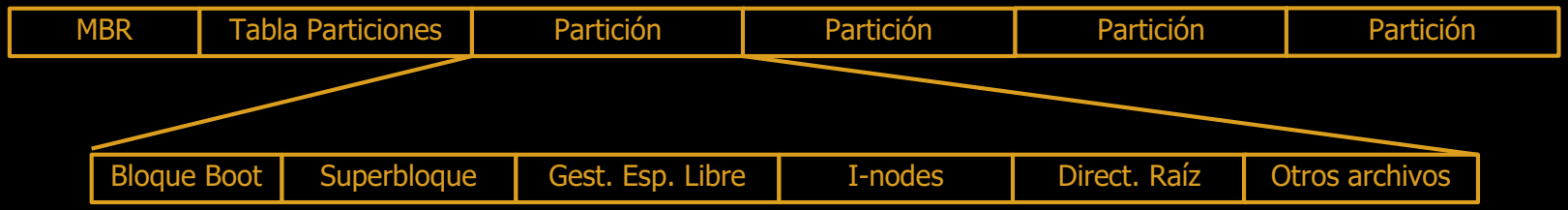
OPERACIONES SOBRE DIRECTORIOS

- Los directorios admiten, por lo general, las siguientes operaciones:
  - **Creación:** Incluye automáticamente los archivos "." y "..".
  - **Borrado:** Sólo si está vacío.
  - **Apertura:** Necesario para poder operar.
  - **Cierre:** Cuando se ha acabado de operar.
  - **Lectura:** A partir de la posición actual. Devuelve la siguiente entrada.
  - **Renombrar:** A veces se sustituye por copia y borrado.
  - **Enlazar:** Para que una archivo aparezca en varios directorios.
  - **Desenlazar:** Romper un enlace.
- Los enlaces creados de esta manera son los ***hard links*** (enlaces duros o reales), diferentes de los ***symbolic links*** (enlaces simbólicos), en los que se crea un nuevo archivo cuya única función es redirigir al que ya existía.

IMPLEMENTACIÓN DE DIRECTORIOS

- Los directorios **se implementan como ficheros.**
- Los atributos o metadatos de los ficheros **pueden almacenarse en:**
  - El directorio que los contiene.
  - El inode del fichero, al que se hace referencia en el directorio.
- En ambos casos las entradas que almacenan esta información pueden ser:
  - De longitud fija:
    - Tamaño de nombres limitado.
    - Espacio desperdiciado.
  - De longitud variable:
    - Puede producirse fragmentación dentro de la lista de entradas.
- **La búsqueda en directorios puede acelerarse mediante:**
  - Tablas *hash* con los nombres de ficheros.
  - Caches de búsqueda en directorios.

IMPLEMENTACIÓN DEL SISTEMA DE FICHEROS



- **MBR (*Main Boot Record*):** Sector de Arranque. Se lee al arrancar el sistema. Redirige al Bloque Boot de una partición.
- **Tabla de particiones:** Información sobre las particiones del disco.
- **Bloque Boot:** Contiene la información de arranque del sistema operativo.
- **Superbloque:** Información sobre el sistema de ficheros.

CONSISTENCIA

JOURNALING (I)

- **Concepto básico:** Llevar un registro de lo que el sistema va a hacer antes de hacerlo.
  - Si el sistema falla se consulta el registro y se lleva a cabo la tarea pediente.
- Usado en sistemas de ficheros actuales: Ext4, ReiserFS, NTFS…
- **Ejemplo:** Borrado de un fichero.
  1. Eliminar el fichero del directorio.
  2. Añadir el inode a la lista de inodes libres.
  3. Añadir los bloques de disco a la lista de bloques libres.
- El orden no es relevante, pero si se ejecutan sólo algunos de estos pasos se producen problemas:
  - Si 1 y 2 pero no 3: Se pierden bloques permanentemente.
  - Si 1 pero no 2 ni 3: Se pierden i-nodes y bloques permanentemente.
  - Si 2 pero no 1 ni 3: Dos ficheros con el mismo i-node.
  - Si 3 pero no 1 ni 2: Dos ficheros con los mismos bloques.
  - …

JOURNALING (II)

- Un sistema de ficheros con Journaling:
  - Escribe en el registro la operación a realizar.
  - Realiza todos los pasos de la operación.
  - Marca en el registro la operación como realizada.
- Las operaciones deben ser **idempotentes**:
  - Pueden repetirse tantas veces como se desee sin que cambie el resultado.
- **Operaciones idempotentes:**
  - Marca el i-node “x” como libre.
- **Operaciones no idempotentes**:
  - Añade el bloques “y” a la lista de bloques libres.
  - **Puede hacerse idempotente:**
    - Busca en la lista de bloques libres el bloque “y” y, si no está presente, añádelo.
    - **Son más costosas.**
- Pueden agruparse varias operaciones para realizar el conjunto atómicamente.

ALMACENAMIENTO ESTABLE (I)

- Es necesario proteger el sistema de ficheros frente a errores:
  - Errores de escritura.
  - Caídas durante el proceso de escritura.
  - Fallos de sectores.
- **Almacenamiento estable:** cuando se puede asegurar que, una vez dada una orden de escritura:
  - O se lleva a cabo.
  - O se deja la información original.
- Se puede implementar a nivel de *software*, pero precisa de la posibilidad de detectar errores. Para ello se destinan espacios ECC (de control de errores) para cada bloque durante el formateo.
- Usa dos discos que almacenan la misma información:
  - Sólo habría error si fallaran a la vez el mismo sector en ambos discos.
- Si falla un sector se marca como malo y se reubica la información a otro.
  - Todos los discos cuentan con sectores de reserva.

ALMACENAMIENTO ESTABLE (II)

- Se definen tres operaciones:
  - **Escritura estable:**
    Escritura en disco 1, comprobación. → Escritura en disco 2, comprobación.
  - **Lectura estable:**
    Lectura en disco 1, comprobación. Si falla, lectura en disco 2, comprobación.
  - **Recuperación frente a errores:**
    Se comparan discos tras una caída. Si distintos se copia del disco 1 al 2.
- Ejemplo de funcionamiento. 5 momentos posibles para error: