Sistema de ficheros

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

Licencia Creative Commons

Licencia Creative Commons

Contenido

Introducción

- Sistema de Ficheros
  - Uno de los componentes más importantes de un SO
  - Provee abstracciones para gestionar los dispositivos de almacenamiento secundario qué
    - Son no volátiles
    - Se pueden compartir entre procesos
    - Se pueden organizar
    - Independientes del dispositivo físico
  - Son los ficheros
  - Establece un espacio de nombres a través del cual se puede referenciar unívocamente un fichero
  - Son los nombres simbólicos

Introducción

- Archivo o Fichero
  - Conjunto de información relacionada organizada como una secuencia de bytes que tiene un nombre
    - Se clasifican según criterios del usuario
  - Son dispositivos lógicos gestionados por el SO
    - Los programas acceden con las llamadas de sistema de E/S
      - open, read, write, close, ..
    - Y algunas llamadas específicas
      - unlink, chmod, chown, ...
  - Se pueden definir reglas de acceso para que sean compartidos

Introducción

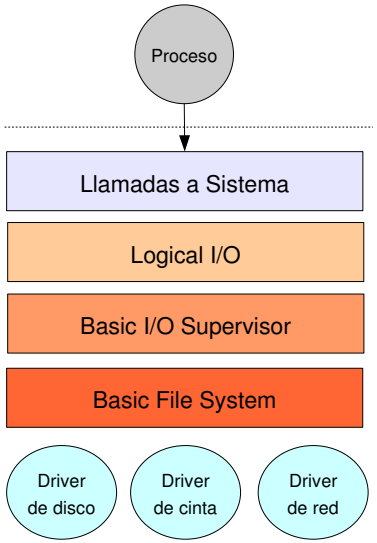
- Responsabilidades del SF
  - Asignar espacio libre a los ficheros
  - Liberar el espacio eliminado de los ficheros
  - Encontrar/Almacenar los datos de los ficheros
  - Garantizar las protecciones de los ficheros
  - ... y todo esto de manera transparente al usuario

Introducción

- Enlace (*link*)
  - Relación entre un nombre simbólico y una @ en el disco
  - Estas relaciones pueden ser N:1
    - Varios nombres para una misma @ en el disco
- 2 tipos de enlaces
  - Enlaces duros (*hard links*)
    - Sólo dentro de un mismo dispositivo
  - Enlaces simbólicos (*soft links*)
    - Pseudo-enlaces
    - Fichero especial que contiene el nombre de otro fichero
    - Interpretados por el S.O para simular *hard links*

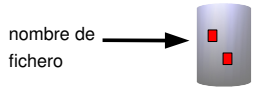
Introducción

- Arquitectura del SF
  - Basic File System
    - Se encarga de mover bloques entre la memoria y los dispositivos
    - No sabe lo que es un fichero
  - Basic I/O Supervisor
    - Se encarga de gestionar la E/S sobre los ficheros
    - Se encarga de la planificación de los accesos
  - Logical I/O
    - Proporciona la abstracción fichero



Directorios

- Directorio
  - Contiene información sobre los archivos
    - atributos
      - tipo de archivo
      - fechas de creación, acceso, modificación, ...
      - propietario
      - permisos
      - tamaño
      - ...
    - ubicación en el dispositivo de almacenamiento
  - Es un archivo especial gestionado por el Sistema
    - no accesible directamente por los usuarios
  - Permite traducir los nombres simbólicos de los ficheros a su ubicación en el sistema de ficheros

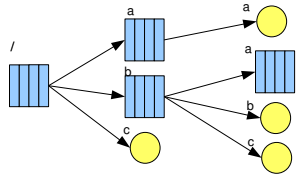


Operaciones sobre directorios

- Operaciones sobre un directorio
  - Buscar una entrada
    - Localizar en el directorio la entrada correspondiente a un fichero
  - Crear una entrada
    - Al crearse un nuevo archivo o subdirectorio
  - Borrar una entrada
    - Al eliminarse un archivo o subdirectorio
  - Enumerar entradas
    - Permite obtener una lista de todos los archivos o subdirectorios dentro del directorio
  - Actualizar entrada
    - Al cambiar algún atributo de un archivo o subdirectorio

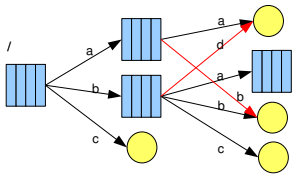
Estructura de los directorios

- Estructura externa: jerárquica o en árbol
  - Existe un directorio raíz
  - Cada directorio puede contener ficheros y/o directorios
  - Cada fichero/directorio se puede referenciar de dos maneras
    - Nombre absoluto (único): Camino desde la raíz + nombre
    - Nombre relativo: Camino desde el directorio de trabajo + nombre
      - / separa los componentes del camino
      - . indica el propio directorio
      - .. indica el directorio padre en el árbol



