Tema-4-Sistema-de-archivos.pdf

fer__luque

Sistemas Operativos

2º Grado en Ingeniería Informática

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

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

Tema 4 Sistema de archivos

Interfaz de los Sistemas de Archivos

Concepto de archivo

Colección de información relacionada y almacenada en un dispositivo de almacenamiento secundario. Su espacio de direcciones lógicas es contiguo. Estructura interna (lógica):

- **Secuencia de bytes**: el tipo de archivo determina su estructura, ej. texto se interpreta como caracteres, líneas y páginas.
- Secuencia de registros de longitud fija.
- Secuencia de registros de longitud variable.

Tipos de archivos: regulares, directorios, de dispositivo (bloques y caracteres), FIFO, socket, enlace simbólico.

Formas de acceso: Secuencial y aleatorio

Atributos (metadatos de archivo):

- **Nombre**. Información para usuario de aplicación
- **Tipo**. Los SOs soportan diferentes tipos de archivo (no confundir con diferentes tipos de formatos de archivo)
- **Localización**. Información sobre su localización en el dispositivo de almacenamiento secundario
- **Tamaño** (*size*). Tamaño actual del archivo
- **Protección** (*permissions*). Permite controlar las operaciones permitidas sobre el archivo. (Junto con identificación permite determinar quién puede leer, escribir y ejecutar el archivo)
- Tiempos y fechas e identificación de usuarios y/o grupos. Necesario para protección, seguridad y monitorización

Operaciones sobre archivos:

Gestión

- **Crear**: open(), link(), symlink(), mknod(), mkfifo()
- **Borrar**: unlink()
- **Renombrar**: rename()
- **Copiar**: rename()
- **Establecer y obtener atributos**: stat(), chmod(), chown(), open(), read(), write(), ...

Procesamiento

- **Abrir y Cerrar**: open() y close().
- **Leer y Escribir** (modificar, insertar, borrar información): read() y write().
- Cambiar el puntero de lectura/escritura: lseek().

Estructura de directorios

Colección de nodos conteniendo **información** acerca de todos los archivos (Organización de archivos). Tanto la estructura de directorios como los archivos residen en el **almacenamiento secundario**. Los archivos de tipo directorio contienen y mantienen la organización del sistema de archivos.

La **organización de los directorios** debe proporcionar:

- **Eficiencia** en la localización de una rchivo en la estructura de directorios
- **Denominación** adecuada a las necesidades de los usuarios:
    - Dos usuarios pueden llamar de la misma forma a diferentes archivos
    - El mismo archivo (metadatos) puede tener varios nombres: enlaces
- **Agrupación**: Permitir agrupar los archivos de forma lógica según sus propiedades o lo que considere el usuario.

Diseño

Árbol: Búsquedas eficientes, agrupación de archivos, permite nombres de camino absoluto y relativo

Grafo: Permite compartición de archivos y directorios, lo que requiere algunas medidas de seguridad

Protección

Consiste en proporcionar un acceso controlado a los archivos: Quién puede acceder y qué operaciones puede hacer. **Tipos de acceso**:

- **Sobre archivos**: leer, escribir, ejecutar, añadir, borrar
- **Sobre directorios**: leer, escribir (crear nuevos archivos/directorios), borrar y cambiar a un directorio.

Listas de acceso

Solución: Hacer el acceso dependiente del identificativo del usuario, UID.

Las **listas de acceso** de usuarios individuales tienen el problema del tamaño, lo que se soluciona con **clases** de usuario:

- **Propietario**: UID del propietario
- **Grupo**: GID del archivo
- **Público**: UID que no sean ni propietarios ni del grupo

Semánticas de consistencia

Especifican cuándo las modificaciones de los datos de un archivo compartido se observan por otros usuarios.

Semántica de Unix

- La escritura se observa tras finalizar la operación
- Varios niveles de compartición

Semánticas de sesión (Sistema de archivos de Andrew)

- La escritura en un archivo no se observa por el resto tras la operación de escritura, solamente se actualiza al cerrar la sesión

Archivos inmutables

- Si un archivo se declara como compartido, no se puede modificar

Funcionalidad básica del SA

- Tener conocimiento de **todos los archivos del sistema de archivos**
- 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 (offset o puntero de R/W) en **direcciones físicas** del disco usando *Logical Block Addressing* (LBA). Los usuarios usan direcciones lógicas, no físicas

Aspectos de diseño del Sistema de Archivos

Estructura del sistema de archivos

Un SA posee dos ámbitos de diseño diferentes:

- Cómo **debe ver el usuario** el sistema de archivos (abstracciones que proporciona):
  - Definir archivo y sus atributos
  - Definir operaciones permitidas
  - Definir la estructura de directorios
- Definir los **algoritmos y estructuras de datos** que deben crearse para establecer la correspondencia entre sistema de archivos lógico y los dispositivos físicos donde se almacenan

Por cuestiones de 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.

