TEMA-4-GESTION-DE-ARCHIVOS.pdf

mrg23

Sistemas Operativos

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

TEMA 4: GESTIÓN DE ARCHIVOS

1. INTRODUCCIÓN. CONCEPTOS DE ARCHIVOS

Colección de información relacionada y almacenada en un dispositivo de almacenamiento secundario.
Espacio de direcciones lógicas contiguas.
Estructura interna (lógica)

- Secuencia de bytes: el tipo del archivo determina su estructura (texto → caracteres,líneas y páginas, código fuente secuencia de subrutinas y funciones).
- Secuencia de registros de longitud fija
- Secuencia de registros de longitud variable.

Tipos de archivos: regulares, directorios, de dispositivo Formas de acceso: secuencial, aleatorio, otros.

Atributos (metadatos)

- Nombre: única información en formato legible
- Tipo: cuando el sistema soporte diferentes tipos
- Localización: información sobre su localización en el dispositivo
- Tamaño: tamaño actual del archivo
- Protección: controla quién puede leer, escribir y ejecutar
- Tiempo, fecha e identificación del usuario: necesario para protección, seguridad y monitorización.

Operaciones sobre archivos

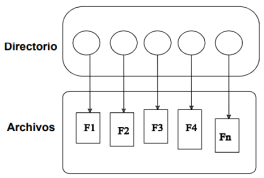
- Gestión:
  - Crear
  - Borrar
  - Renombrar
  - Copiar
  - Establecer y obtener atributos

Procesamiento:

- Abrir y Cerrar
- Leer
- Escribir (modificar, insertar, borrar información)

2. ESTRUCTURA DE DIRECTORIOS

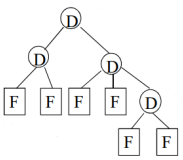
Colección de nodos conteniendo información acerca de todos los archivos → Organización



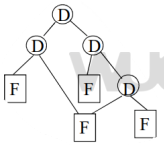
Tanto la estructura de directorios como los archivos residen en el almacenamiento secundario.

La organización (lógica) de los directorios debe proporcionar:

- Eficiencia: localización rápida de un archivo.
- Denominación: adecuada a los usuarios.
  - Dos usuarios pueden tener el mismo nombre para diferentes archivos
  - El mismo archivo puede tener varios nombres.
- Agrupación: agrupar los archivos de forma lógica según sus propiedades (Ej. todos los programas en C)
- En árbol
  - Necesidad de búsquedas eficientes
  - Posibilidad de agrupación
  - Directorio actual (de trabajo)
  - Nombres de camino absolutos y relativos



- En grafo
  - Compartición de subdirectorios y archivos
  - Más flexibles y complejos



3. PROTECCIÓN Y CONSISTENCIA

a. PROTECCIÓN

Básicamente consiste en proporcionar un acceso controlado a los archivos

- lo que puede hacerse
- por quién

Tipos de acceso

- Leer
- Escribir
- Ejecutar
- Añadir
- Borrar
- Listar

LISTAS Y GRUPOS DE ACCESO

Principal solución a la protección: hacer el acceso dependiente del identificativo del usuario.
Las listas de acceso de usuarios individuales tienen el problema de la longitud.
Solución con clases de usuario:

- propietario.
- grupo.
- público

Propuesta alternativa: Asociar un password con el archivo. Problemas:

- Recordar todos.
- Si solo se asocia un password acceso total o ninguno.

b. CONSISTENCIA

Especifican cuándo las modificaciones de datos por un usuario se observan por otros usuarios.
Ejemplos:

1. Semántica de Unix
   - La escritura en un archivo es directamente observable.
   - Existe un modo para que los usuarios compartan el puntero actual de posicionamiento en un archivo.
2. Semánticas de sesión (Sistema de archivos de Andrew)
   - La escritura en un archivo no es directamente observable.
   - Cuando un archivo se cierra, sus cambios sólo se observan en sesiones posteriores.
3. Archivos inmutables
   - Cuando un archivo se declara como compartido, no se puede modificar.

4. ESTRUCTURA DEL SISTEMA DE ARCHIVOS

FUNCIONES BÁSICAS DEL SISTEMA DE ARCHIVOS

Tener conocimiento de todos los archivos del sistema.
Controlar la compartición y forzar la protección de archivos.
Gestionar el espacio del sistema de archivos.

- Asignación/liberación del espacio en disco.

Traducir las direcciones lógicas del archivo en direcciones físicas del disco.

- Los usuarios especifican las partes que quieren leer/escribir en términos de direcciones lógicas relativas al archivo.