Estructura de los directorios (II)

- En grafo
  - Generalización de la estructura de arbol
    - Permitiendo enlaces múltiples
  - Dos tipos
    - Acíclicos: el S.F. verifica que no se creen ciclos
    - Cíclicos: el S.F. ha de controlar los ciclos infinitos

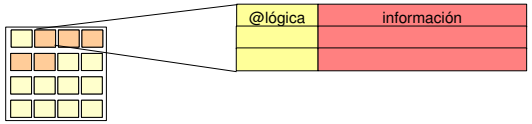


Estructura de los directorios (III)

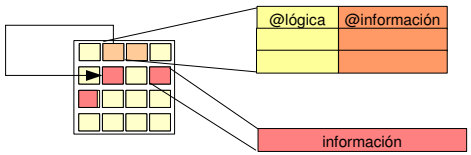
- Problemas de los directorios en grafos
  - Backups
    - No hacer copias del mismo fichero
  - Eliminación de un fichero
    - *Soft links*
      - El sistema no comprueba si hay *soft links* a un fichero
    - *Hard links*
      - Contar el número de referencias al fichero
      - Borrarlo cuando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

Estructura de los directorios (IV)

- Organización de la información
  - Todos los atributos dentro del directorio



  - Algunos atributos en el directorio
    - el resto en un registro de cabecera del fichero



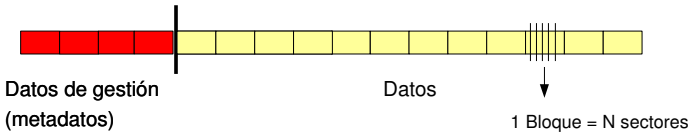
Estructura de los directorios (V)

- Estructura interna
  - Lista
  - Tablas de dispersión (*hashes*)
  - Árboles balanceados

Protecciones de los archivos

- El SF permite asignar diferentes permisos a los archivos
  - De esta manera podemos establecer diferentes niveles de acceso según quién acceda y que operación intente realizar
  - Algunos tipos de permisos:
    - Ninguno
    - Conocimiento
    - Ejecución
    - Lectura
    - Adición
    - Actualización
    - Cambios de protección
    - Borrado
  - Algunos clases de usuarios:
    - Usuario específico
    - Grupo de usuarios
    - Todos
  - Las clases pueden estar predefinidas o ser definidas por los usuarios (ACLs)

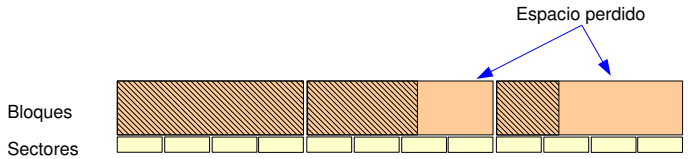
Organización del disco



- Información organizada en bloques
  - Sector: unidad de transferencia (definida por el Hw)
  - Bloque: unidad de asignación (definido por el SO)
    - que tamaño definimos?
- Metadatos
  - Que bloques se han asignado a cada fichero?
  - Que bloques no están siendo utilizados

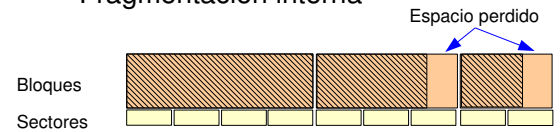
Bloques de tamaño fijo

- Todos los bloques tienen el mismo tamaño
  - Muy sencillo de implementar
  - Compromiso en el tamaño de bloque
    - Eficiencia
    - Fragmentación interna

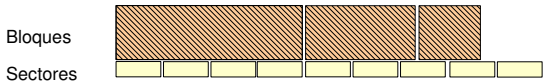


Bloques de tamaño variable

- Bloques sin compartir sectores
  - Fragmentación interna



- Bloques compartiendo sectores
  - Uso eficiente del espacio
  - Complejidad muy elevada en la implementación



Asignación de archivos

- Proporcionar espacio de almacenamiento secundario a los archivos
- El SF utiliza una estructura donde guarda la relación entre el archivo y su espacio asignado
  - Normalmente accesible a través del directorio
  - Almacenada en el SF (opcionalmente en memoria)
- El espacio se asigna en forma de bloques contiguos (secciones)
  - cuantos consecutivos?

