

## T5 - Sistema de Ficheros

SO-Grado 2010-2011

5.1

## Licencia

- Este documento puede contener partes de las transparencias de la asignatura Sistemas Operativos del plan de estudios 2003 de la Facultat d'Informàtica de Barcelona
- Este documento puede contener partes de las transparencias que se proporcionan con el libro:
  - Operating Systems Concepts 8th edition. Silberschatz, Galvin and Gagne ©2009

5.2

## Índice

---

- Introducción
- Arquitectura de Sistema de Ficheros y VFS
- Conceptos y Servicios Básicos
- Organización y Gestión del Espacio en Disco
- Trabajar con Ficheros
- Fiabilidad
- Otros SFs & Arquitecturas

5.3

- 
1. ¿Qué es un Sistema de Ficheros?
  2. ¿Qué es un Archivo o Fichero?
  3. Tipos de Fichero
  4. ¿Para qué necesitamos un SF?

## INTRODUCCIÓN

5.4

## ¿Qué es un Sistema de Ficheros?

- Uno de los componentes más visibles del SO
- Proporciona el mecanismo para almacenar, localizar y recuperar datos y programas del SO y de los usuarios
- Es una colección de ficheros
- Y una estructura de directorios
  - Establece un espacio de nombres a través del cual se puede referenciar unívocamente un fichero
- Provee abstracciones para gestionar los dispositivos de almacenamiento secundario que:
  - Son no volátiles
  - Se pueden compartir entre procesos
  - Se pueden organizar
  - Independientes del dispositivo físico

5.5

## ¿Qué es un Archivo o Fichero?

- Conjunto de información relacionada organizada como una secuencia de bytes (espacio de direcciones lógico contiguo) que tiene un nombre
  - Se clasifican según criterios del usuario
    - Datos (alfanumérico, binario, etc.)
    - Programa (fichero ejecutable)
- 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

5.6

## Tipos de Fichero – Extensión (no Linux)

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

5.7

## ¿Para qué necesitamos un SF?

- Responsabilidades del Sistema de Ficheros
  - ▶ 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

5.8

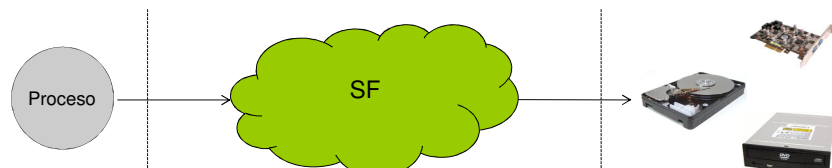
1. Necesidades del SF
2. Arquitectura de Capas del SF
3. VFS – Virtual File System

## ARQUITECTURA DEL SF & VFS

5.9

### Necesidades del sistema de ficheros

- Traducir los accesos desde la solicitud de la interfaz de usuario/aplicación hasta los drivers que controlan los dispositivos de almacenamiento.
- Internamente deberá implementar el propio sistema de ficheros que se está utilizando
  - Definir la apariencia del SF para el usuario
    - ▶ Definir archivo y sus atributos
    - ▶ Operaciones sobre archivos
    - ▶ Estructura directorios para organizar archivos
  - Definir algoritmos y estructuras para mapear SF lógico en los dispositivos físicos de almacenamiento secundario



## Arquitectura de Capas del SF

- Sistema de ficheros lógico
  - Proporciona la abstracción fichero para realizar la E/S
  - Protección, seguridad
  - Directorios
  - Información sobre archivo para siguiente nivel
- Modulo de organización archivos
  - Archivos – bloques lógicos/físicos
  - Administrador espacio libre
- Sistema de ficheros básico
  - Emitir comandos al driver para leer/escribir bloques físicos



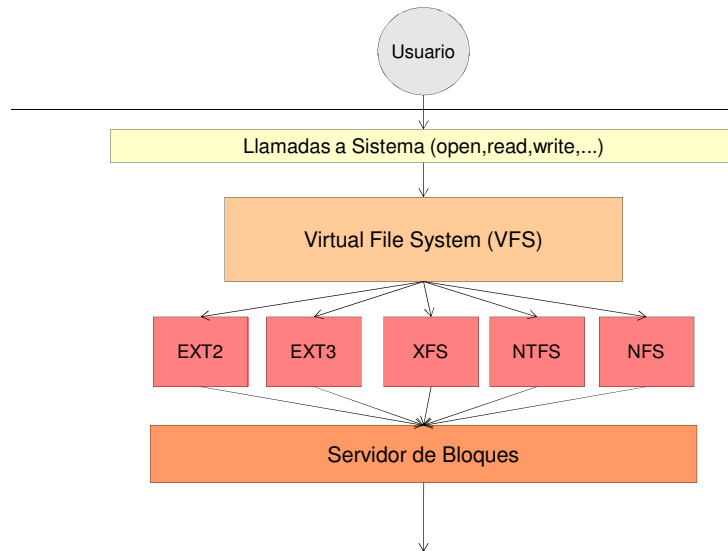
5.11

## VFS: Virtual File System

- Los SOs soportan diferentes sistemas de ficheros
  - Linux: Ext2, Ext3, FAT, ISO9660, XFS, RaiserFS, NTFS, ...
  - Es necesaria **una capa en la arquitectura de SF que permita abstraer todos los sistemas de ficheros que acepta el SO como si fuera uno sólo**
- Virtual File Systems (VFS) proporciona un mecanismo orientado a objetos para implementar sistemas de ficheros
  - VFS permite la *misma interfaz* de llamadas a sistema *para ser usada con diferentes tipos de sistemas de ficheros*
- Estructura en dos niveles:
  - Estructuras **independientes del sistema de ficheros**
    - ▶ Contiene descripciones de los sistemas soportados
    - ▶ Las llamadas de sistema interaccionan con estas estructuras independientes
  - Estructuras **dependientes del sistema de ficheros**
    - ▶ accedidas a través de las operaciones descritas en el VFS
      - sys\_open(), sys\_read(), sys\_write()...
    - ▶ estructuras internas para identificar ficheros, gestión de espacio de disco, etc

5.12

## VFS: Virtual File System



5.13

1. Enlace (link)
2. Directorio
3. I-Nodo
4. Protecciones de los Archivos
5. Servicios básicos sobre Ficheros
6. Operaciones sobre Directorios