Un sistema de archivos posee dos problemas de diseño diferentes:

1. Definir cómo debe ver el usuario el sistema de archivos
   - definir un archivo y sus atributos
   - definir las operaciones permitidas sobre un archivo
   - definir la estructura de directorios
2. Definir los algoritmos y estructuras de datos que deben crearse para establecer la correspondencia entre el sistema de archivos lógico y los dispositivos físicos donde se almacenan.

Organización en niveles (capas)

- Por eficiencia, el SO mantiene una tabla indexada (por descriptor de archivo) de archivos abiertos.
- Bloque de control de archivo: estructura con información de un archivo en uso

5. MÉTODOS DE ASIGNACIÓN DE ESPACIO

a. CONTIGUO

Cada archivo ocupa un conjunto de bloques contiguos en disco

Ventajas:

- Sencillo: solo necesita la localización de comienzo (nº de bloque) y la longitud
- Buenos tanto el acceso secuencial como el directo

Desventajas:

- No se conoce inicialmente el tamaño
- Derroche de espacio (problema de la asignación dinámica → fragmentación externa)
- Los archivos no pueden crecer, a no ser que se realice compactación → ineficiente.

Asociación lógica a física:

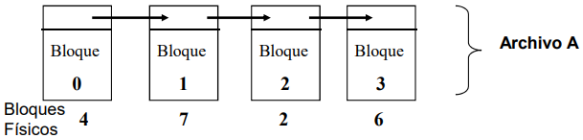
Supongamos que los bloques de disco son de 512 bytes:

Dirección lógica (DL)/512 C(cociente), R(resto)

- Bloque a acceder = C + dirección de comienzo
- Desplazamiento en el bloque = R

b. NO CONTIGUO - ENLAZADO

Cada archivo es una lista enlazada de bloques de disco. Los bloques pueden estar dispersos en el disco.



Ventajas:

- Evita la fragmentación externa.
- El archivo puede crecer dinámicamente cuando hay bloques de disco libres → no es necesario compactar.
- Basta almacenar el puntero al primer bloque del archivo.

Desventajas:

- El acceso directo no es efectivo (si el secuencial)
- Espacio requerido para los punteros de enlace. Solución: agrupaciones de bloques (clusters).
- Seguridad por la pérdida de punteros. Solución: lista doblemente enlazada (overhead)

Asociación lógica a física: (dirección = 1 byte)

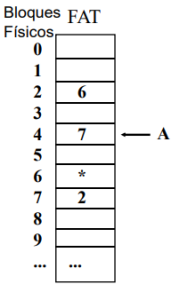
Dirección lógica (DL)/511 C(cociente), R(resto)

- Bloque a acceder = C-ésimo
- Desplazamiento en el bloque = R + 1

TABLA DE ASIGNACIÓN DE ARCHIVOS (FAT)

Variación del método enlazado (Windows y OS/2)

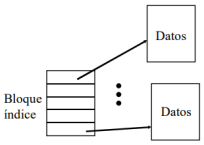
– Reserva una sección del disco al comienzo de la partición para la FAT.
– Contiene una entrada por cada bloque del disco y está indexada por número de bloque de disco.
– Simple y eficiente siempre que esté en caché
– Para localizar un bloque solo se necesita leer en la FAT se optimiza el acceso directo.
– Problema: pérdida de punteros doble copia de la FAT



c. NO CONTIGUO - INDEXADO

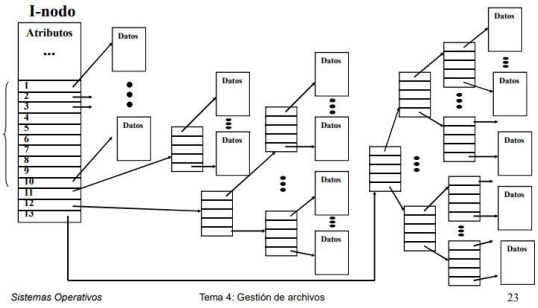
Todos los punteros a los bloques están juntos en una localización concreta: bloque índice.
El directorio tiene la localización a este bloque índice y cada archivo tiene asociado su propio bloque índice.
Para leer el i-ésimo bloque buscamos el puntero en la i-ésima entrada del bloque índice

Ventajas
- Buen acceso directo
- No produce fragmentación externa



Desventajas
- Posible desperdicio de espacio en los bloques índices
- Tamaño del bloque índice. Soluciones:
  (a) Bloques índices enlazados
  (b) Bloques índices multinivel
    - Problema: acceso a disco necesario para recuperar la dirección del bloque para cada nivel de indexación
    - Solución: mantener algunos bloques índices en memoria principal
  (c) Esquema combinado (Unix)