Asignación de archivos (II)

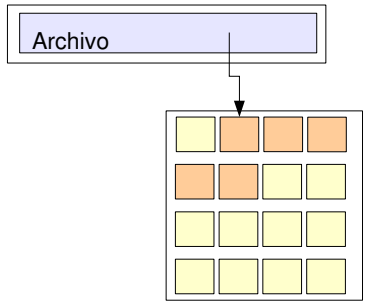
- Asignación previa
  - Determinar el tamaño del archivo *a priori* en su creación
  - La mayoría de las veces es difícil de saber
- Asignación dinámica
  - Cada vez que se necesita más espacio se produce una asignación

Asignación de archivos (III)

- Asignación por secciones de tamaño variable
  - Mayor localidad
  - Fragmentación externa
  - Técnicas de asignación
    - First fit
    - Best fit
    - Nearest fit
- Asignación por bloques
  - Cada vez que se necesita un bloque se asigna
  - Más flexible

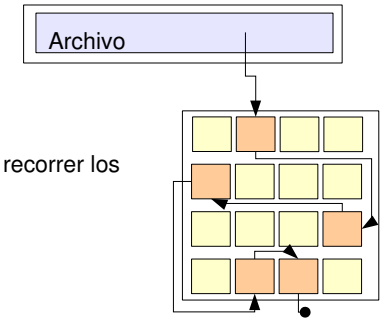
Asignación de archivos (IV)

- Asignación contigua
  - Todos los bloques del archivo se asignan de manera consecutiva
  - Se necesita una única entrada por archivo con:
    - Bloque inicial
    - Longitud del archivo
  - Ventajas:
    - Acceso eficiente al disco
    - Localización del bloque i-ésimo sencilla
  - Desventajas
    - se produce fragmentación externa
    - necesita asignación previa
      - CDROMs, DVDs, ...



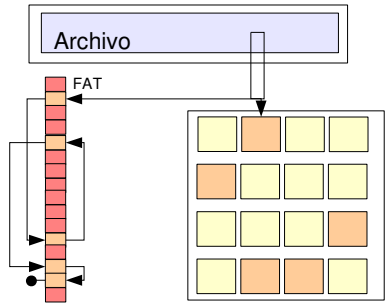
Asignación de archivos (V)

- Asignación encadenada
  - Cada bloque de datos reserva espacio para un puntero que indica cual es el siguiente bloque del archivo
  - Se necesita una entrada por archivo con:
    - Bloque inicial
  - Ventajas:
    - asignación previa o dinámica
    - no hay fragmentación externa
  - Desventajas:
    - para acceder al bloque i-ésimo hay que recorrer los anteriores
      - adecuado para accesos secuenciales
      - terrible para accesos directos
    - Poca fiabilidad



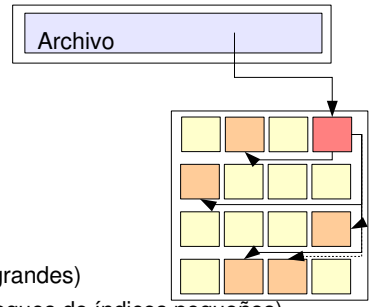
Asignación de archivos

- Asignación encadenada en tabla
  - Los punteros en lugar de guardarse en los bloques de datos se guardan todos juntos en una tabla
    - Esta tabla se suele llamar FAT (File Allocation Table)
  - Se necesita una entrada por archivo con:
    - Bloque inicial
  - Ventajas sobre la anterior:
    - Para acceder al bloque i-ésimo basta con acceder a la tabla
    - Se puede replicar la tabla para aumentar la fiabilidad
    - Se puede utilizar para gestionar el espacio libre (*bitmap*)



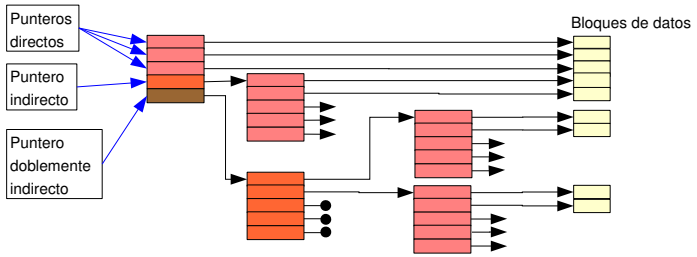
Asignación de archivos (VI)