## CONCEPTOS Y SERVICIOS BÁSICOS

5.14


## 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 (p.ej. Una misma partición)**
  - Enlaces simbólicos ( soft-links )
    - ▶ Pseudo-enlaces
    - ▶ Fichero especial que contiene el nombre de otro fichero
      - Hay SF que NO son capaces de procesarlo como pseudo-enlace
        - **P.ej.: FAT**
      - Hay SOs que implementan su propia versión de “soft-link”
        - Windows
    - ▶ Interpretados por el S.O para simular hard links

5.15

## 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

nombre de  
fichero → 

5.16



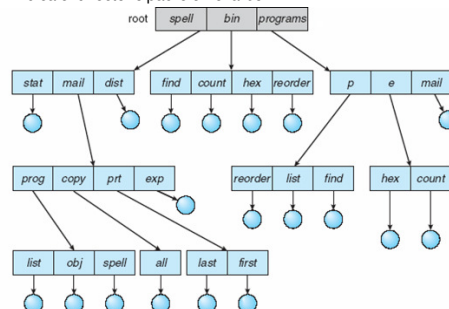
## Alternativas al uso de Directorios

- Sistemas basados en motores de búsqueda
  - Los ficheros se almacenan en una estructura interna que gestiona el SO
  - El usuario busca archivos en base a
    - ▶ Palabras clave, contenidos, información de contexto
  - A cada fichero se le inicializan adecuadamente los valores de los parámetros que se emplean para las búsquedas
  - Similar a una Base de Datos
- Ejemplos:
  - WinFS (Windows), Desktop Search (Google), Spotlight (Mac OS)

5.17

## Estructura Externa del Directorio

- Evolución desde 1 nivel, 2 niveles, **estructura jerárquica o 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



5.18



## Estructura Interna del Directorio

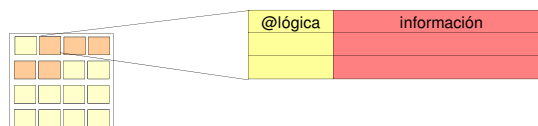
- ¿Cómo guardan la información de lo que contiene el directorio?
  - Lista
  - Tablas de Dispersión (Hash)
  - Árboles Balanceados
- Complejidad vs Rendimiento vs Necesidad

5.21

## Organización de la Información en el Directorio

- Todos los atributos dentro del directorio

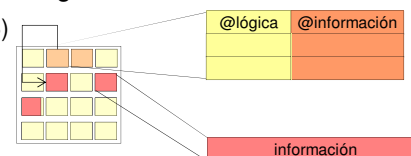
▶ (CP/M)



- Algunos atributos en el directorio

- El resto en un registro de cabecera del fichero

▶ (MS-DOS)



- Todos los atributos en una estructura externa

- **Sistema de I-Nodos**

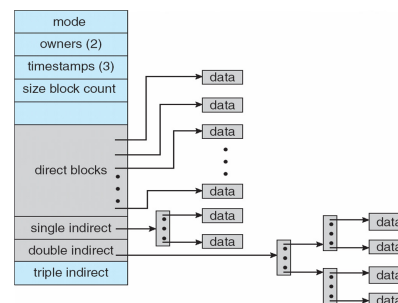
▶ (UNIX)

5.22

## I-Nodo: ¿Qué es?

- Estructura que almacena toda la información relativa a un fichero
  - tamaño
  - tipo
  - protecciones
  - propietario, grupo
  - tiempos de acceso, modificación, creación
  - #enlaces al inodo
  - Índices a los bloques de datos (Indexación Multinivel)
    - ▶ Lo hablaremos más adelante...

- Se trata de una estructura interna



5.23

## I-Nodo: Ventajas

- Toda la información sobre el fichero está localizada en una única estructura
  - se carga a memoria cuando va a ser usada
- La mayoría de los ficheros en un sistema Unix son de pequeño tamaño
  - Basta con los índices directos
- Los punteros a bloques indirectos permiten tener ficheros muy grandes
  - Es poco costoso acceder a distintos bloques en ficheros grandes
    - ▶ Aunque sean no consecutivos

5.24

## 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
- Algunas clases de usuarios:
  - Usuario específico
  - Grupo de usuarios
  - El resto de usuarios
- Las clases pueden estar predefinidas o ser definidas por los usuarios (ACLs)

5.25

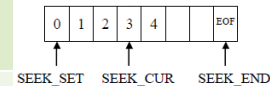
## Listas de Acceso y Grupos

- Algunas clases de usuarios:
    - Usuario específico, Grupo de usuarios, El resto de usuarios
  - Las clases pueden estar predefinidas o ser definidas por los usuarios (ACLs)
  - Ejemplo
    - Modos de acceso
      - Leer (Read), Escribir (Write), Ejecutar (Execute)
- |                       |   |   |       |
|-----------------------|---|---|-------|
| a) <b>propietario</b> | 7 | ⇒ | RWX   |
|                       |   |   | 1 1 1 |
| b) <b>grupo</b>       | 6 | ⇒ | RWX   |
|                       |   |   | 1 1 0 |
| c) <b>el resto</b>    | 1 | ⇒ | RWX   |
|                       |   |   | 0 0 1 |
- *Revisar Tema 2 (Procesos)*

5.26

## Servicios básicos sobre Ficheros

Servicio	Llamada a sistema	Flags a revisar
Crear/Abrir fichero	open, creat	O_TRUNC, O_CREAT
Leer de / escribir en fichero	read / write	
Modificar ptr lectura/escritura	lseek	SEEK_SET, SEEK_CUR, SEEK_END
Crear / eliminar enlace a fichero	link / unlink	
Cambiar permisos de un fichero	chmod	
Cambiar propietario/grupo de un fichero	chown/chgrp	
Obtener información del Inodo	stat, lstat, fstat	



5.27

## Ejemplo: Acceso aleatorio a fichero

### ■ Qué hace el siguiente fragmento de código?

```
fd = open("abc.txt", O_RDONLY);
while (read(fd, &c, 1) > 0) {
    write(1, &c, 1);
    lseek(fd, 4, SEEK_CUR);
}
```



