Sistema de ficheros

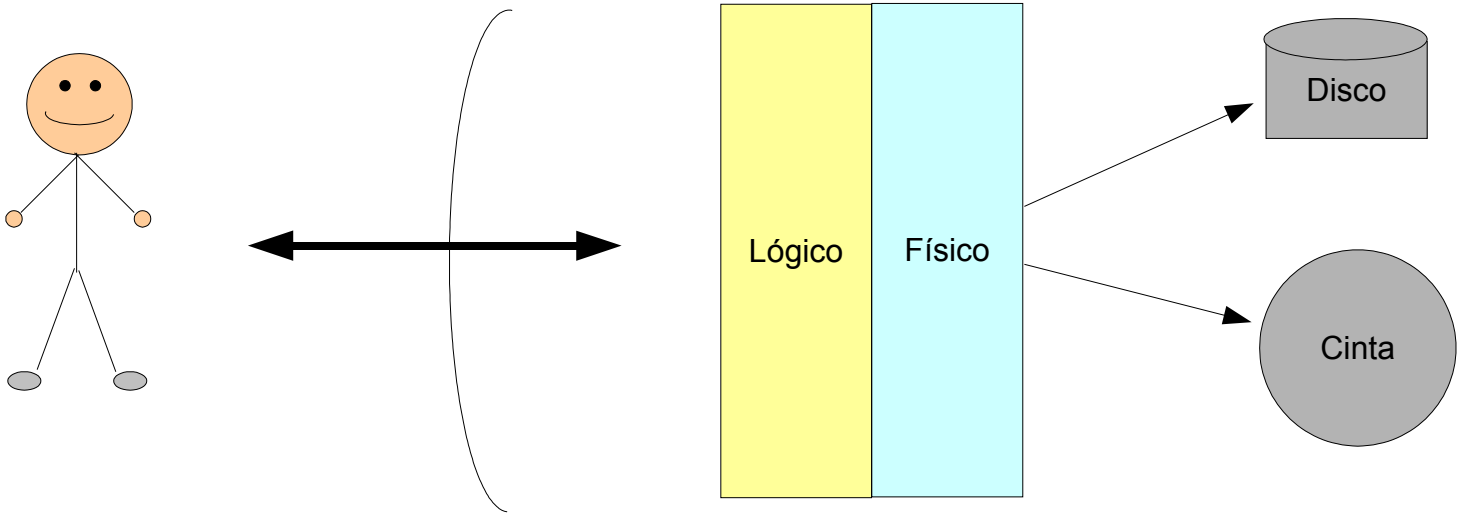
Sistemas Operativos (SO)
Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Contenido

- Definiciones previas
- Organización de los directorios
- Estructura interna
  - Estructuras de datos en el disco
  - Estructuras de datos en memoria
- Planificación
- Ejemplos
  - Onion
  - UNIX
  - Windows NT

Definiciones previas

- Sistema de ficheros:
  - Conjunto de datos + algoritmos de gestión sobre los dispositivos de almacenamiento
  - Uno de los aspectos más visibles del S.O.
  - Permite un acceso a los datos independiente del dispositivo

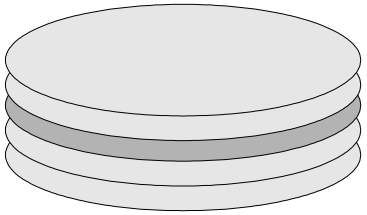
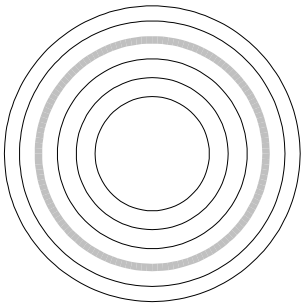
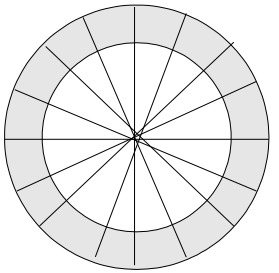


Deficiones previas (II)

- Responsabilidades del SF
  - Asignar espacio libre a los ficheros
  - Liberar el espacio eliminado de los ficheros
  - Encontrar/Almacenar los datos de los ficheros

Deficiones previas (III)

- Disco
  - Sistema de almacenamiento masivo
  - Dividido en caras/pistas/sectores

  - Tiempo total =

    Tiempo de posicionamiento/busqueda: posicionarse sobre la pista correcta

    + Tiempo de espera: esperar a que pase el sector deseado

    + Tiempo de transferencia: leer los datos

Definiciones previas (IV)

- Archivo o Fichero
  - Conjunto de información relacionada organizada como una secuencia de bytes
  - Se clasifican según criterios del usuario
  - Estructura definida por su uso
- El S.O. puede conocer el tipo (estructura) de los ficheros
  - Más costoso
  - Menos flexible
  - Más prestaciones
    - Impedir ciertos tipos de operaciones inválidas
    - Realizar automáticamente ciertas operaciones

Definiciones previas (V)