- Asignación indexada
  - Por bloques o por secciones
  - Existe un bloque índice de un nivel para cada archivo
  - Este índice contiene una entrada para cada
    - bloque: #bloque
    - sección: inicio y longitud de la sección
  - Al final hay un puntero al siguiente bloque índice (o a NULL)
  - Ventajas:
    - Buen acceso secuencial y directo
  - Desventajas:
    - Perdida de espacio (bloques de índices grandes)
    - Muchos accesos en ficheros grandes (bloques de índices pequeños)



Asignación de archivos (VII)

- Asignación indexada multinivel
  - En el bloque índice existen algunos apuntadores indirectos
    - apuntan a nuevos bloques índices
  - Se crea una estructura jerárquica de índices
  - Ventajas
    - Muy pocos accesos incluso en ficheros grandes
    - Poca perdida de espacio en ficheros pequeños

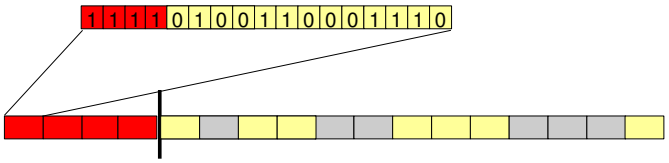


Gestión del espacio libre

- El S.F. tiene una estructura que gestiona el espacio libre del dispositivo (Tabla de asignación de disco)
  - Nos indica que zonas del disco se encuentran libres
- Típicamente se usa
  - Tabla de bits
  - Secciones libres encadenadas
  - Indexación
  - Lista de bloques libres

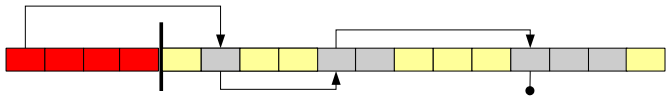
Gestión del espacio libre (II)

- Tabla de bits (*bitmap*)
  - Contiene un bit por cada bloque del disco
    - 0 = bloque libre
    - 1 = bloque ocupado
  - Ventajas:
    - Relativamente fácil encontrar un bloque libre o un grupo contiguo de bloques libres
    - Ocupa poco espacio (se puede tener en memoria)



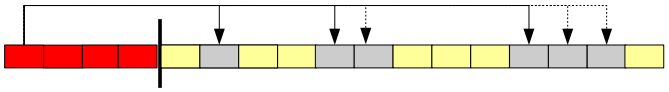
Gestión del espacio libre (III)

- Secciones libres encadenadas
  - Apuntamos al primer bloque de un grupo de bloques libres
  - En ese bloque se guarda
    - Número de bloques libres consecutivos
    - Dirección del siguiente grupo
  - Ventajas
    - No requiere espacio adicional para guarda que bloques están libres
  - Problemas
    - La gestión de la lista puede ser ineficiente (si entran en juego muchos bloques)
    - Fragmentación



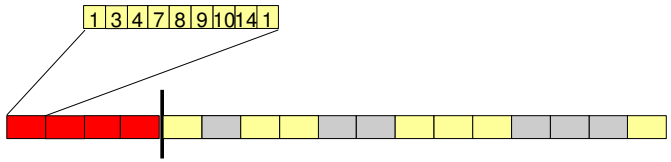
Gestión del espacio libre (IV)

- Indexación
  - Trata el espacio libre como un archivo
  - Utiliza la misma técnica de indexación vista para archivos
  - Ventajas
    - Acceso eficiente tanto para asignación por secciones como por bloques



Gestión del espacio libre (V)

- Lista de bloques libres
  - Se reserva una zona del disco donde se almacena una lista con los bloques libres
  - La lista es demasiado grande para mantenerse en memoria
    - Si se trata como una pila, la cima puede estar en memoria
    - Si se trata como una cola, el principio y el final pueden estar 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.

RAID

- Redundant Array of Inexpensive Disks
  - Utilizar varios discos para conseguir mayor rendimiento y/o fiabilidad y/o capacidad
  - Como si fueran 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/discos han de ser del mismo tamaño.

RAID

- Supongamos que tenemos N discos de tamaño B.
- Nivel 0 (Striping)
  - Capacidad total: N*B
  - 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

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 bases 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

UNIX

- Los archivos son flujos de bytes sin ningún formato específico
- Tipos de archivos
  - Ordinarios
  - Directorios
  - Links
  - Especiales (character, block, socket)
    - Se utilizan para acceder a dispositivos tanto físicos como lógicos
  - Named pipes
    - Ficheros especiales de comunicación

UNIX

- Protecciones
  - Tres clases de usuarios y tres tipos de permisos básicos
  - Clase propietario
    - se aplica al propietario del archivo
  - Clase grupo
    - se aplica a los usuarios que pertenecen al grupo del archivo
  - Clase otros
    - cualquier usuario no incluido en las dos anteriores
  - Permisos:
    - lectura (r)
    - escritura (w)
    - ejecución (x)