Unix (s5fs):



6. GESTIÓN DE ESPACIO LIBRE

El sistema mantiene una lista de los bloques que están libres: lista de espacio libre.
La FAT no necesita ningún método.
A pesar de su nombre, la lista de espacio libre tiene diferentes implementaciones:

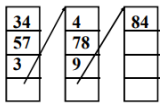
1. Mapa o Vector de Bits

10010001
11111101
11100000
11111110
00000000
11100011
11100000

– Cada bloque se representa con un bit (0-Bloque libre; 1- Bloque ocupado).
– Fácil encontrar un bloque libre o n bloques libres consecutivos. Algunas máquinas tienen instrucciones específicas.
– Fácil tener archivos en bloques contiguos – Ineficiente si no se mantiene en memoria principal.

2. Lista enlazada
   – Enlaza todos los bloques libres del disco, guarda un puntero al primer bloque en un lugar concreto
   – No derrocha espacio
   – Relativamente ineficiente → No es normal atravesar bloques vacíos.

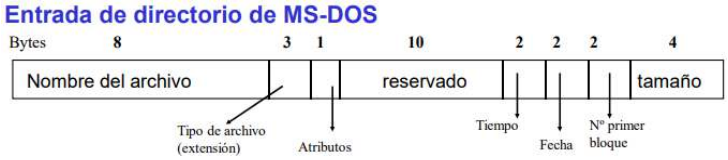
3. Lista enlazada con agrupación
   – Cada bloque de la lista almacena n-1 direcciones de bloques libres
   – Obtener muchas direcciones de bloques libres es rápido



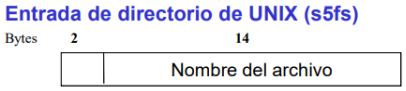
4. Cuenta – Cada entrada de la lista: una dirección de bloque libre y un contador del nº de bloques libres que le sigue

7. IMPLEMENTACIÓN DE DIRECTORIOS

Contenido de una entrada de directorio. Casos:
(a) Nombre de Archivo + Atributos + Dirección de los bloques de datos (DOS)



(b) Nombre de Archivo + Puntero a una estructura de datos que contiene toda la información relativa al archivo (UNIX)



Cuando se abre un archivo:
– El SO busca en su directorio la entrada correspondiente
– Extrae sus atributos y la localización de sus bloques de datos y los coloca en una tabla en memoria principal
– Cualquier referencia posterior usa la información de dicha tabla

Posibilidades respecto a la implementación:

1. Lista lineal
   – Sencillo de programar
   – Consume tiempo en las creaciones, búsquedas, ..., si no se utiliza una caché software
2. Tabla hash
   – Decrementa el tiempo de búsqueda
   – Dificultades:
      - Tamaño fijo de la Tabla hash
      - Dependencias de la función hash sobre el tamaño de la tabla
      - Necesita previsión para colisiones

Implementación de archivos compartidos (o enlace):

1. Enlaces simbólicos
   – Se crea una nueva entrada en el directorio, se indica que es de tipo *enlace* y se almacena el camino de acceso absoluto o relativo del archivo al cual se va a enlazar
   – Se puede usar en entornos distribuidos
   – Gran número de accesos a disco
2. Enlaces absolutos (o *hard*)
   – Se crea una nueva entrada en el directorio y se copia la dirección de la estructura de datos con la información del archivo
   – Problema al borrar los enlaces: solución → Contador de enlaces

1/6

Este número es indicativo del riesgo del producto, siendo 1/6 indicativo de menor riesgo y 6/6 de mayor riesgo.

ING BANK NV se encuentra adherido al Sistema de Garantía de Depósitos Holandés con una garantía de hasta 100.000 euros por depositante. Consulta más información en ing.es

Que te den **10 € para gastar** es una fantasía. ING lo hace realidad.

Abre la **Cuenta NoCuenta** con el código **WUOLAH10**, haz tu primer pago y llévate 10 €.

Quiero el cash

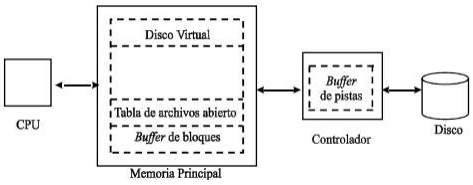
Consulta condiciones **aquí**

do your thing

8. EFICIENCIA Y RENDIMIENTO

Los discos suelen ser el principal cuello de botella del rendimiento del sistema.
La eficiencia depende de la asignación de disco y de la implementación de directorios utilizada
Para proporcionar mejor rendimiento:

1. Caché de disco: secciones de M.P. con bloques usados.
2. Discos virtuales o discos RAM: almacén temporal. Su contenido es controlado por el usuario.