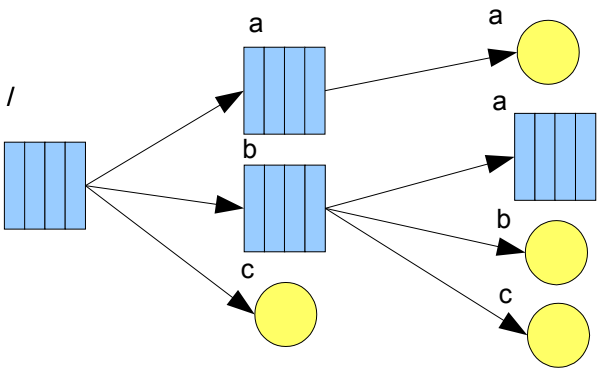
- Directorio
  - Estructura de datos que permite organizar los ficheros
  - Ofrece una visión jerárquica del sistema de ficheros
  - Los ficheros tienen 2 tipos de nombres
    - @ lógica: Nombre dentro del directorio (“fichero.txt”)
    - @ física: Nombre dentro del dispositivo (cara/pista/sector)
  - Permite traducir de la @ lógica de un fichero a su @ física

Definiciones previas (VI)

- Enlace (*link*)
  - Relación entre una @lógica y una @física
  - Estas relaciones pueden ser N:1
    - Varias @lógicas para una misma @física
- 2 tipos de enlaces
  - Enlaces duros (*hard links*)
    - Sólo dentro de un mismo dispositivo
  - Enlaces blandos (*soft links*)
    - Pseudo-enlaces
    - Fichero especial que contiene la @lógica de otro fichero
    - Interpretados por el S.O para simular *hard links*

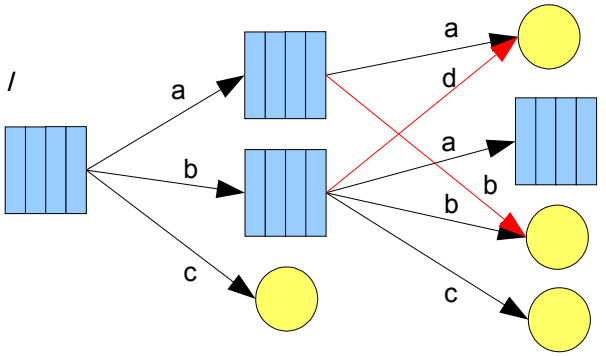
Organización de los directorios

- En árbol
  - Existe un directorio raíz
  - Cada directorio puede contener ficheros y/o directorios
    - El usuario puede realizar operaciones sobre sus ficheros/directorios
  - Cada fichero/directorio se puede referenciar de dos maneras
    - Nombre absoluto (único): Camino desde la raíz + nombre
    - Nombre relativo: Camino desde el directorio actual + nombre



Organización de los directorios (II)

- En grafo
  - Generalización de la estructura de arbol
    - Mediante el uso de *links*
  - Dos tipos
    - Acícliclos: el S.F. verifica que no se creen ciclos
    - Cíclicos: el S.F. ha de controlar los ciclos infinitos



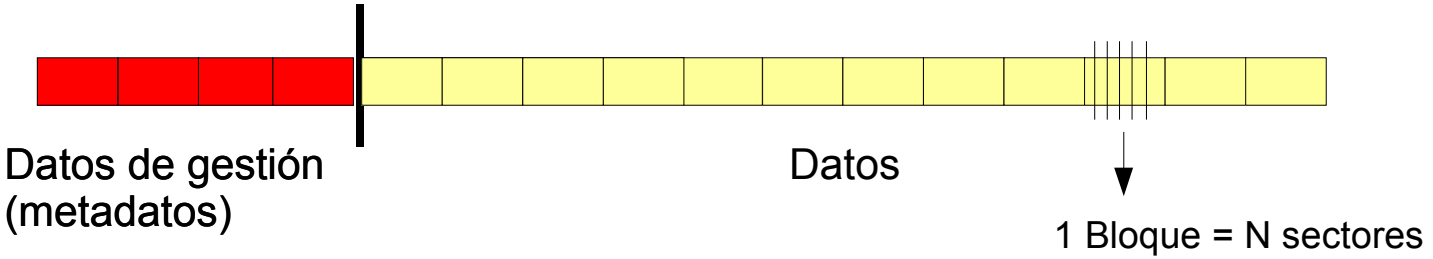
Organización de los directorios (III)

- Problemas de los directorios en grafos
  - Backups
    - No hacer copias del mismo fichero
  - Eliminación de un fichero
    - *Soft links*
      - El sistema no comprueba si hay *soft links* a un fichero
    - *Hard links*
      - Contar el número de referencias al fichero
      - Borrarlo cuando este llegue a cero

Estructura interna

- Un sistema de ficheros ha de definir
  - Tipos de ficheros
  - Atributos de los ficheros (protecciones, ...)
  - Conjunto de operaciones permitidas
- Estructuras
  - en disco (metadatos)
    - Gestión del espacio libre
    - Gestión del espacio ocupado
    - Organización de la información (directorios)
  - en memoria
    - Tabla de ficheros dinámica
    - Tabla de ficheros estática