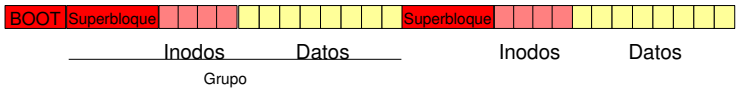
UNIX

- Llamadas a sistema
  - chown – permite cambiar el propietario de un archivo
  - chgrp – permite cambiar el grupo de un archivo
  - link – asocia un nuevo nombre a un archivo
  - symlink – crear un enlace simbólico que apunta a un archivo
  - unlink – elimina una enlace a un archivo
  - readlink – lee el contenido de un enlance simbólico
  - flock – coloca un lock en un archivo
  - stat, lstat, fstat – permiten obtener la información almacenada en el inodo del fichero

UNIX

- Llamadas a sistema sobre directorios
  - mkdir – crea un nuevo directorio
  - rmdir – elimina un directorio
  - opendir – abre un directorio para recorrer sus entradas
  - readdir – permite leer las entradas de un directorio
  - closedir – cierra un canal asociado a un directorio

UNIX



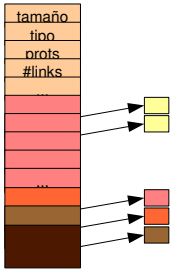
- Superbloque
  - Formato del SF (tamaño de bloque, #inodos, #bloques de datos,...)
  - Lista de bloques libres
  - Lista de inodos libres
- Asignación indexada multinivel
  - Índices almacenados en los inodos

UNIX

- Directorios
  - Son archivos ordinarios que se acceden a través de llamadas especiales del SO
  - Organizados en forma de grafo
  - Información de los ficheros en los inodos
  - Entradas del directorio:
    - Nombre
    - #inodo

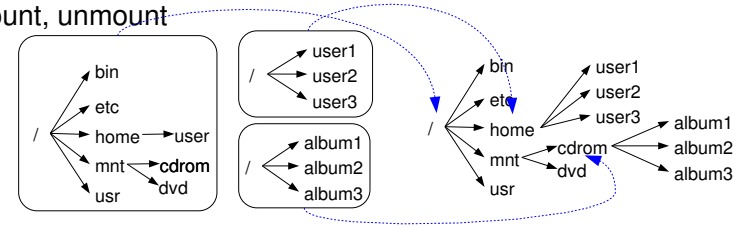
UNIX

- Inodo
  - Bloque que almacena toda la información relativa a un fichero
    - tamaño
    - tipo
    - protecciones
    - propietario, grupo
    - tiempos de acceso, modificación, creación
    - #enlaces al inodo
  - 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)