9. RECUPERACIÓN

Como los archivos y directorios se mantienen tanto en MP como en disco, el sistema debe asegurar que un fallo no genere pérdida o inconsistencia de datos.
Distintas formas:

1. Comprobador de consistencia:
   – Compara los datos de la estructura de directorios con los bloques de datos en disco y trata cualquier inconsistencia.
   – Más fácil en listas enlazadas que con bloques índices
2. Usar programas del sistema para realizar copias de seguridad (backup) de los datos de disco a otros dispositivos y de recuperación de los archivos perdidos

10. ESTRUCTURA Y PLANIFICACIÓN DE DISCO

ESTRUCTURA DEL DISCO

Desde el punto de vista del SO, el disco se puede ver como un array de bloques (B0 , B1 ,...,Bn-1)
La información se referencia por una dirección formada por varias partes:
– unidad (número de dispositivo),
– superficie (o cara),
– pista, y
– sector
Existe un esquema de asociación de la dirección de un bloque lógico Bi a dirección física (pista, sector, ...).
– El área de asignación más pequeña es un bloque (1 o más sectores).
– Fragmentación interna en los bloques.

PLANIFICACIÓN DE DISCO

El SO puede mejorar el tiempo medio de servicio del disco Una petición se atiende en tres fases:

1. Posicionamiento de la cabeza en la pista o cilindro.
2. Latencia: espera a que pase el bloque deseado.
3. Transferencia de los datos entre MP y disco.

$$t^o\_total\ de\ servicio = t^o posicionamiento + t^o\_latencia + t^o\_transferencia$$

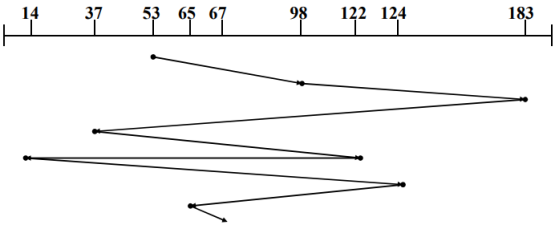
La planificación intenta minimizar el tiempo de posicionamiento distancia de posicionamiento. Si el disco está ocupado, las peticiones se encolan.
Información necesaria para una petición:

- Si la operación es de entrada o de salida
- Dirección de bloque
- Dirección de memoria a donde, o desde donde, copiar los datos a transferir
- Cantidad de información a transferir

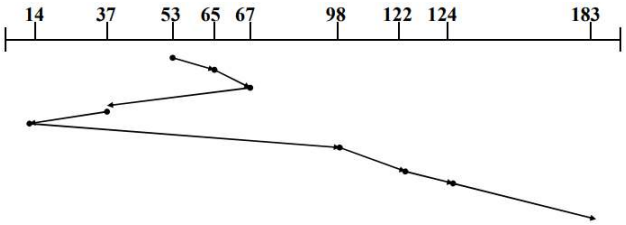
Existen distintos algoritmos de planificación de peticiones

- Ejemplo: Cola de peticiones (números de pistas) 98, 183, 37, 122, 14, 124, 65, 67. Inicialmente la cabeza está en la pista 53

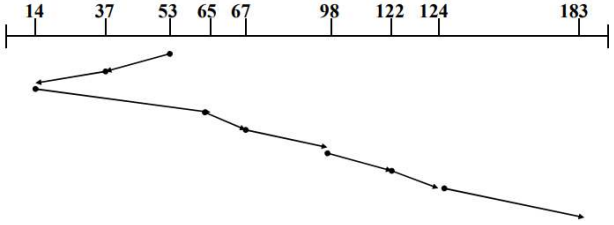
  - FCFS (Fist Come First Served)



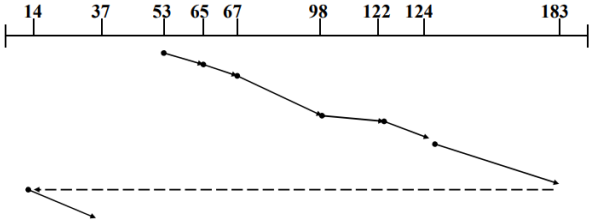
  - SSTF (Shortest Seek Time First): Primero la petición cuyo tºde posicionamiento sea más corto



  - LOOK: Solo se mueve hacia una dirección si hay peticiones pendientes.