Estructuras en disco



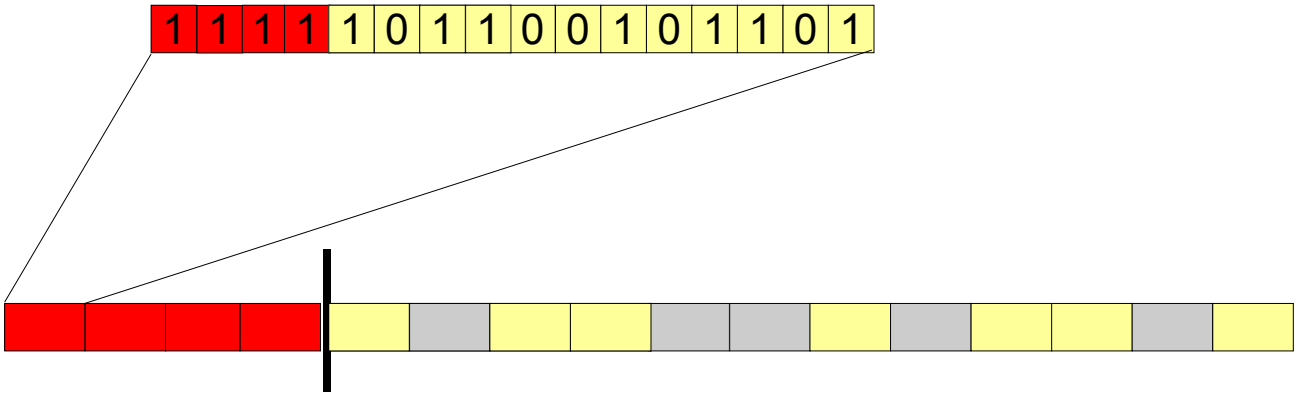
- Información organizada en bloques
  - Sector: unidad de transferencia (definida por el Hw)
  - Bloque: unidad de asignación (definido por el SO)
    - Compromiso en el tamaño de bloque
      - Eficiencia
      - Fragmentación

Gestión del espacio libre

- El S.F. tiene una estructura que gestiona el espacio libre del dispositivo
  - Mapa de bits
  - Bloques enlazados
  - Bloques agrupados
- Cuando se pide espacio
  - Se busca espacio en la estructura
  - Se marca como ocupado (eliminandolo de la lista)
- Reutilización del espacio liberado
  - Se añade nuevamente a la lista

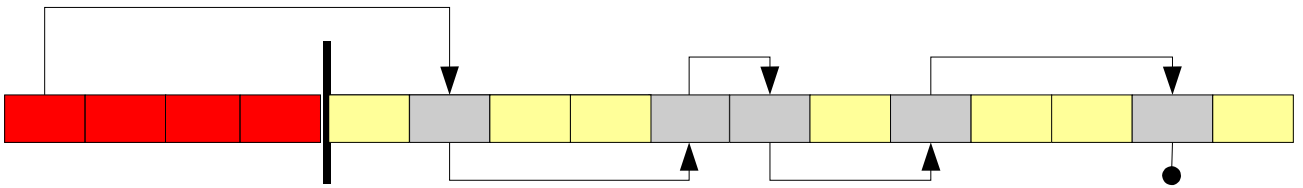
Gestión del espacio libre (II)

- Mapa de bits (*bitmap*)
  - Cada bloque tiene asignado un bit que indica si esta ocupado o libre
  - Simple y eficiente (si cabe en memoria)
    - Discos grandes puede no caber



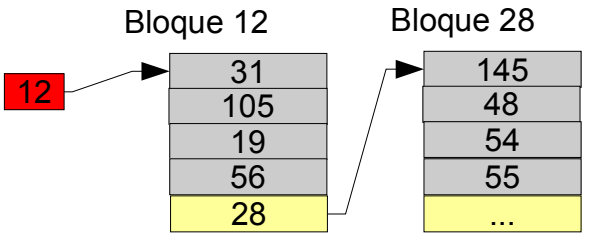
Gestión del espacio libre (III)

- Bloques encandenados
  - Apuntador al 1er bloque libre
  - En cada bloque libre hay un apuntador al siguiente bloque libre
  - Muy sencillo, pero ineficiente
    - Un acceso a disco por cada elemento de la lista
    - Intentar reducir el número de acceso agrupando apuntadores

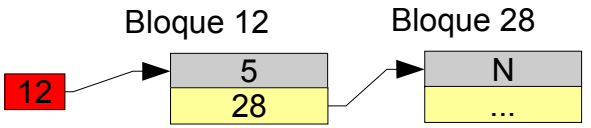


Gestión del espacio libre (IV)

- Bloques agrupados
  - En cada bloque libre guardamos N-1 direcciones de bloques libres
  - El último apuntador del bloque apunta al siguiente bloque con de direcciones



- Bloques agrupados consecutivos
  - Apuntamos al primer bloque de un grupo de bloques libres
  - En ese bloque se guarda
    - Número de bloques libres consecutivos
    - Dirección del siguiente grupo

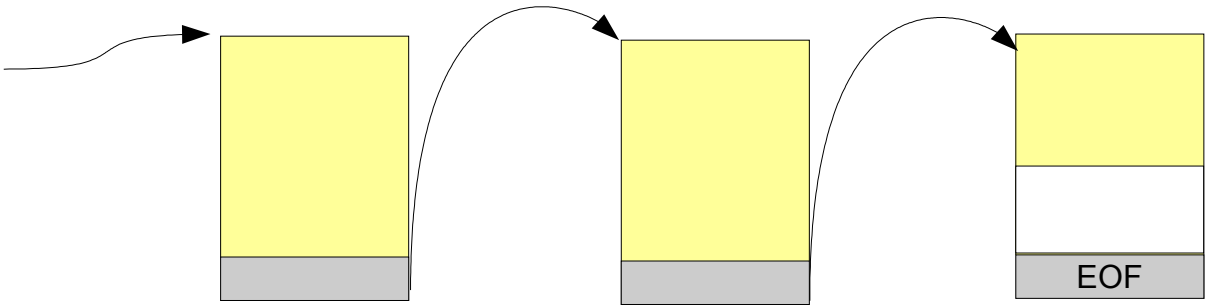


Gestión del espacio ocupado

- Asignación consecutiva
  - Cada fichero ocupa un conjunto de bloques consecutivos
  - Acceso muy rápido
    - poco movimiento del cabezal de disco
  - Buscar un espacio libre suficiente consecutivo
    - First fit
    - Best fit
    - Worst fit
  - Fragmentación externa
    - El espacio libre se distribuye en espacios pequeños dificiles de aprovechar
    - Hay que compactar la información
  - Que pasa si el fichero crece o no hay un espacio suficiente?