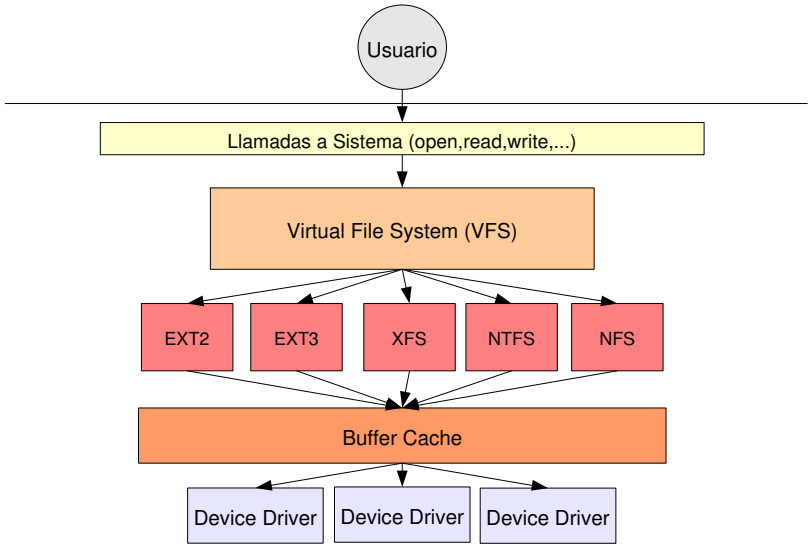


UNIX

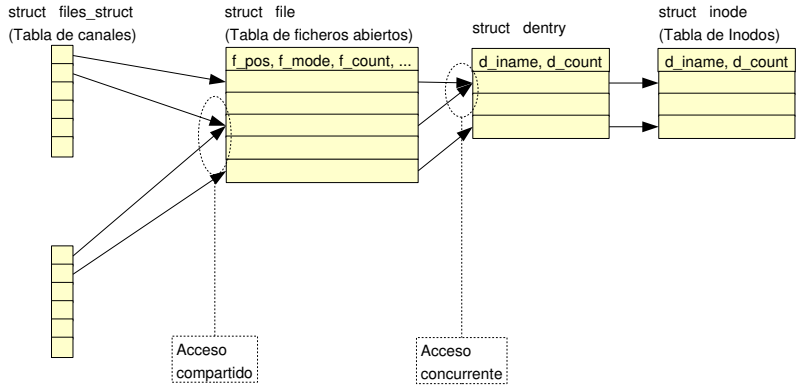
- Para poder acceder al SF de un dispositivo primero se ha de *montar*
- Existe un dispositivo raíz que se *monta* en la / del sistema de ficheros
- Los demás dispositivos de almacenamiento se pueden *montar* en cualquier punto del SF
  - mount, unmount



Linux



Estructuras: Linux/UNIX



UNIX

- Tabla de ficheros abiertos
  - Información sobre cada obertura sobre un fichero
- Tabla de Inodos
  - 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
  - Cache de bloques
  - Unificada para todos los accesos a dispositivos

Linux: VFS

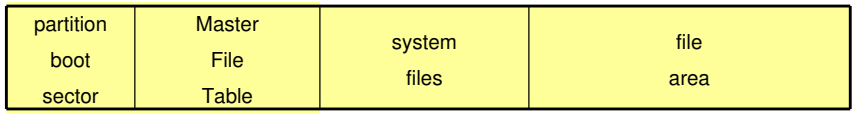
- Virtual File System
  - Los sistemas operativos soportan diferentes sistemas de ficheros
    - Linux: Ext2, Ext3, FAT, ISO9660, XFS, RaiserFS, NTFS, ...
  - Estructura en dos niveles:
    - Estructuras independientes del sistema de ficheros
      - contiene descripciones de los sistemas soportados
        - 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, ...

NTFS

- Ofrece:
  - Cuotas de disco
  - Encripción
  - Puntos de montaje de volúmenes
  - Ficheros dispersos
  - Journaling

NTFS

- New Technology File System
- Disco dividido en CLUSTERs:
  - Un cluster contiene varios sectores:
    - Disco de más de 4 GB: 1 cluster = 16 sectores
- En NTFS todo son ficheros
- Formato:



NTFS

- Partition boot sector
  - 16 sectores (principio de la partición)

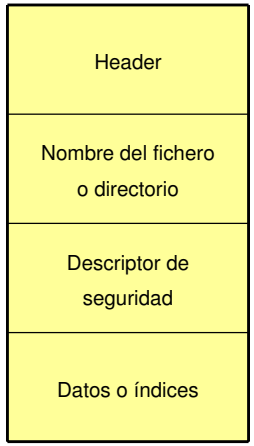
Byte offset	Longitud del campo	Nombre
0x00	3 bytes	Jump
0x03	LONGLONG	OEM ID (S/N)
0x0B	25 bytes	BPB (info partición)
0x24	48 bytes	EBPB (@MTF)
0x54	426 bytes	Bootstrap code
0x01FE	WORD	End of sector marker

NTFS

- Master File Table (MFT)
  - 12% del espacio del disco duro
    - Espacio reservado: MTF-zone
    - Se incrementa y se reduce de forma dinámica
      - Crecer: doblar el espacio actual
      - Reducir: mitad del espacio actual
  - Compuesta por RECORDs < 4KB
    - 1 record guarda información de 1 fichero
    - En algunos casos, se necesita mas de un record por fichero

NTFS

- Record:



Información sobre el fichero y el record:
-timestamp, link count, journaling

Nombre “regular” del fichero
-puede haber mas de uno

Describe quien lo posee y quien tiene acceso

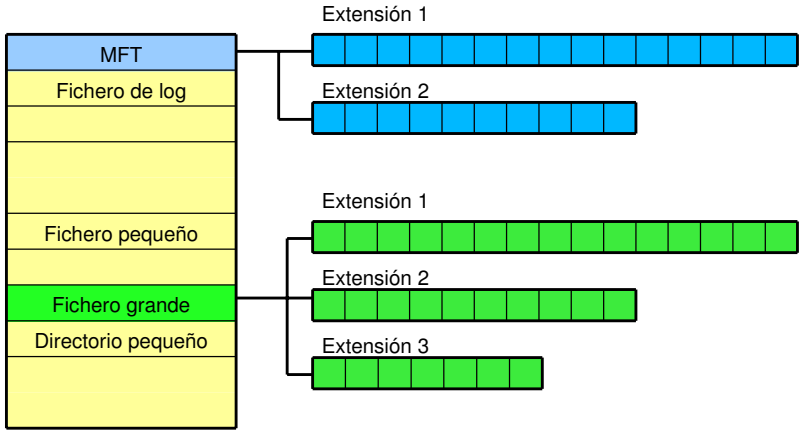
Datos del fichero
Indices a los datos del fichero