Encontraréis este código en: ejemplo1.c

### ■ Y este otro?

```
fd = open("abc.txt", O_RDONLY);
size = lseek(fd, 0, SEEK_END);
printf("%d\n", size);
```



Encontraréis este código en: ejemplo2.c

5.28

## Operaciones sobre Directorios (S1)

- Crear/Borrar un directorio
- Cambiar/devolver el directorio de trabajo
  - Directorio en el que se halla ubicado actualmente el usuario
- Crear/Borrar una entrada
  - Al crearse/eliminarse un archivo o subdirectorio
- Buscar una entrada
  - Localizar en el directorio la entrada correspondiente a un fichero
- Actualizar entrada
  - Al cambiar algún atributo de un archivo o subdirectorio
- Enumerar entradas
  - Permite obtener una lista de todos los archivos o subdirectorios dentro del directorio
- Leer una entrada

5.29

1. Organización del disco
2. Asignación de espacio
3. Gestión espacio libre

## ORGANIZACIÓN Y GESTIÓN DEL ESPACIO EN DISCO

5.30

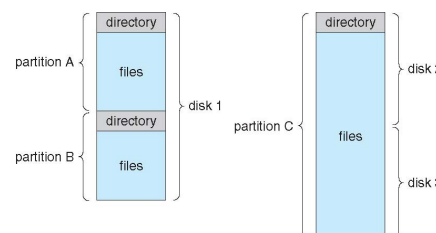
## Particiones

- Tipos de particiones
- Acceso a una partición
- Contenido de una partición
  - Metadatos
  - Datos
- Acceso a disco

5.31

## Particiones

- Partición o volumen o sistema de ficheros
  - Conjunto de sectores consecutivos al que se le asigna un identificador único y, por tanto, identidad propia para que sean gestionados por el SO como una entidad lógica independiente
    - C:, D: (Windows); /dev/hda1, /dev/hda2 (UNIX)
  - Cada partición tiene su propia estructura de directorios y ficheros independiente de las otras particiones
- Un disco puede estar sub-dividido en varias particiones
- Una partición puede tener asociados varios discos físicos
- Al formatear se está creando la estructura mínima necesaria para gestionar el sistema de ficheros

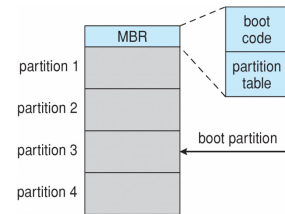


5.32



## Tipos de Particiones

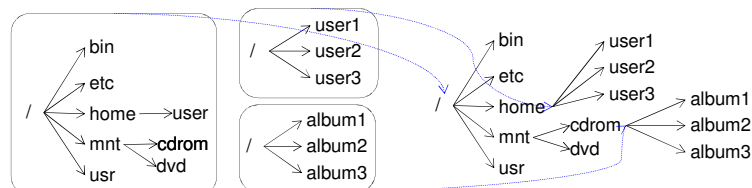
- **Master Boot Record (MBR)**
  - Es una partición sólo para gestión
  - Primer sector (512 bytes) del disco
  - Reservado para el cargador de arranque del SO (bootloader) y tabla de particiones
- **Partición Primaria**
  - Se utiliza para arrancar un sistema operativo, aunque no esté instalado en la misma partición
  - Pueden haber hasta 4 particiones primarias por disco
    - ▶ Limitación establecida por el MBR
- **Partición Extendida**
  - Puede ser subdividida en varias particiones lógicas
  - Es un “contenedor” de particiones lógicas
- **Partición Lógica (unidad lógica)**
  - Sub-parte (o totalidad) de una partición extendida
- **Espacio Swap**
  - Puede manejarse en una partición aparte (pero no siempre)



5.33

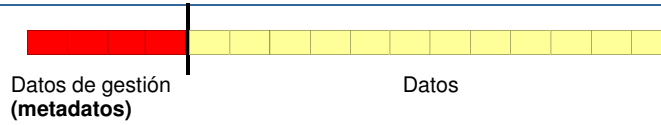
## Acceso a Particiones

- 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 directorio del SF (punto de montaje)
  - **Montar:** incluir el dispositivo (la partición) en el SF que maneja el SO para que sea accesible a través de un directorio (punto de montaje)
    - ▶ Mount (para montar), umount (para desmontar)
      - # mount -t ext2 /dev/hda1 /home
      - # umount /dev/hda1



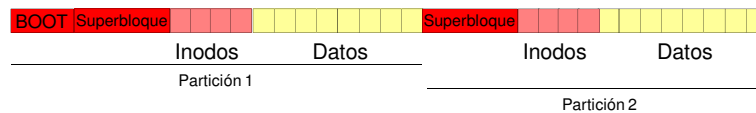
5.34

## Contenido de una partición : Metadatos



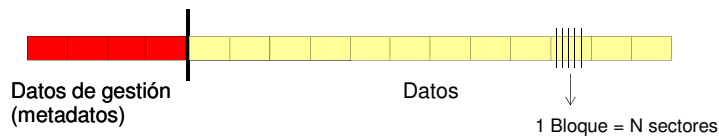
### ■ Metadatos

- Sector de Arranque (BOOT)
  - ▶ Información básica para arrancar el SO instalado en la partición
- Superbloque
  - ▶ Formato del SF (tamaño bloque, #inodos (si el SF los utiliza), #bloques de datos, ...)
  - ▶ Qué bloques se han asignado a cada fichero
  - ▶ Qué bloques no están siendo utilizados (Lista de bloques libres, inodos libres, ...)
- Ejemplo: 2 particiones de sistemas de ficheros basados en Inodos:



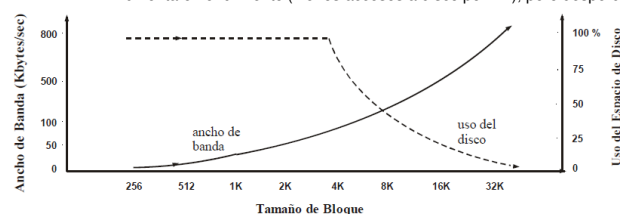
5.35

## Contenido de una partición : Datos



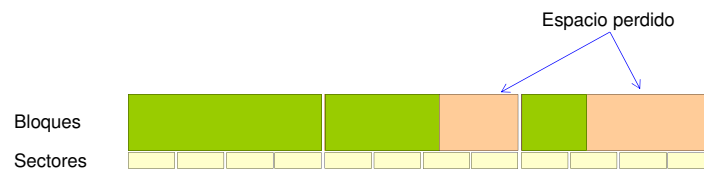
### ■ Datos

- Información organizada en **bloques**
- **Sector**: unidad de transferencia (definida por el Hw)
- **Bloque**: unidad de asignación (definido por el SO)
- qué tamaño definimos? (Fijo/Variable, Grandes/Pequeños)
  - ▶ Bloques Pequeños
    - Aprovecha mejor el espacio, pero hay que hacer muchos accesos
  - ▶ Bloques Grandes
    - Aumenta el rendimiento (menos accesos a disco por KB), pero desperdicia espacio



## Datos: Bloques de tamaño fijo

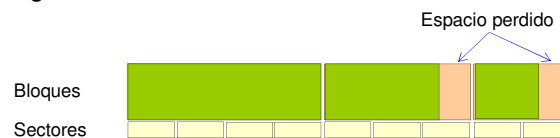
- Todos los bloques tienen el mismo tamaño
  - Muy sencillo de implementar
  - Compromiso en el tamaño de bloque
    - Eficiencia
    - Fragmentación interna



5.37

## Datos: Bloques de tamaño variable

- Bloques sin compartir sectores
  - Fragmentación interna



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



5.38

## Planificación de acceso a disco

- Coste (en tiempo) de acceso a disco
  - $T_{\text{total}} = T_{\text{posicionamiento}} + T_{\text{espera}} + T_{\text{transferencia}}$
- Existen algoritmos para minimizar el coste de acceso
- Algunos de ellos son:
  - First Come First Served
    - ▶ Simple y justo
    - ▶ No optimiza los accesos al disco
  - Shortest Seek Time First ( SSTF )
    - ▶ Sirve primero las peticiones más cercanas
    - ▶ Puede provocar inanición
  - 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.

5.39

## Gestión del espacio de disco

- Gestión del espacio ocupado
  - Asignación contigua
  - Asignación enlazada en tabla (FAT)
  - Asignación indexada multinivel
- Gestión del espacio libre
  - Mapa bits
  - Lista recursos libres

5.40

## Asignación de espacio

- 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?
- ... o en forma de bloques remotos
- Diversos mecanismos de asignación, pero nos centraremos en:
  - Asignación contigua
  - Asignación enlazada en tabla (FAT)
  - Asignación indexada multinivel

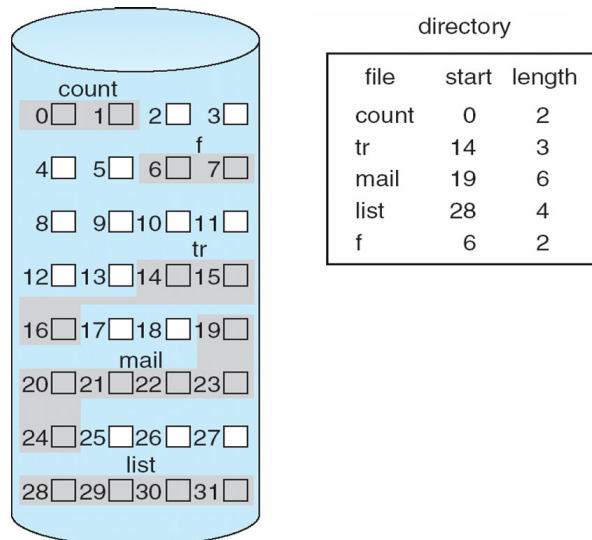
5.41

## Asignación Contigua

- Todos los bloques del archivo se asignan de manera consecutiva
  - CDROM, DVDs, ...
- 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 (determinar el tamaño a priori)

5.42

## Asignación Contigua



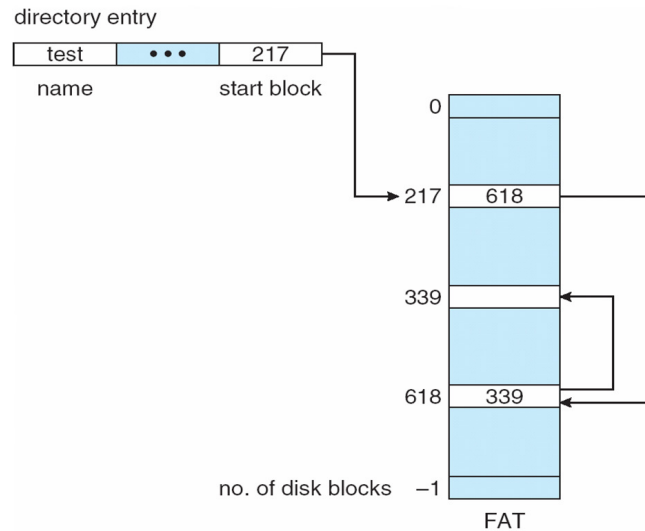
5.43

## Asignación Encadenada en Tabla (FAT)

- 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)
- Desventajas:
  - Problemas con discos grandes (tabla grande)