Gestión del espacio ocupado (II)

- Asignación entralazada
  - Cada bloque de datos apunta al siguiente
    - perdida de espacio en el bloque de datos
  - Acceso secuencial eficiente
  - Acceso directo muy ineficiente
    - hay que recorrer los n-1 bloques anteriores para acceder al deseado
  - Poco fiable
    - Si se pierde un bloque se pierde todo el fichero

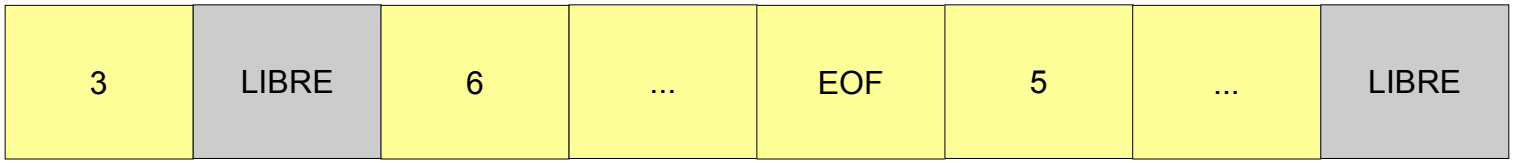


Gestión del espacio ocupado (III)

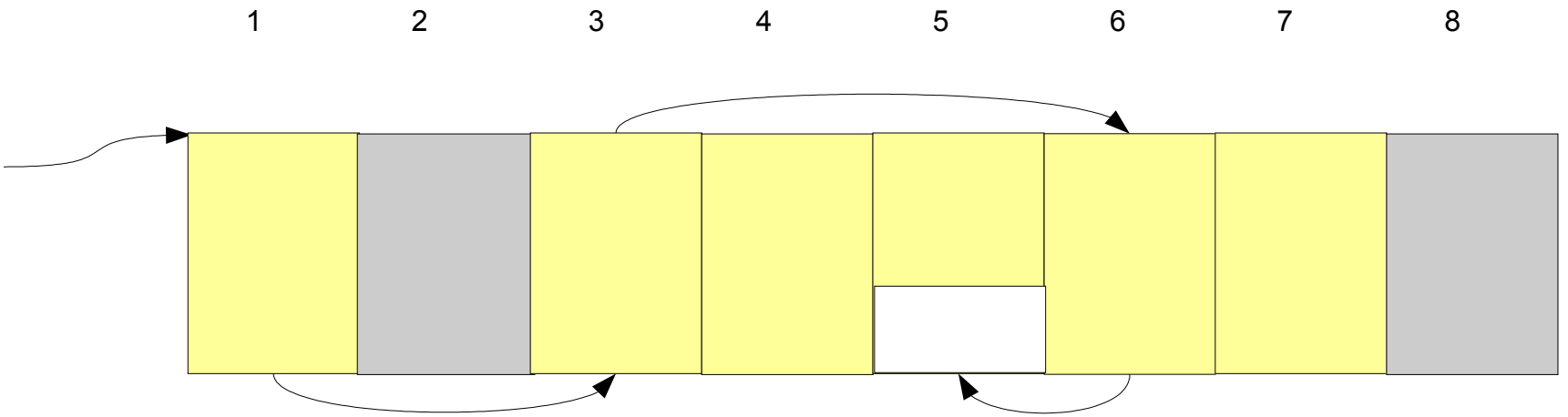
- Tabla de asignación entralazada (FAT)
  - Reservamos una parte del disco para almacenar los apuntadores
    - Un apuntador por bloque
    - Apunta al siguiente bloque o indica EOF
  - Acceso directo más rapido
    - Menos accesos a disco
    - Si es pequeña se puede guardar en memoria
  - Tamaño de los apuntadores
    - Nos limita el número del bloques máximo del dispositivo
    - FAT16 / FAT32
  - Mayor fiabilidad
    - Se pueden tener varias copias de la tabla
  - Se puede utilizar como *bitmap* de espacio libre

Gestión del espacio ocupado (IV)

File Allocation Table (FAT)