NTFS

- 2 clases de records:
  - Dependen del tamaño del fichero a que referencia:
    - Si el fichero < 1500 bytes
      - Record con atributos residentes
      - Incluyen los datos del fichero
      - Optimizan el acceso a disco
    - Si el fichero > 1500 bytes
      - Record con índices
      - índices que apuntan a bloques de información

NTFS

- Visión de la MFT:

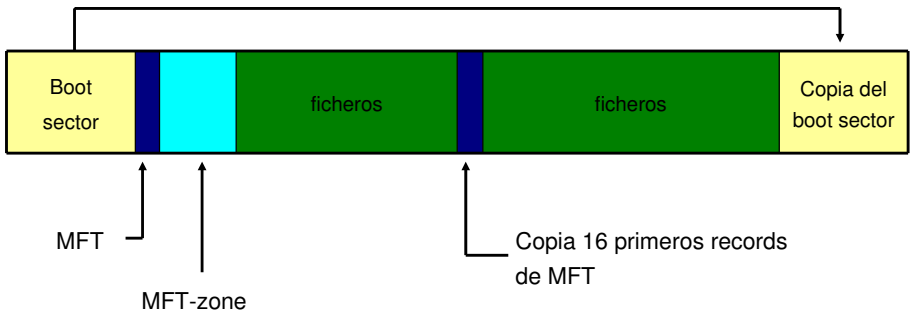


NTFS

- 16 primeros records reservados:
  - Útiles para el mantenimiento del SF:
    - $MFT: puntero a MFT
    - $MFTmirr: copia del MFT
    - $LogFile: fichero de journaling
    - $Volume: Información sobre el volumen
    - $AttrDef: listado de atributos del volumen
    - $.: Directorio raiz
    - $Bitmap: Bitmap del espacio libre
    - $Boot: Sector de boot
    - $Quota: fichero de información sobre quotas
    - $Upcase: Relación entre ficheros con nombre en mayúsculas y minúsculas
    - $Extend: extensión de la información

NTFS

- Particion real:



NTFS

- Directorios:
  - Compuesto por bloques:

Nombre fichero/directorio	Atributos de seguridad	Record de la MFT

  - B-tree:
    - Inserción en orden
    - Búsqueda dicotómica
      - #accesos = $\log_2$ #ficheros

NTFS

- Directorios: búsqueda de Vtext.dll

Vcmd.exe
Spchtel.dll
Speech.cnt
Speech.dll
Speech.hlp
Vcauto.tlb
Vcmshl.dll
Vdict.dll
Vtext.dll
Vtxauto.lib
wrapSAPI.dll
Xcommand.dll
Xlisten.dll
Xtel.dll
Xvoice.dll

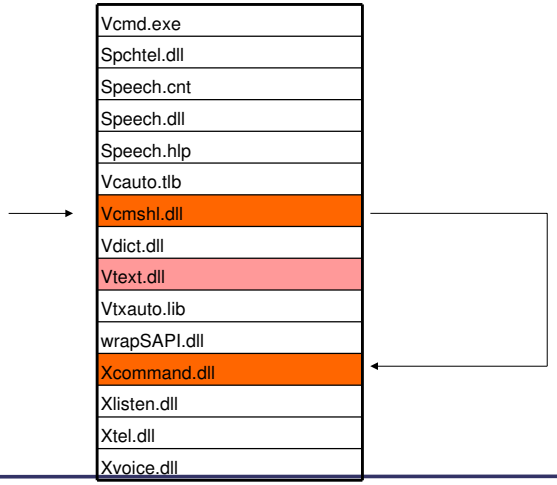
NTFS

- Directorios: búsqueda de Vtext.dll

Vcmd.exe
Spchtel.dll
Speech.cnt
Speech.dll
Speech.hlp
Vcauto.tlb
Vcmshl.dll
Vdict.dll
Vtext.dll
Vtxauto.lib
wrapSAPI.dll
Xcommand.dll
Xlisten.dll
Xtel.dll
Xvoice.dll