5.44

## Asignación Encadenada en Tabla (FAT)



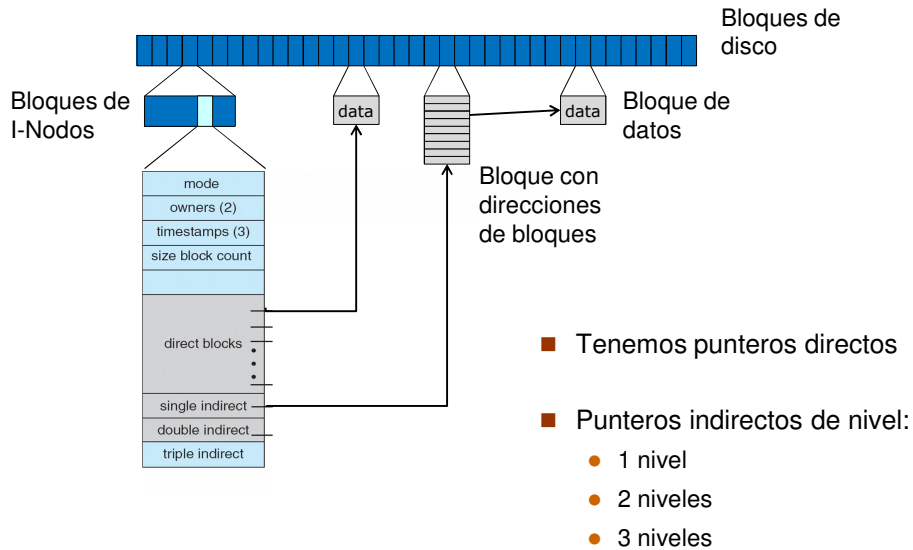
5.45

## Asignación Indexada Multinivel

- Se crea una estructura jerárquica de bloques índice
  - Cada apuntador permite acceder a un bloque de datos, pero...
- ...en el bloque índice existen algunos apuntadores indirectos
  - apuntan a nuevos bloques índices
- Ventajas:
  - Muy pocos accesos incluso en ficheros grandes
  - Poca pérdida de espacio en ficheros pequeños
- Inconvenientes:
  - Añadir o borrar datos que no están al final del fichero

5.46

## Asignación Indexada Multinivel (I-Nodos)

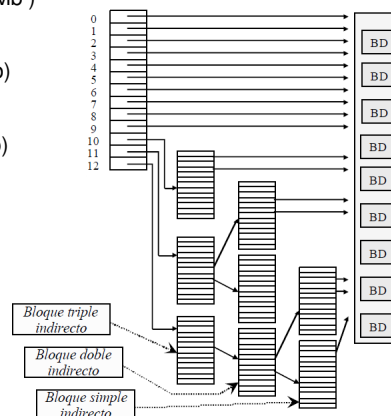


5.47

## Asignación Indexada Multinivel (I-Nodos)

### ■ Índices a los bloques de datos (1/4 Kb)

- 10 índices directos
  - ▶ ( 10 bloques = 10/40Kb )
- 1 índice indirecto
  - ▶ ( 256/1024 bloques = 256Kb/4Mb )
- 1 índice indirecto doble
  - ▶ ( 65K/1M bloques = 65Mb/4 Gb )
- 1 índice triple indirecto
  - ▶ ( 16M/1G bloques = 16Gb/4 Tb )



5.48



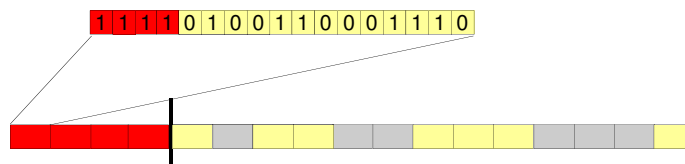
## Gestión del espacio libre

- El S.F. tiene una estructura que gestiona el espacio libre del dispositivo (Tabla de asignación de disco)
  - Nos indica qué zonas del disco se encuentran libres
- Diversos mecanismos, pero nos centraremos en:
  - Mapa de bits
  - Lista de recursos libres

5.49

## Mapa de bits (bitmap)

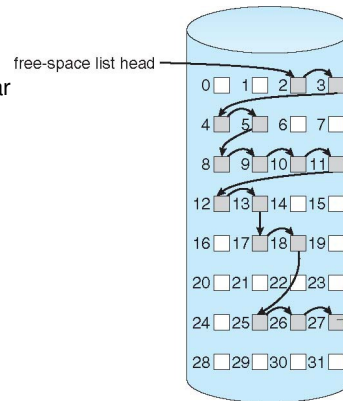
- Contiene un bit por cada bloque del disco
  - 0 = bloque libre
  - 1 = bloque ocupado
- Ventajas:
  - Relativamente fácil encontrar un bloque libre o un grupo contiguo de bloques libres
  - Ocupa poco espacio (se puede tener en memoria)



5.50

## Lista de recursos libres

- Puede ser por bloques o secciones (grupo de bloques consecutivos)
- Apuntamos al primer bloque de un grupo de bloques libres
- En ese bloque se guarda
  - Si es por bloques
    - Un puntero al siguiente bloque libre
  - Si es por secciones
    - Número de bloques libres consecutivos
    - Dirección del siguiente grupo
- Ventajas
  - No requiere espacio adicional para guardar qué bloques están libres
- Problemas
  - La gestión de la lista puede ser ineficiente
    - si entran en juego muchos bloques
  - Fragmentación



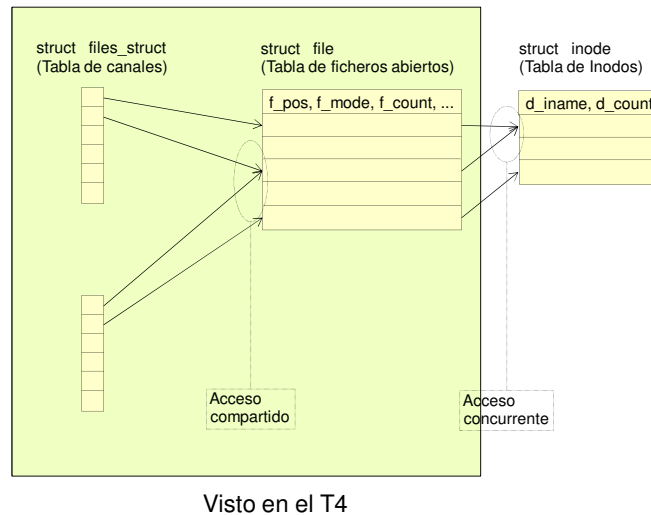
5.51

1. Estructuras Linux/UNIX
2. Relación llamadas a sistema y estructuras de datos (Linux)
3. Ejemplos

## ACCESO A FICHEROS

5.52

## Estructuras Linux/UNIX



5.53

## Estructuras Linux/UNIX

- **Tabla de ficheros abiertos**
  - Información sobre cada obertura (y sus accesos) sobre un fichero
    - ▶ La llamada "lseek" sólo implica actualizar el desplazamiento de la entrada correspondiente en esta tabla. NO implica acceso a disco
- **Tabla de Inodos**
  - Mantiene en memoria 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
- **Pros y Contras en general...**
  - Pros: información en memoria → NO acceder al disco
  - Contras: consumo de memoria