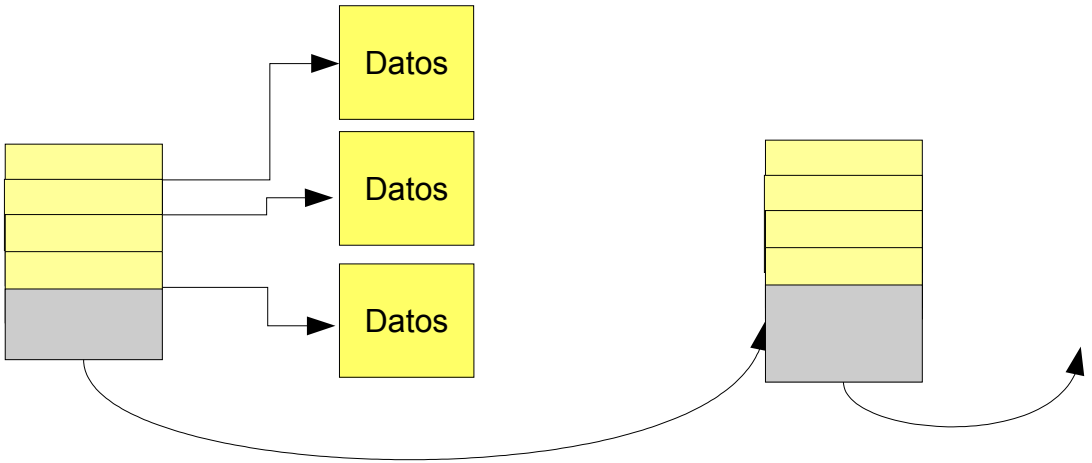


Bloques de datos



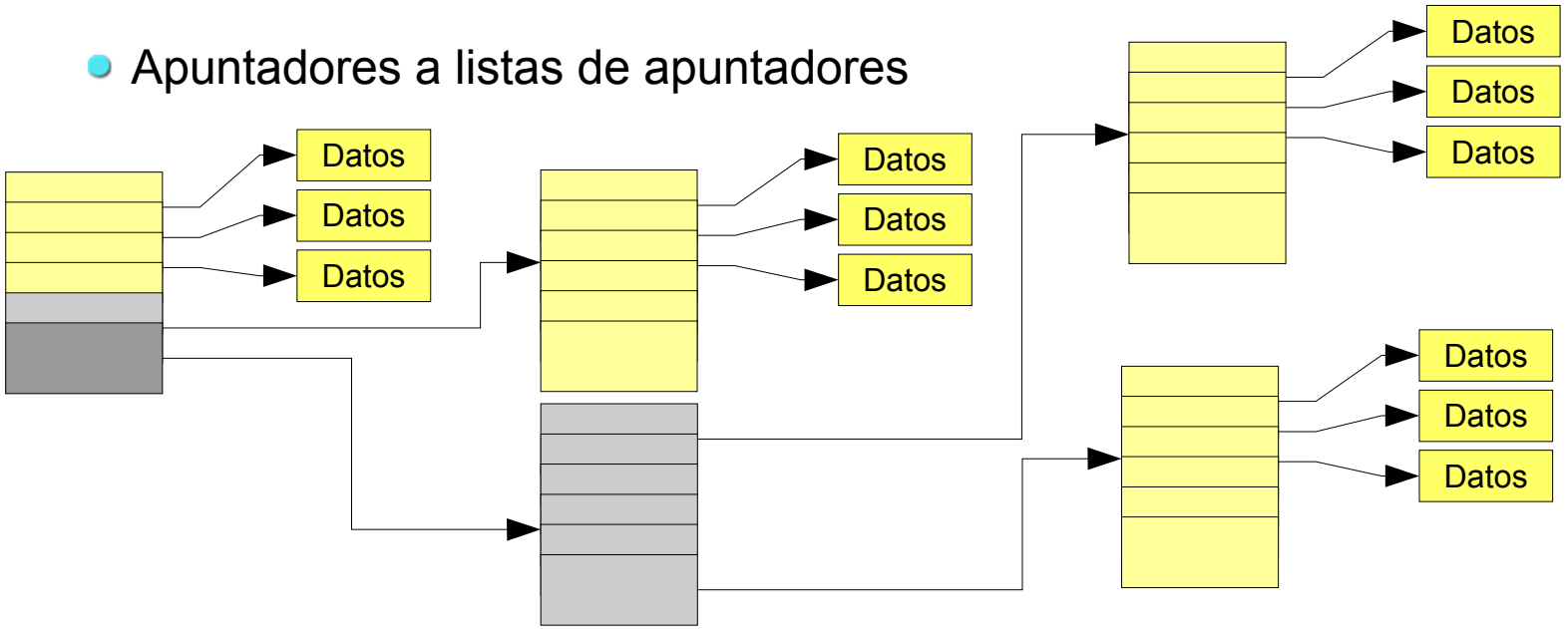
Gestión del espacio ocupado (V)

- Asignación indexada
  - Guardamos todos los apuntadores de un fichero en un bloque de datos
  - Acceso directo muy rápido
  - Perdida de espacio si no se usan todos los apuntadores
  - El último apuntador apunta a un nuevo bloque de apuntadores



Gestión del espacio ocupado (VI)

- Asignación indexada multinivel
  - Bloques de indices grandes
    - Mucho espacio despercidado en ficheros pequeños
  - Bloques de indices pequeños
    - Demasiadas indirecciones en ficheros grandes
  - Solución mixta
    - Apuntadores a listas de apuntadores



Organización de la información

- Directorios
  - Función principal: traducción @lógica a @física
  - Información mínima
    - nombre del fichero (@lógica)
    - @física
      - 1er bloque ocupado (asignación enlazada)
      - 1er bloque de índices (asignación indexada)
  - Información adicional de los ficheros
    - Protecciones
    - Tipo de fichero (datos, directorio, *soft link*, ...)
    - Tamaño
    - fechas de creación
    - #enlaces (para tener *hard links*)