  - C-LOOK



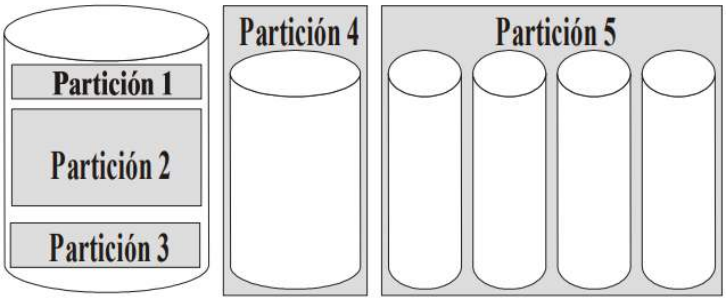
11. SELECCIÓN DE UN ALGORITMO DE PLANIFICACIÓN

En general, para cualquier algoritmo, el rendimiento depende mucho del número y tipo de las peticiones
– si la cola está prácticamente vacía, cualquier algoritmo es válido

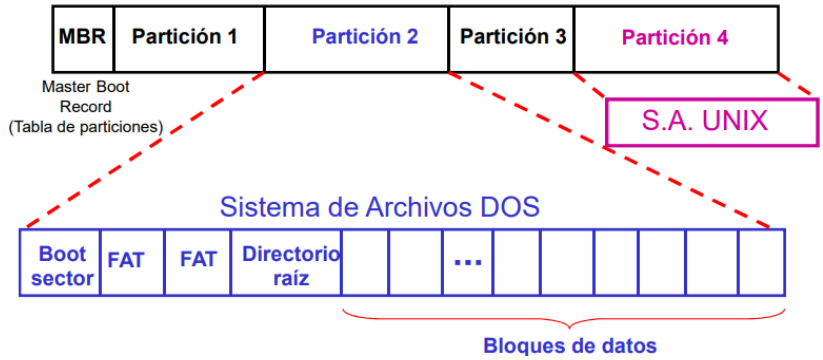
El servicio de peticiones puede estar muy influenciado por el método de asignación de espacio en disco utilizado.
– con asignación contigua: peticiones que reducen el tiempo de posicionamiento
– con asignación no contigua (enlazado o indexado): mayor aprovechamiento de disco pero mayor tiempo de posicionamiento.

12. GESTIÓN DE DISCO

Partición del disco



Ejemplo de organización en particiones:



Formateo del disco:

- Físico: pone los sectores (cabecera y código de corrección de errores) por pista.
- Lógico: escribe la información que el SO necesita para conocer y mantener los contenidos del disco (un directorio inicial vacío, FAT, lista de espacio libre, ...).

Bloque de arranque para inicializar el sistema localizado por bootstrap.
Métodos necesarios para detectar y manejar bloques dañados.

13. GESTIÓN DEL ESPACIO DE INTERCAMBIO

La memoria virtual requiere el uso del disco como extensión de la memoria principal.
Su uso depende de los algoritmos de gestión de memoria del SO

- Intercambio: procesos completos
- Paginación: páginas de procesos.

La cantidad necesaria depende normalmente de la computadora:PC, Workstation, ...
Algunos SO's permiten el uso de múltiples espacios de intercambio.

Asignación del espacio de intercambio. Dos posibilidades:

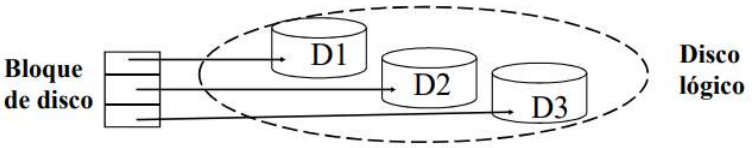
- En el propio sistema de archivos (p.e. Windows) fácil de implementar pero ineficiente
- En una partición independiente:
  - no utiliza estructura de directorios ni sistema de archivos.
  - utiliza su propios algoritmos → eficiente.
  - Problema: aumentan las necesidades de espacio en disco → fragmentación interna

Algunos SO's admiten ambos métodos (Solaris2)

14. EFICIENCIA Y SEGURIDAD DE DISCO

Técnica de entremezclamiento (Striping) de disco

- Un grupo de discos se trata como una unidad. Cada bloque está formado por subbloques, y cada uno de éstos se almacenan en un disco diferente.
- Los discos realizan el posicionamiento y transfieren sus bloques en paralelo decrementa el tiempo de transferencia del bloque



RAID (Redundant Array of Independent Disks)

- Mejoras en el rendimiento y la seguridad
- Organizaciones:
  - Sombra (shawdowing): mantiene duplicados de cada disco
  - Paridad entremezclada de bloques: los datos se escriben en cada disco del array en un bloque y se tiene un bloque extra de paridad en otro disco los datos dañados de un disco se obtienen a partir de los datos del resto de discos

