

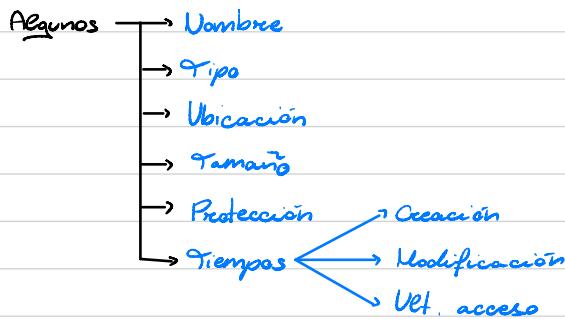
## Esquema Gestión de Archivos Tema 4

### Interfaz

Objetivos → Optimizar caminos frecuentes menos llamadas al sistema  
 ↘ Favorecer accesos secuenciales

### Atributos de archivo / metadatos

Def → Datos que mantiene el SO para describir el archivo  
 ↘ Implementar abstracción



### Sendo - Sist. de Archivos

Objetivo: acceder a datos o recursos del SO como si fueran archivos aunque no lo sean

Ejemplos: /proc, /dev, /devfs

Búfer de E/S → Nivel → De SO → De bibliotecas

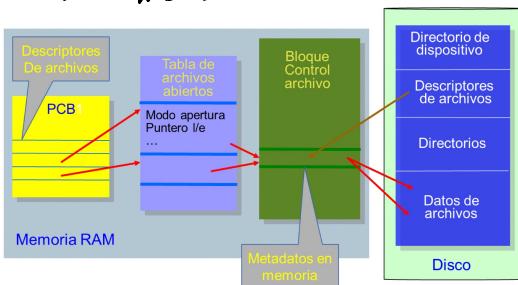
Def. Mem. intermedia temp. para mejorar rendimiento

Objetivos → Acoplar vel. CPU y disco  
 → Igualar tamaño datos transfr.  
 → Minimizar tiempo de bloq.  
 ↘ Desacoplar E/S de gest. mem.

Caché de disco: búferes E/S disco + algorit. de reemplazo

## Diseño

### Gestión de memoria



### Almacenamiento secundario

#### Asignación de ficheros

- Continuar
  - Encadenada/Enlazada
  - Indexada
- Y *Cómo guardar ficheros*

#### Gestión espacio libre

- Tabla de bits
- Porciones encadenadas
- Indexación
- lista de bloques libres

## Asignación de ficheros

→ Continua: todo el fichero guardado solo

Func. → Ds de inicio y conquistar

Dd / tam bloque = Cociente CC , Resto(R)

Bloque = C

Desplazamiento = R

Ventajas : - Sencillo

- Acceso directo y rápido

Desventajas: - Asig. dinámica  $\Rightarrow$  fragmentación

- Archivos no pueden aumentar

→ Enlazada cada bloque apunta al siguiente

Func → Punteros bloque inicio y final

Cálculos igual pero acceso secuencial

## Ventajas - Simple

- Creación dinámica (No fragmentación)

Desventajas: - Acceso aleatorio inef.

- Punteros consumen espacio Mejorar: clústeres
  - Problema secundario por perdida de punteros

Solución: cinta doble enarcada  $\rightarrow$  sobrecarga >

Variante FAT (File Allocation Table) en pendrives y discos

Función → Tabla FAT con 1 entrada / bloque e indexada por n bloques

Copia en caché para más eficacia

→ Indexar

Func → 1 o varios bloques/archivo como tablas de indices

Ventajas - Acceso aleatorio

- Acceso dinámico  $\Rightarrow$  No fragmentación

Desventajas - Desperdicio de bloques

- Tamaño bloque indices

inode Cada archivo en disco  $\Rightarrow$  1 estructura "inode" con propiedades

### Directorio en Linux

Estructura lista, cada entrada

<inode> <long. registro> <long. nombre> <tipo> <nombre>

### Área de intercambio

Uso: depende del algoritmo  $\rightarrow$  intercambio  $\Rightarrow$  Procesos completos

Lugar  $\rightarrow$  Página => Página sucia

Lugar  $\rightarrow$  Archivo (Windows)  $\Rightarrow$  Fácil pero fragmentación

Partición de disco (Linux)  $\Rightarrow$  Asig. continua  $\Rightarrow$  Eficiente

### RAID (Redundant Array of Inexpensive Disks)

Varios discos funcionan como uno. Cada bloque  $\Rightarrow$  subbloques, 1 por disco

Ventajas:

- Mejor tolerancia a fallos

- Transferencia en paralelo.

Esquemas  $\rightarrow$  RAID 0 - Mirroring

$\rightarrow$  RAID 5 - Distrib. con paridad

### Planificación de disco

Objetivo: Reorganizar peticiones de SO para reducir tiempo.

Ejemplos:

- $\rightarrow$  noop
- $\rightarrow$  deadline
- $\rightarrow$  cfq

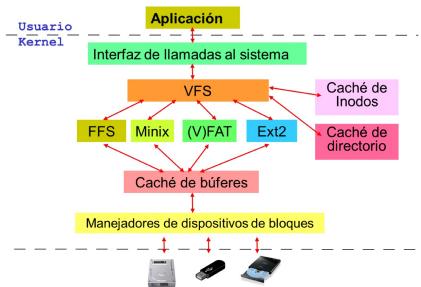
## Implementación $\Rightarrow$ Virtual File System (VFS)

Def  $\rightarrow$  Capa de software entre usuario y SAs que da una interfaz única e indep. para todos estos

SA soportados:

- Basados en disco: gestionan partición de disco **Ext2**
- Basados en red: SA en red **NFS / AFS**
- Especiales: interfaz a otro objeto / proc

## Estructura



Modelo de archivo común  $\Rightarrow$  Capa de representar todos los tipos

SA traducen org. física a modelo común

$\rightarrow$  Datos + métodos para operar

$\hookrightarrow$  El archivo tiene punteros a estos

Objetos usados

Obl.: Superbloque: SA montado  $\Rightarrow$  Directorio de disp.

Obl.: inode: info de un SA específico

Obl.: archivo: interacción proceso-archivo abierto

Obl.: dentry: enlace entrada direc.-archivo

Objeto inodo metadatos de un archivo + datos de gestión caché de inodos

Campos:

- i-est ptre. existencia inodos
- i-dentrey ptre. existencia dentrey
- i-count contador de uso
- i-ops operaciones sobre inodos
- u info sist. archivos

Objeto archivo se crea cuando se abre el archivo

Estructura "file" No tiene imagen en disco

Campos:

- f-pos ptre l/e
- f-mode r/w, permisos
- f-op operaciones de archivos
- f-count contador de uso

Traducción de nombres

Esp. nombre SO → 2 inodos

Esp. nombre usuario → "pathname"

El Kernel establece las correspondencias

→ proceso lento y complejo

Solución: Objeto dentrey

asocia componente del pathname con inodo

VFS lee entrada a directorio ⇒ transforma de: <sup>dentrey</sup>

Campos d-name - nombre

d-inode - inodo

d-mounts - dentrey de la raíz de SA montado

d-cores - dentrey de pto. de montaje