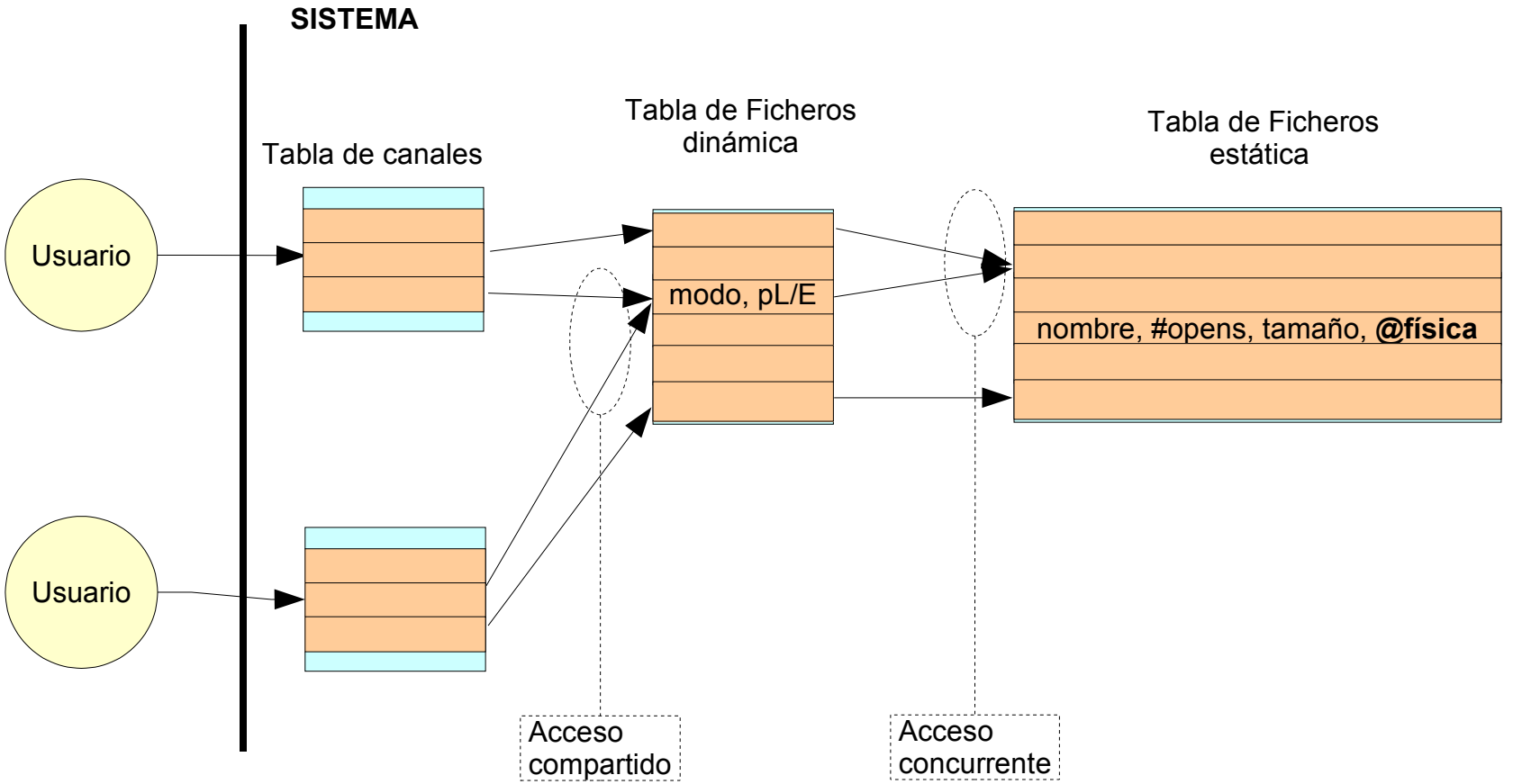
Organización de la información (II)

- Dónde esta la información adicional?
  - En los directorios (directorios con mucha información)
  - En otro lugar apuntado por la dirección física (directorios con poca información)

Estructuras en memoria

- leer(“fichero”,#bloque,&buf,lon)
  - Flexible
  - En cada operación hay que localizar el fichero
- fd = abrir(“fichero”,...)
  - localiza el fichero a partir del path relativo o absoluto
  - comprueba los permisos
- leer(fd,#bloque,&buf,lon)
  - localiza el bloque de datos y accede a él
  - El usuario tiene que llevar la cuenta de lo que esta leyendo/escribiendo
- leer(fd,&buf,lon) y posicionar(fd,num_byte)
  - permiten acceso secuencial y aleatorio

Estructuras en Memoria



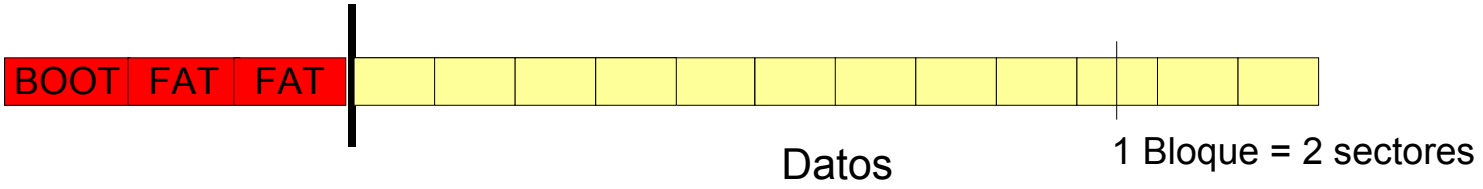
Planificación de disco

- $T_{total} = T_{posicionamiento} + T_{espera} + T_{transferencia}$
- First Come First Served
  - Simple
  - Justo
  - No optimiza los accesos al disco
- Shortest Seek Time First (SSTF)
  - Sirve primero las peticiones más cercanas
  - Puede provocar inanición

Planificación del disco (II)

- SCAN (algoritmo del ascensor)
  - Se hace un barrido disco sirviendo las peticiones que se encuentra a su paso
  - Cuando llega al final da la vuelta y sigue atendiendo peticiones
- LOOK
  - Cómo SCAN pero si no quedan peticiones por atender en la dirección actual da la vuelta
- C-SCAN, C-LOOK
  - Igual que las anteriores pero siempre empiezan por el principio.