5.54

## Relación llamadas a sistema y estructuras de datos (Linux)

- Las llamadas a sistema modifican las estructuras de datos y pueden generar accesos a disco (o no)
- Open
  - Implica acceder a todos los bloques de datos de los directorios implicados y a todos los bloques que contengan los i-nodos necesarios
  - El último acceso a disco es el del i-nodo del fichero que abrimos
  - Modifica la tabla de canales, la tabla de ficheros abiertos, y, potencialmente, la tabla de dentry's y de inodos (solo si es la primera vez que accedemos al fichero)

5.55

## Relación llamadas a sistema y estructuras de datos (Linux)

- Read
  - Implica leer los bloques de datos (suponiendo que no están en la buffer cache).
  - Puede suponer 1 o N accesos a disco dependiendo del tamaño de los datos que queremos leer
  - También influye en el número de accesos si los bloques de datos están apuntados por índices directos (en el i-nodo) o índices indirectos (en bloques de datos)
- Objetivo: Saber calcular que accesos a disco y de que tipo generan las llamadas a sistema de gestión de E/S. Saber qué modificaciones generan en las tablas de gestión de E/S las llamadas a sistema de E/S.

5.56

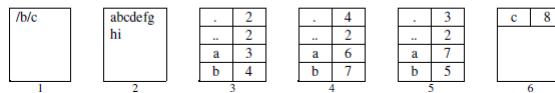
## Ejemplo

**Inodes**

tipus	dir	dir	dir	link	dat	dat
blocs de dades	3	5	4,6	1		2
núm i-node	2	3	4	5	6	7

L'inode 2 es l'arrel

**Blocs de dades**

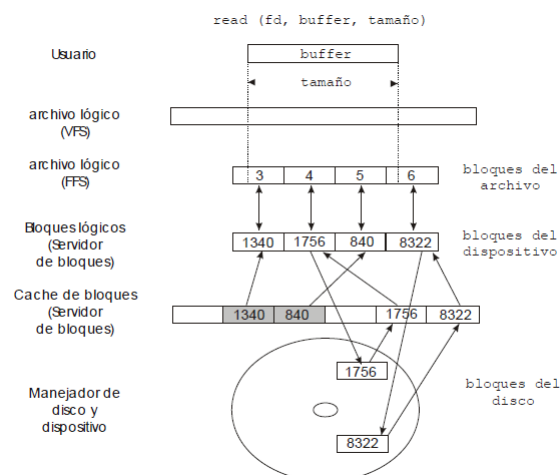


```
fd = open("/a/b", O_RDONLY);
read(fd, buffer, 100);
fd = open("/b/d", O_WRONLY);
```

Open(/a/b..) → i-nodo 2 + bloque 3 + i-nodo 3 + bloque 5 + i-nodo 5 + bloque 1 + i-nodo 2 + bloque 3 + i-nodo 4 + bloque 4 + bloque 6 + i-nodo 8 → 12 accesos a disco

5.57

## Ejemplo de Flujo de datos



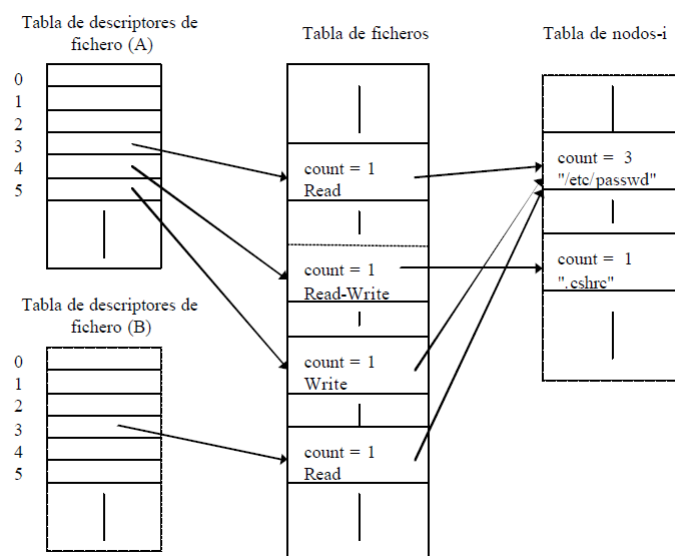
5.58

## Ejemplo del uso de las tablas internas

- Un proceso "A" abre:
  - `fd1 = open("/etc/passwd", O_RDONLY);`
  - `fd2 = open(".cshrc", O_RDWR);`
  - `fd3 = open("/etc/passwd", O_WRONLY);`
- Un proceso "B" abre:
  - `fd1 = open("/etc/passwd", O_RDONLY);`

5.59

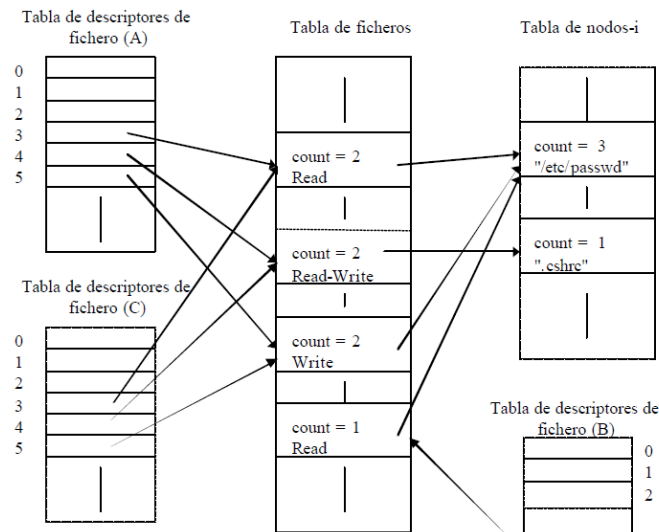
## Ejemplo del uso de las tablas internas



5.60

## Ejemplo del uso de las tablas internas

- El proceso "A" crea un proceso hijo, "C": como afecta un fork



5.61

1. RAID
2. Journaling

## FIABILIDAD

5.62

## RAID

- Redundant Array of Inexpensive Disks
  - Su objetivo es proporcionar robustez y/o mejorar rendimiento
    - ¿Qué pasa si un disco falla?
    - ¿Cómo podemos aumentar la velocidad de transacción al disco si no podemos disponer de discos más rápidos?
- Múltiples discos que trabajan cooperativamente para ofrecer 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
- Existen diversos esquemas de configuración (implementación)
  - Se pueden combinar entre sí