Métodos de asignación de espacio en disco

Contiguo

Los datos de un archivo se almacenan en un **conjunto de bloques contiguo en disco**.

Ventajas:

- Metadatos de localización (nº primer bloque asignado y número de bloques asignados)
- Buen acceso secuencial y directo

Desventajas:

- No se conoce inicialmente el tamaño que puede requerir un archivo, lo que puede producir fragmentación externa
- La asignación dinámica de espacio a archivos agravará la fragmentación externa con el tiempo
- Los archivos no pueden crecer a no ser que se realice compactación del espacio de disco

Correspondencia dirección lógica - dirección física

Dir_logica/tama_bloque => cociente(C), resto(R)

Y el acceso a los datos:

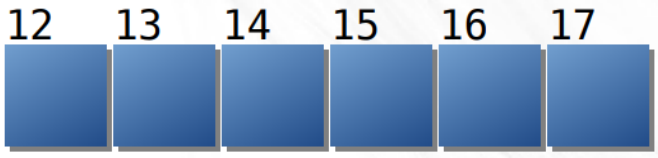
Nº de bloque = Nº primer bloque asignado + C

Desplazamiento dentro de bloque = R

Ej.: Metadatos de localización: (12,6)

read(fd,buf,1) y R/W = 1028; 1028/512 => C=2, R=4

E/S de bloque 12+2=14, buf=dir_base_bloque + 4



Enlazado

los datos se almacenan en bloques de disco **que no tienen que ser consecutivos**, mediante una lista enlazada de bloques. El metadato de localización es únicamente el número del primer bloque

Ventajas:

- Evita la fragmentación externa
- Resuelve el crecimiento dinámico del archivo, evitando la compactación

Desventajas:

- Acceso directo no efectivo (solo secuencial)
- Espacio desperdiciado por almacenar el número del siguiente bloque. Solución: Grupos de bloques (*clusters*)
- Problema de seguridad. Un bloque dañado implica la pérdida de la lista. Solución: Lista doblemente enlazada

Correspondencia dirección lógica - dirección física

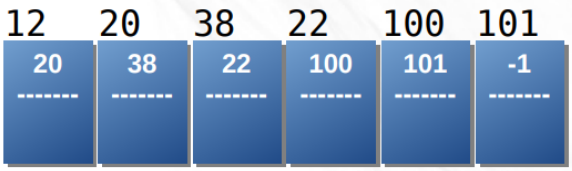
Dir_logica/(tama_bloque – tamaño nº_sig_bloque) => cociente(C), resto(R)

Y el acceso a los datos:

Nº de bloque = C-ésimo bloque de la lista.

Desplazamiento dentro de bloque = tamaño nº_sig_bloque+R

```
Ej: Metadatos de localización: 12
Espacio para Nº_sig_bloque: 1 Byte
read(fd,buf,1) y R/W = 1028; 1028/(512-1) => C=2, R=6
Bloque a acceder: 2-ésimo
Desplazamiento dentro del bloque 2-ésimo: 1 + 6 = 7
E/S del bloque 12 (bloque 0-ésimo)
E/S del bloque 20 (bloque 1-ésimo)
E/S del bloque 38 (bloque 2-ésimo)
```



Enlazado, variante FAT

Una variación del método enlazado inicialmente desarrollada en OS/2, DOS y Windows. Las siglas de *File Allocation Table*. Este método reserva una **sección del disco consecutiva** al principio de una partición para almacenar la FAT.

Esta FAT contiene una entrada por cada bloque de disco y está indexada por número de bloque de disco. Para localizar un bloque, solo tenemos que leer en la FAT, por lo que se optimiza el acceso directo, siempre que la FAT se mantenga en memoria principal.

El principal problema de este método es la posible pérdida de enlaces si un bloque resulta dañado. Para solucionarlo, se pueden mantener dos copias de la FAT en el disco.

FAT en MP y Memoria Secundaria

- Tabla FAT en memoria principal y en memoria secundaria.

Ej.: Metadatos de localización: 4

read(fd,buf,1) y R/W = 1028; 1028/512 => C=2, R=4

Bloque a acceder: 2-ésimo => seguir punteros en FAT: 0-ésimo=4, 1-ésimo=7, 2-ésimo=2

Desplazamiento dentro del bloque 2-ésimo: 4

E/S del bloque 2, buf=dir_base_bloque + 4

0	*
1	*
2	6
3	*
4	7
5	*
6	-1
7	2
8	*
9	*
10	18
11	*
12	20
13	-1
...	...

Indexado

Los números de bloque asignados a un archivo están juntos en una localización concreta: **bloque índice**.

El metadato de localización indica el número de bloque índice y cada archivo tiene asociado su propio bloque índice. Para leer el i-ésimo bloque buscamos el número de bloque de la i-ésima entrada del bloque índice.