Onion (DOS)

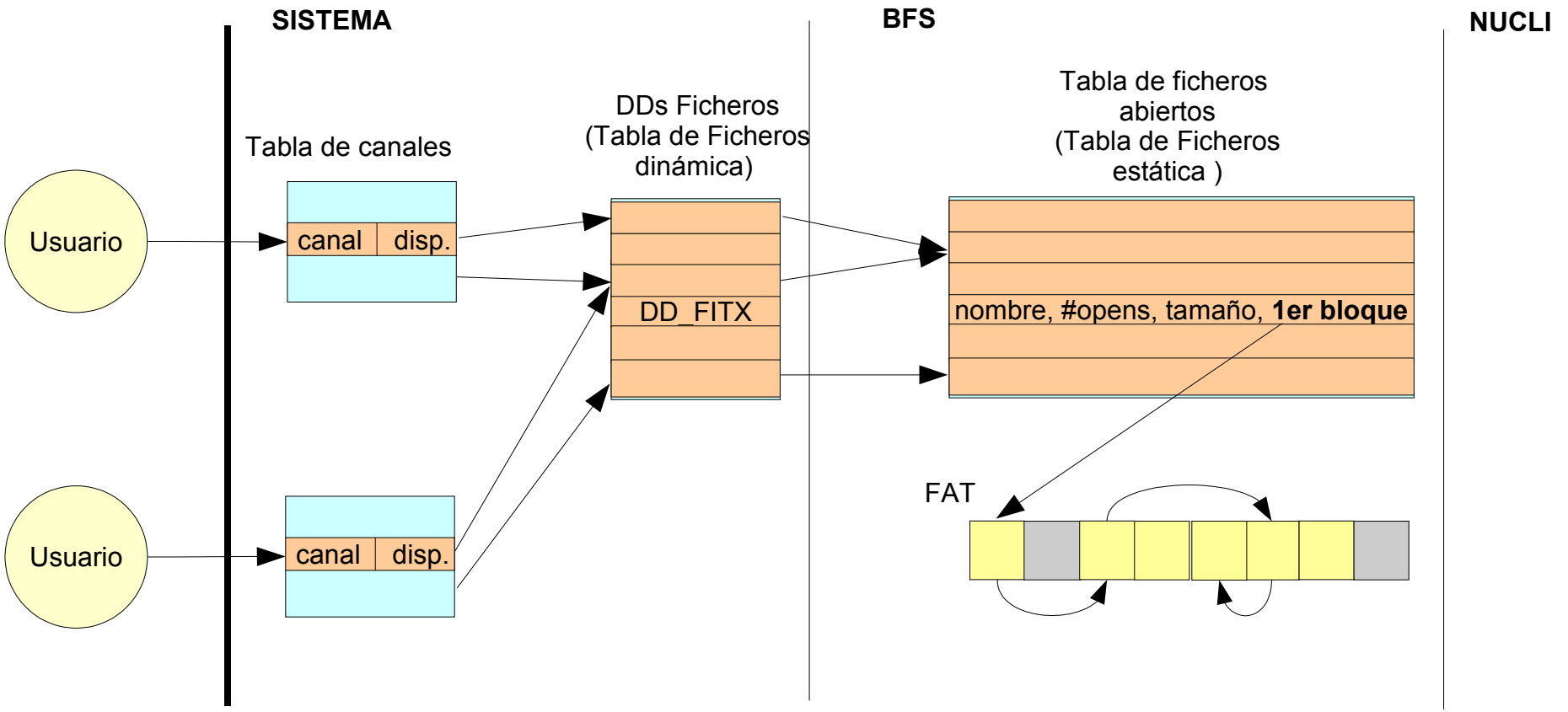


- FAT
  - 12 bits por entrada
  - Marca también el espacio libre
  - Ocupa 1 bloque (1 Kb / 12 bits = 768 entradas)
- En grafo
  - No hay *hard links*
  - Pueden haber *soft links*

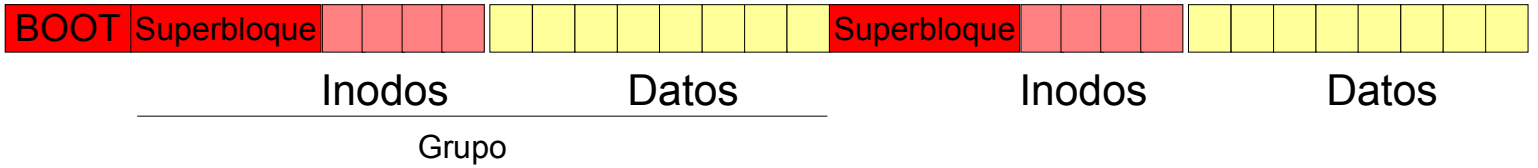
Onion (DOS)

- Directorios
  - Información en el directorio
  - 32 bytes por entrada
    - Nombre + Extensión (8.3)
    - Primer bloque de datos
    - Atributos (Subdirectory, System, Hidden, Read-only, ...)
    - Tamaño

Onion (DOS)


SISTEMA
BFS
NUCLI
Tabla de canales
DDs Ficheros
(Tabla de Ficheros
dinámica)
Tabla de ficheros
abiertos
(Tabla de Ficheros
estática)
Usuario
canal
disp.
DD_FITX
nombre, #opens, tamaño, 1er bloque
Usuario
canal
disp.
FAT