5.63

## Journaling

- Las escrituras se escriben en la buffer cache y se sincronizan cada cierto tiempo
  - Problema: si el servidor sufre una anomalía antes (pérdida de corriente, ...) el sistema de ficheros puede quedar inconsistente
- Solución: Aplicar ideas de bases de datos
  - Se guarda un registro de las transacciones al disco (fichero de log)
    - Se utiliza para reconstruir las operaciones en caso de inconsistencia
  - Si el sistema se cae, las transacciones persisten en el registro y entonces se pueden completar o deshacer
  - Se puede utilizar para metadatos y/o datos
- SF con Journaling: XFS, NTFS, ReiserFS, JFS, Ext3

5.64



1. NTFS
2. SF distribuidos
3. SF en máquinas de tiempo real

## OTROS SF & ARQUITECTURAS

5.65

## NTFS – New Technology File System

- Sistema de Ficheros creado por Microsoft (Windows NT)
  - Basado en:
    - HPFS (High Performance File System) → IBM/Microsoft OS/2
    - HFS (Hierarchical File System) → Mac OS
- Disco dividido en CLUSTERS
  - Un cluster contiene varios sectores
    - Disco de más de 4GB: 1 cluster = 16 sectores
- En NTFS todo son ficheros
- Formato:

partition boot sector	Master File Table	system files	file area
-----------------------------	-------------------------	-----------------	--------------

5.66

## NTFS: Partition Boot Sector

### ■ Partition Boot Sector

- 16 sectores reservados al principio de la partición

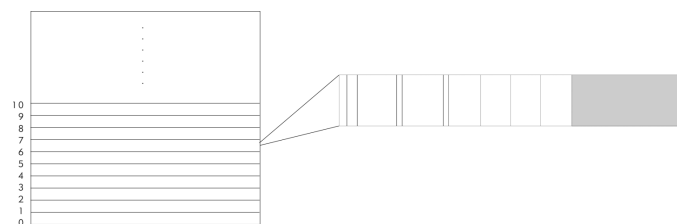
Byte offset	Longitud del campo	Nombre
0x00	3 bytes	Jump
0x03	LONGLONG	OEM ID (S/N)
0x0B	25 bytes	BPB (info partición)
0x24	48 bytes	EBPB (@MTF)
0x54	426 bytes	Bootstrap code
0x01FE	WORD	End of sector marker

5.67

## NTFS: Master File Table

### ■ Master File Table (MFT)

- Una parte de los metadatos (system files) de una partición NTFS
- Almacena una lista de RECORDS que contienen ATRIBUTOS



### ■ Espacio reservado: MFT-zone

- 12% del espacio del disco duro
- Se incrementa y se reduce de forma dinámica
  - ▶ Crecer: doblar el espacio actual
  - ▶ Reducir: mitad del espacio actual

5.68

## NTFS: Records & Atributos

### ■ Records

- Estructura que guarda información de un fichero agrupando atributos
  - Normalmente relación 1 record por fichero
  - A veces se necesitan más records por fichero

### ■ Atributos

- Información relacionada a un fichero
  - Desde bits (ej: permisos de lectura/escritura) hasta exabytes (ej: video)
- No hay distinción entre los datos de un fichero y los atributos que lo describen

Header	Información sobre el fichero y el record: -timestamp, link count, journaling
Nombre del fichero o directorio	Nombre "regular" del fichero -puede haber mas de uno
Descriptor de seguridad	Describe quien lo posee y quien tiene acceso
Datos o índices	Datos del fichero Índices a los datos del fichero

5.69

## NTFS: Records & Atributos

### ■ 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 atributos de datos NO residentes
    - Record con índices
    - Índices que apuntan a bloques de información
    - Si hay tantos índices que no caben en un record de la MFT, se utiliza un puntero a otro record de la MFT que contendrá índices

5.70

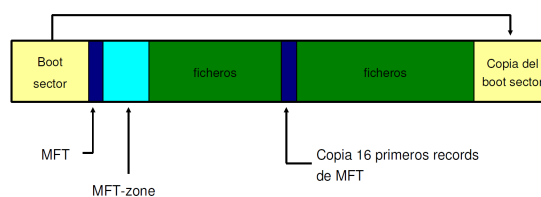
## NTFS: System Files (metadatos)

- Los 16 primeros RECORDS de la MFT están reservados para metadatos
  - Del 0 al 10:
    - ▶ \$MFT: puntero a MFT
    - ▶ \$MFTmirr: copia del MFT
    - ▶ \$LogFile: fichero de journaling
    - ▶ \$Volume: Información sobre el volumen
    - ▶ \$AttrDef: listado de atributos del volumen
    - ▶ \$.: Directorio raíz
    - ▶ \$Bitmap: Bitmap del espacio libre
    - ▶ \$Boot: Sector de boot
    - ▶ \$BadClus: lista de clusters con errores
    - ▶ \$Secure: base de datos de ACLs
    - ▶ \$Upcase: Relación entre ficheros con nombre en mayúsculas y minúsculas
    - ▶ \$Extend: Extensión de la información
  - Del 11 al 15 para futuros metadatos

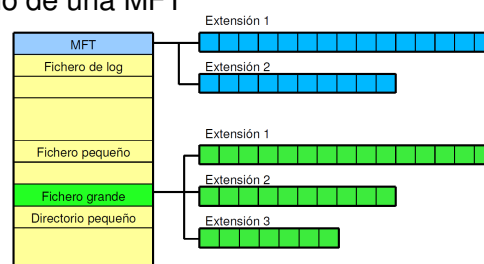
5.71

## NTFS: Visión de una Partición

### ■ Ejemplo de una partición



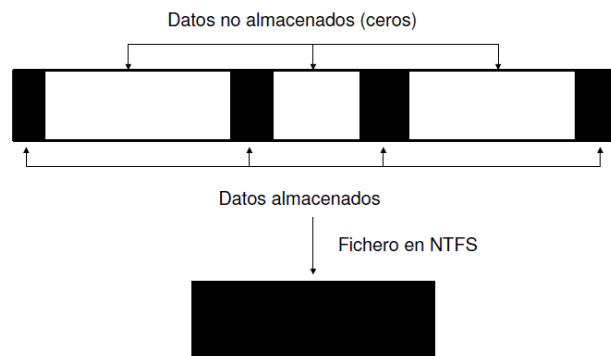
### ■ Ejemplo de una MFT



5.72

## NTFS: Ficheros Dispersos

- Ficheros Dispersos
  - Ahorra espacio en disco
  - No se almacenan en el disco las largas cadenas 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