15. SUBSISTENCIA DE ARCHIVOS UNIX

i-nodo: representación interna de un archivo.
Un archivo tiene asociado un único i-nodo, aunque éste puede tener distintos nombres (enlaces)
Si un proceso:

- Crea un archivo se le asigna un i-nodo
- Referencia a un archivo por su nombre se analizan permisos y se lleva el i-nodo a memoria principal

CONTENIDO DE UN I-NODO

- Identificador del propietario del archivo: UID, GID.
- Tipo de archivo (regular, directorio, dispositivo, cauce). Si es 0 el i-nodo está libre.
- Permisos de acceso
- Tiempos de acceso: última modificación, último acceso y última vez que se modificó el i-nodo
- Contador de enlaces
- Tabla de contenidos para las direcciones de los datos en disco del archivo
- Tamaño

16. ARCHIVOS REGULARES Y DIRECTORIOS

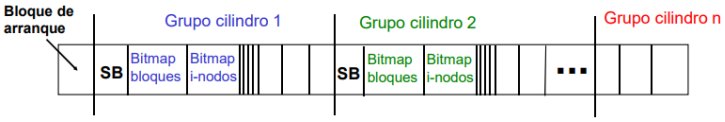
Archivo regular

– Secuencia de bytes, comenzando en el byte 0.
– Los procesos acceden a los datos mediante un desplazamiento (offset).
– Los archivos pueden contener agujeros (lseek, write).

Directorios

– Diferencias entre BSD y SVR4 solventadas utilizando una biblioteca estándar de manejo de directorios.
– Un directorio es un archivo cuyos datos son secuencias de entradas. Dos entradas especiales son . y ..
– Entradas de directorios vacías número i-nodo = 0
– Cada proceso tiene asociado un directorio actual.
– El i-nodo del directorio raíz se almacena en una variable global.

- FFS - Fast File System
  Partición dividida en uno o más grupos consecutivos de cilindros.
  La información del superbloque tradicional se divide en:
  - Superbloque FFS: número, tamaño y localización de grupos de cilindros, tamaño bloque, nº total de i-nodos y bloques.
  - Cada grupo de cilindros tiene su propia lista de i-nodos libres y de bloques libres.



  Cada grupo de cilindros contiene una copia duplicada del superbloque.
  El tamaño mínimo del bloque es 4KB no se necesita tercer nivel de indexación en el i-nodo.
  El bloque se puede dividir en fragmentos direccionables y asignables lista de bloques libres mediante mapa de bits.
  Sólo los bloques directos del i-nodo pueden contener fragmentos.
  Entradas variables en directorios y enlaces simbólicos

  Políticas de asignación:
  – Intenta situar i-nodos de archivos de un único directorio en el mismo grupo de cilindros.
  – Crea un directorio en un grupo diferente al de su padre distribuye datos uniformemente.
  – Intenta asignar bloques de datos en el mismo grupo que su i-nodo correspondiente.
  – Cambia de grupo cuando se alcancen ocupaciones preestablecidas.

CONTENIDO DEL SUPERBLOQUE

- Tamaños (sistema de archivos total, listas de inodos, …).
- Número de bloques libres en el sistema de archivos.
- Tamaño de la
- Número de i-nodos libres en el sistema.
- Además, cuando está en memoria tiene:
  – Campos de bloqueos para bloques e i-nodos libres.
  – Un campo indicando que el superbloque ha sido modificado.

17. INTRODUCCIÓN AL SUBSISTEMA DE ARCHIVOS

Estructuras de datos en memoria principal relacionadas con los archivos:

1) Tabla de i-nodos: el núcleo lee del sistema de archivos el i-nodo cuando se opera con él.
2) Tabla de archivos: mantiene información del puntero de lectura/escritura y los permisos de acceso del proceso al archivo.
3) Tabla de descriptores de archivos: identifica los archivos abiertos por el proceso 1 y 2 son estructuras globales, 3 es una estructura local a cada proceso

18. CONTENIDO DE UN I-NODO DE LA TABLA DE I-NODOS

El núcleo mantiene en la tabla de i-nodos, además del contenido del i-nodo, los siguientes campos:

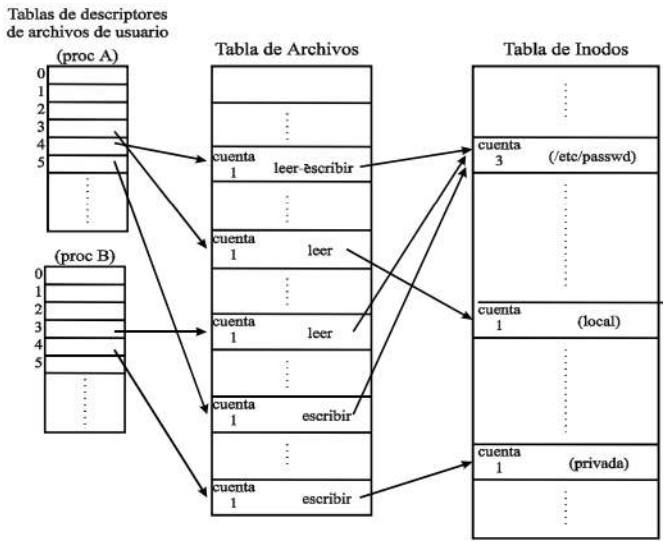
- El **estado** del i-nodo en memoria, indicando si:
  - El i-nodo está bloqueado.
  - Un proceso está esperando a que el i-nodo se desbloquee.
  - La representación en memoria del i-nodo difiere de la copia en disco por cambio en los datos del i-nodo.
  - La representación en memoria del archivo difiere de la copia en disco por cambio en los datos del archivo.
  - » El archivo es un punto de montaje.

- El **número de dispositivo lógico** del sistema de archivos que contiene al archivo.
- El **número de i-nodo**.
- **Punteros a otros i-nodos** en memoria: se usa una estructura hash para enlazar los i-nodos cuya clave de acceso se basa en el número de dispositivo lógico y el número de i-nodo. Existe una lista de i-nodos libres.
- **Contador de referencias**: número de referencias actuales al i-nodo (p.e. cuando varios procesos abren el mismo archivo). Un i-nodo sólo estará en la lista de i-nodos libres cuando su contador de referencias sea 0.

19. TABLA DE ARCHIVOS Y TABLA DE DESCRIPTORES DE ARCHIVOS

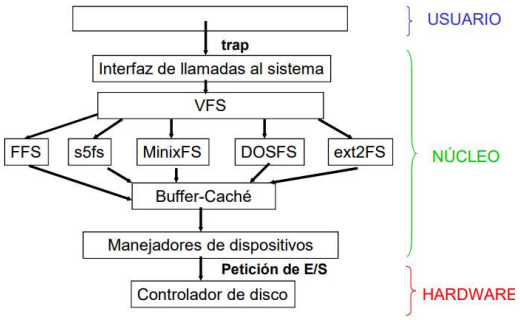
- Tabla de archivos, contenido de cada entrada:
  - puntero al i-nodo correspondiente de la tabla de inodos
  - puntero de lectura/escritura
  - permisos de acceso
  - modo de apertura del archivo
  - contador de entradas de las tablas de descriptores de archivos asociados con esta entrada
- Tabla de descriptores de archivos, contenido de cada entrada:
  - puntero a la entrada de la tabla de archivos correspondiente

ESTRUCTURAS DE DATOS DESPUÉS DE QUE ABRAN DOS PROCESOS ABRAN ARCHIVOS



El sistema abre para cada proceso tres archivos por defecto (primeras entradas de la tabla de descriptores de archivos):
- stdin (teclado)
- stdout y stderr (pantalla)

- UFS - Unix file system
  - ufs = interfaz vnode/vfs (virtual node/ virtual file system) + implementación FFS (Fast File System)
  - Objetivos:
    - Soporte para distintos tipos de sistemas de archivos simultáneamente.
    - Diferentes particiones con diferentes sistemas de archivos.
    - Transparencia en el acceso a un sistema de archivos remoto.
    - Añadir nuevos tipos de sistemas de archivos de forma modular.
- VNODE/VFS



20. ALGORITMO A BAJO NIVEL DEL S.A.

- iget = devuelve un i-nodo de la tabla de i-nodos identificado previamente, si no está, lo carga en la tabla.
- iput = libera un i-nodo de la tabla de i-nodos
- namei = convierte un pathname a un número de i-nodo
- alloc y free = asignan y liberan bloques de disco
- ialloc e ifree = asignan y liberan i-nodos de disco
- bmap = traducción de direcciones lógicas a físicas