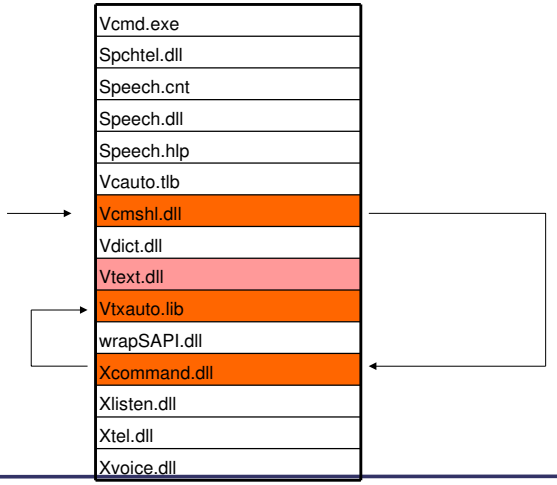
NTFS

- Directorios: búsqueda de Vtext.dll



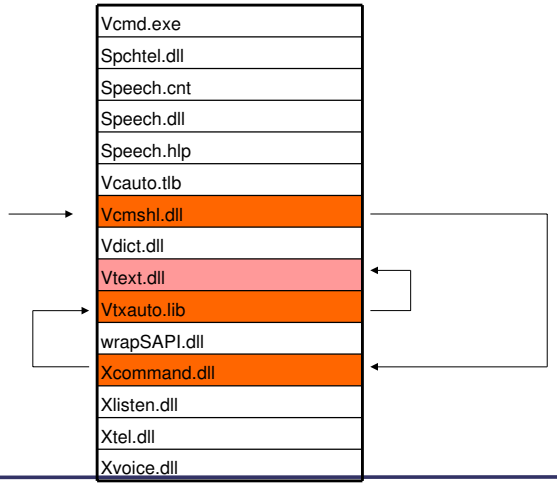
NTFS

- Directorios: búsqueda de Vtext.dll



NTFS

- Directorios: búsqueda de Vtext.dll



NTFS

- Hardlinks:
  - Semejantes a Linux
  - Solamente dentro de la misma partición
  - API:
    - CreateHardLink (nombre, fichero existente, atributos de seguridad);
  - Línea de comandos:
    - ln.exe

NTFS

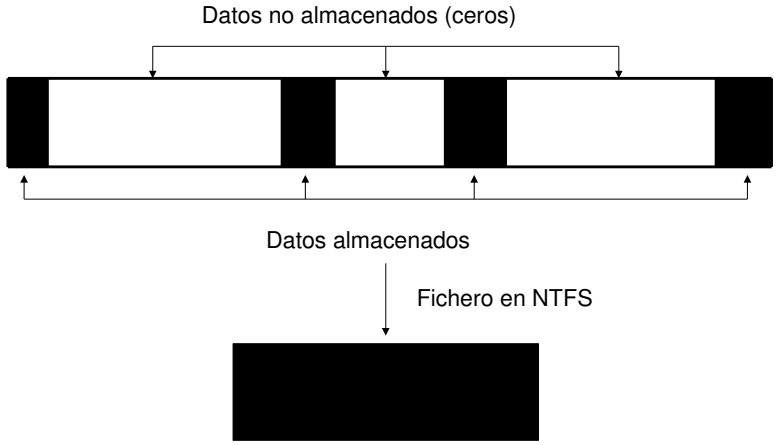
- Softlinks:
  - Semejantes a Linux
  - Fichero con extensión .lnk
  - Contiene:
    - Permisos
    - Ruta de acceso y nombre del fichero
  - Se crean mediante menú contextual:
    - Crear acceso directo

NTFS

- Ficheros dispersos:
  - No se almacenan en el disco las cadenas largas de ceros
  - Existe un atributo en el record que indica si el fichero es disperso
  - NTFS detecta si el acceso al fichero coincide con una cadena de ceros
  - Ahorra espacio en el disco

NTFS

- Ficheros dispersos:



NTFS

- Journaling:
  - Transacciones almacenadas en $LogFile
  - Tamaño de 2 MB a 4 MB
  - Una vez la transacción se ha finalizado, se elimina del $LogFile
  - Cada 5 segundos se estudia el $LogFile para llevar a disco y eliminar transacciones

NTFS

- Encriptación:
  - A nivel de fichero
  - API:
    - EncryptFile
    - DecryptFile
  - Servicio de W2K – EFS
  - Algoritmo: RSA

NTFS

- Compresión:
  - Efectuada en bloques de 16 clusters
  - Utiliza la fragmentación de ficheros para implementarla