5.73

## 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

5.74

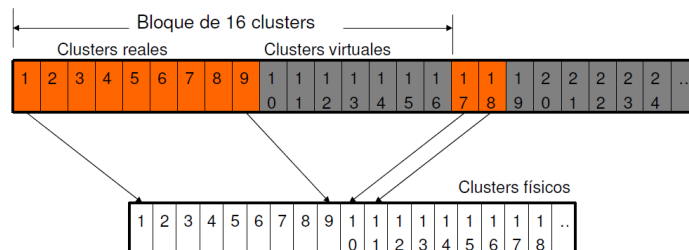
## NTFS: Encriptación

- A nivel de fichero y transparente al usuario
- Servicio de W2K – Encrypted File System (EFS)
- API:
  - EncryptFile
  - DecryptFile
- Algoritmo: RSA

5.75

## NTFS: Compresión

- El usuario puede activar la compresión de ficheros/directorios/disco
  - A partir de ese momento se comprime (modificar) descomprime (acceder) automáticamente
    - ▶ Transparente al usuario
- Cada fichero es comprimido en grupos de 16 clusters
  - Cada grupo se comprime/descomprime de forma aislada de los demás (son independientes)



5.76

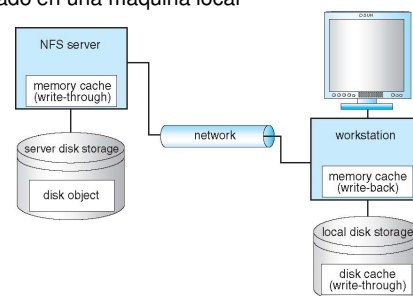
## NTFS: Alternate Data Streams

- Alternate Data Streams (ADS)
  - Conjuntos de datos alternativos que se organizan en streams adicionales al conjunto de datos que componen un fichero
    - ▶ Ej: Resumen de propiedades de un fichero mp3 (autor, nombre, etc)
  - Destinado para...
    - ▶ Actuar como servidor de ficheros para clientes Mac
    - ▶ Resumen de propiedades/características del fichero
    - ▶ Rastreo de volúmenes
- Originalmente implementado para compatibilidad con Mac OS
- Peligro de uso incorrecto para atacar sistemas
  - Se pueden insertar datos sin que sean visibles en el fichero de forma directa
    - ▶ Ej: un fichero de texto (.txt) que sólo contiene 10 bytes de texto, pero además tiene un ADS que contiene un ejecutable de 20 Megs
      - El SO sólo mostraría los 10 bytes de texto, pero podría llegar a ejecutar el contenido del ADS
  - Solucionado en la versión de NTFS de Windows 7

5.77

## SF Distribuidos

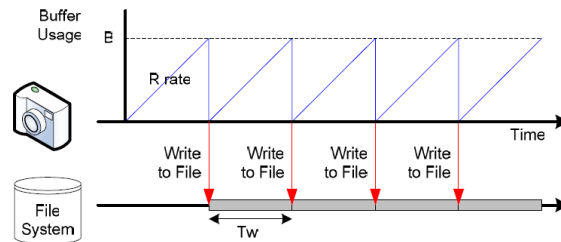
- Network File System
  - Implementación y especificación de un sistema software para acceder a sistemas de ficheros remotos mediante LAN/WAN
    - ▶ Desarrollado inicialmente por Sun (Solaris & SunOS)
  - Permite compartir sistemas de ficheros independientes entre distintas máquinas de forma transparente
    - ▶ Un directorio remoto se monta en un directorio del SF local
    - ▶ Se tiene que proporcionar el nombre del host de la máquina remota para poder montar el SF
  - Aparte de limitaciones por protección, cualquier SF (o directorio de ese SF) de una máquina remota puede ser montado en una máquina local
- Orientado hacia...
  - Cloud Computing & large scale distributed systems (Ej: Google, Yahoo, Facebook)
    - ▶ Google File System
    - ▶ Hadoop Distributed File System



5.78

## SF en máquinas de tiempo real

- Limitación de tiempo para transferencias a “disco”
- El uso de otros dispositivos (memorias flash) que se manejan como “discos” pueden influir en la limitación temporal



5.79

## Necesidades del SF

- Traducir los accesos desde la solicitud de la interfaz de usuario/aplicación a ...
- ...la implementación del propio sistema de ficheros que se está utilizando y ...
- ... finalmente a los drivers que controlan los dispositivos de almacenamiento

5.80



## Arquitectura de Capas del SF

- Logical I/O

- ▶ Proporciona la abstracción fichero para realizar las E/S solicitadas

- El SF se encarga de ofrecerla

- Basic I/O Supervisor

- ▶ Se encarga de gestionar (controlar el inicio y fin) de la E/S sobre ficheros

- Elige el dispositivo basado en el fichero seleccionado

- ▶ Se encarga de la planificación de los accesos y su optimización

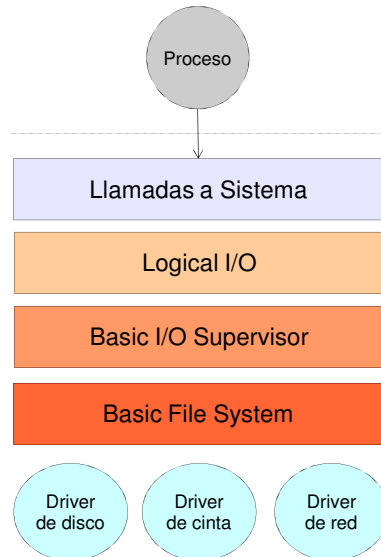
- El SO se encarga de ofrecerla

- Basic File System

- ▶ Se encarga de **mover bloques** entre la memoria y los dispositivos

- ▶ No sabe lo que es un fichero

- El SO se encarga de ofrecerla



5.81