ALGORITMO iget: ACCESO A I-NODO EN MEMORIA

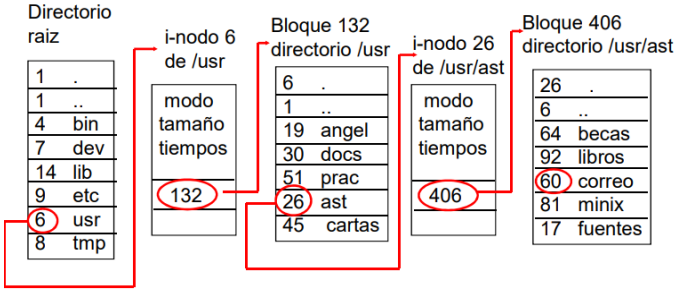
```
Entrada: número de i-nodo de un SA
Salida: i-nodo bloqueado
while (no hecho) {
        if (i-nodo en tabla de i-nodos) {
                if (i-nodo bloqueado) {
                        bloquear (hasta i-nodo desbloqueado);
                        continue; /* vuelve al while */
                }
                bloquear (i-nodo);
                if (i-nodo en la lista i-nodos libres)
                        eliminarlo de la lista de libres;
                contador_referencias ++;
                return (i-nodo);
        }
        else /*i-nodo no está en la tabla de i-nodos*/ {
                if (no hay i-nodos en lista i-nodos libres)
                        return (error);
                liberar i-nodo de la lista i-nodos libres;
                actualizar nº i-nodo y Sistema de Archivos;
                colocar el i-nodo en la entrada correcta de la tabla hash;
                leer i-nodo de disco; /* bread */
                inicializar i-nodo; /*contador_referencias=1*/
                return (i-nodo);
        }
} /* fin de iget */
```

ALGORITMO iput: LIBERACIÓN DE I-NODOS EN MEMORIA

```
Entrada: puntero al i-nodo en memoria (tabla de i-nodos)
Salida: nada
{  bloquear (i-nodo); /*si no lo está*/
        contador_referencias --;
        if (contador_referencias == 0) {
                if (contador_enlaces == 0) {
                        liberar bloques de disco del archivo; /* free */
                        poner tipo de archivo a 0; /* i-nodo libre */
                        liberar (i-nodo); /* ifree */
                } else {
                        if (accedido(archivo) || cambiado(i-nodo) || cambiado(archivo) )
                                actualizar i-nodo de disco;
                        poner i-nodo en la lista de i-nodos libres;
                }
        }
        desbloquear ( i-nodo);
} /* fin de iput */
```

EJEMPLO DE namei

Supongamos el nombre de ruta /usr/ast/correo, ¿cómo se convierte en un nº de i-nodo?



LLAMADAS AL SISTEMA

Devuelven un descriptor	Usan namei	Asignan y liberan i-nodos	Atributos de archivo	E/S archivos	Estructura S.Archivos	Manipulan árbol S.A.
open creat dup pipe close	open creat chdir chroot chown chmod stat link unlink mknod mount umount	creat mknod link unlink symlink	chown chmod stat fstat	read write lseek	mount umount	chdir chown

LLAMADAS AL SISTEMA **OPEN**

Sintaxis: `da = open (<camino>, <opciones>, [<permisos>])`

opciones: modo de apertura (O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_CREAT, ...)

Pasos:

- Convierte el camino en número de i-nodo (namei)
- Si (archivo no existe || no permisos) return (error)
- Asigna una entrada en la tabla de archivos para este i-nodo e inicializa el contador y offset (por defecto, 0)
- Asigna una entrada en la tabla de descriptores de archivos y la enlaza con la entrada de la tabla de archivos anterior
- Desbloquear i-nodo
- Devolver da (descriptor de archivo).

21. CAUCES (PIPES) Y MONTAJE

Permiten la transferencia de datos y la sincronización entre procesos. Capacidad limitada y gestión FIFO.

En la última implementación son bidireccionales y soportan comunicación full-duplex.

Con nombre

- se crean con pipe
- sólo el proceso que lo crea y sus descendientes pueden compartirlo
- son transitorios
- no tienen asociado un nombre de archivo
- tienen asociado un i-nodo en la Tabla de i-nodos

Sin nombre

- se crean con mknod
- todos los procesos con privilegios tienen acceso a él
- no son transitorios, ocupan permanentemente una entrada en un directorio
- tienen asociado un nombre de archivo
- tienen asociado un i-nodo en la Tabla de i-nodos y en disco

MONTAJE Y DESMONTAJE DE UN SISTEMA DE ARCHIVOS

La llamada al sistema mount conecta un sistema de archivos al sistema de archivos existente y la llamada umount lo desconecta.

```
mount (<camino_especial>, <camino_directorio>, <opciones>)
```

El núcleo tiene una tabla de montaje con una entrada por cada sistema de archivos montado:

– Número de dispositivo que identifica el SA montado.
– Puntero a un buffer que contiene una copia del superbloque .
– Puntero al i-nodo raíz del SA montado.
– Puntero al i-nodo del directorio punto de montaje.

Estructuras de Datos después de Montar