UNIX



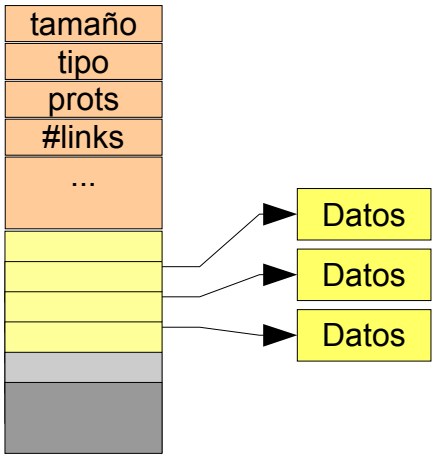
- Superbloque
  - Formato del SF (tamaño de bloque, #inodos, #bloques de datos,...)
  - Lista de bloques libres
  - Lista de inodos libres
- Asignación indexada
  - Almacenada en los inodos
- En grafo
  - *hard links* y *soft links*

UNIX

- Directorios
  - Información de los ficheros en los inodos
  - Entradas del directorio:
    - Nombre
    - #inodo

Inodo

- Bloque que almacena toda la información relativa a un fichero
  - tamaño
  - tipo (directory, file, link, block device, char device, fifo, ...)
  - protecciones
  - tiempos de acceso, modificación, creación
  - #links
- Indices a los bloques de datos (1/4 Kb)
  - 10 indices directos (10 bloques = 10/40Kb)
  - 1 indice indirecto (256/1024 bloques = 256Kb/4Mb)
  - 1 indice indirecto doble (65K/1M bloques = 65Mb/4 Gb)
  - 1 indice triple indirecto (16M/1G bloques = 16Gb/4 Tb)



UNIX

- Tabla de ficheros abiertos (Tabla de ficheros dinámica)
- Tabla de Inodos (Tabla de ficheros estática)
  - Mantiene en memoría una copia de cada inodo que este en uso
    - *cache* de inodos
  - Cuando se hace un open si no estaba se carga en la tabla
  - Cuando se hace el último close se libera de la tabla
- Buffer cache
  - Unificada para todos los accesos a dispositivos

VFS

- Virtual File System
  - Los sistemas operativos soportan diferentes sistemas de ficheros
    - Linux: Ext2, Ext3, FAT, ISO9660, XFS, RaiserFS, ...
    - Windows: NTFS, ISO9660, ...
  - Estructura en dos niveles:
    - Estructuras independientes del sistema de ficheros
      - contiene descripciones de los sistemas soportados
        - file_operations, inode_operations, superblock_operations
      - virtual i-nodes y virtual files
      - Las llamadas de sistema interaccionan con estas estructuras independientes
        - vfs_create, vfs_unlink, ...
    - Estructuras dependientes del sistema de ficheros
      - accedidas a traves de las operaciones descritas en el VFS
      - i-nodes, FAT, ...

Journaling

- Sistemas de ficheros transaccionales
  - Las escrituras se escriben en la buffer cache y se sincronizan cada cierto tiempo
    - Problema: si el servidor sufre una anomalia antes (perdida de corriente, ...) el sistema de ficheros puede quedar inconsistente
  - Programas de verificación al arrancar (fsck, scandisk)
    - Demasiado tiempo en discos grandes
  - Solución: Aplicar ideas de base de datos
    - Se guarda un registro de las transacciones al disco
    - Se utiliza para reconstruir las operaciones en caso de inconsistencia
    - Se puede utilizar para metadatos y/o datos
  - SF con Journaling: XFS, NTFS, ReiserFS, JFS, Ext3

RAID

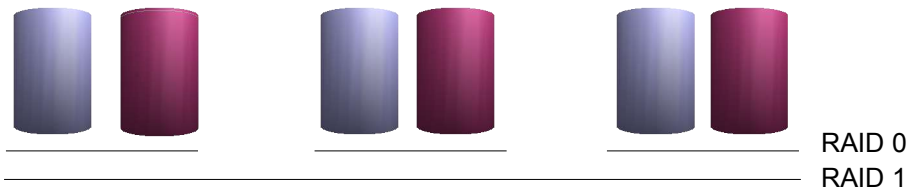
- Redundant Array of Inexpensive disks
  - Utilizar varios discos para conseguir mayor rendimiento y/o fiabilidad y/o capacidad
  - Se utiliza como si fuera un único disco
  - Se puede obtener con una controladora hardware o con un driver software
  - La forma de utilización particular sobre los discos se define por el nivel de configuración (0 a 6).
    - Se pueden combinar entre si.
  - En general, las particiones han de ser del mismo tamaño.

RAID

- Supongamos que tenemos N discos de tamaño B.
- Nivel 0 (Striping)
  - Capacidad total: NxB
  - Se distribuyen los datos entre los N discos
  - Podemos tener diferentes accesos en paralelo sobre los diferentes discos
  - Si falla un disco se pierden muchas “partes” de ficheros

RAID

- Nivel 1 (Mirroring)
  - Capacidad total: N
  - Cada disco es una copia exacta del otro. Hacen falta N fallos para perder la información.
  - Cada escritura supone N accesos. Las lecturas se pueden distribuir entre los discos
- Nivel 1+0
  - N >= 4
  - Se agrupan y una serie de disco para hacer raid 0 y a su vez se replica ese conjunto haciendo raid 1
  - Ej.: N = 6



RAID

- Nivel 5
  - Se guarda información de paridad que permite verificar/reconstruir los datos
    - Cada bit de paridad proviene de información de todos los discos
    - Los bits de paridad se distribuyen a través de todos los discos
  - Lecturas y escrituras en paralelo
  - Capacidad total: (N – 1)*B