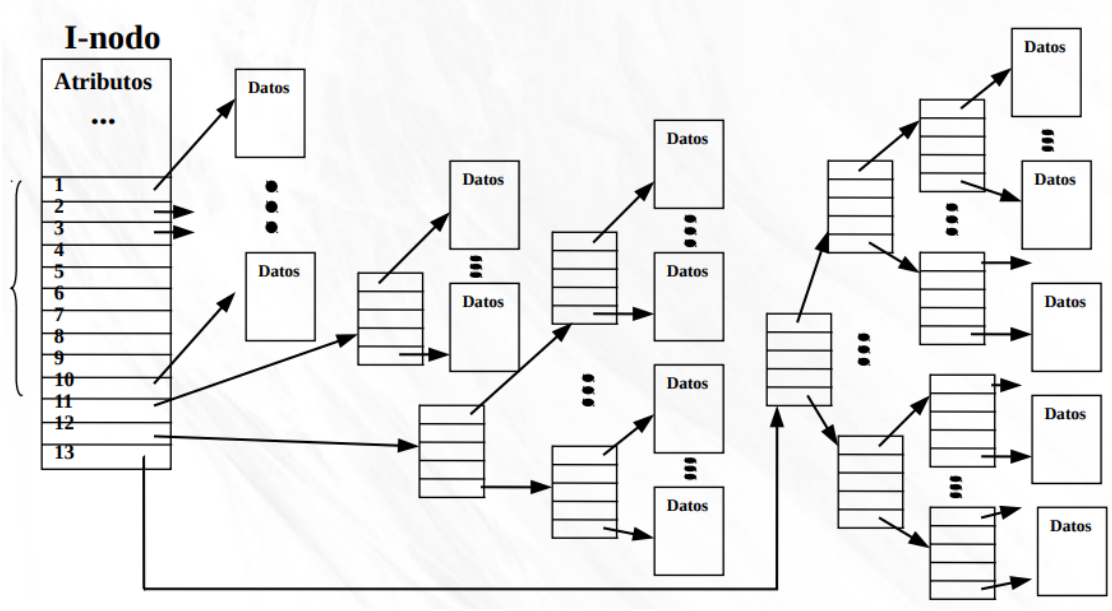
Ventajas:

- Buen acceso directo
- No fragmentación externa

Desventajas

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

Unix (s5fs de SVr2)



Gestión de espacio libre

Para gestionar el espacio libre en un SA, el sistema mantiene una lista de los bloques que están libres. La FAT no necesita ningún método adicional.

Tiene diferentes implementaciones:

Mapa o Vector de Bits

Cada bloque es un bit (0-libre, 1-ocupado). Esto hace que sea fácil encontrar uno o n bloques libres consecutivos, y también el tener archivos en bloques contiguos. Sin embargo es **ineficiente si no se mantiene en MP**.

Lista enlazada

Enlaza todos los bloques libres del disco y guarda el número del primer bloque libre. Esto hace que no se derroche espacio. Relativamente ineficiente porque **proporciona bloques y el acceso directo no es necesario**.

Lista enlazada con agrupación

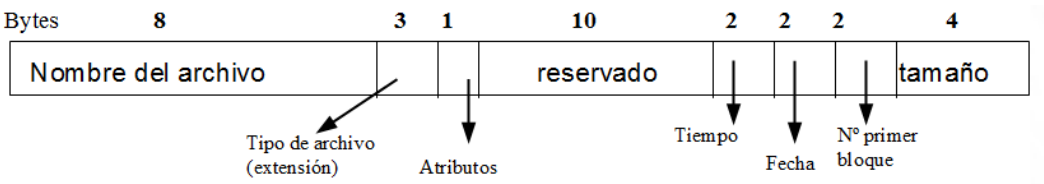
Cada bloque de la lista almacena $n-1$ direcciones de bloques libres, por lo que obtener muchos bloques libres es más rápido.

Cuenta

A cada entrada de la lista con agrupación se le añade el número de bloques libres consecutivos.

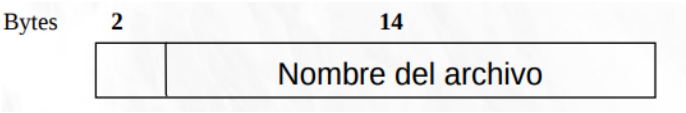
Implementación de directorios

MS-DOS



UNIX

Nombre + referencia a atributos



Cuando se abre un archivo se inicia una **sesión de trabajo**:

- 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 MP
- Cualquier referencia posterior usa la información de dicha tabla

Compartición de archivos (enlaces)

Enlaces simbólicos o blandos (*soft links*)

- Se crea un nuevo archivo de tipo enlace que almacena el camino absoluto o relativo del archivo al que enlaza
- Se puede usar en entornos distribuidos
- Provoca gran número de accesos a disco

Enlaces duros (*hard links*)

- Se crea una nueva entrada en el directorio con un nuevo nombre y se copia la referencia a los atributos
- Problema. Borrar un nombre de archivo no implica borrar los atributos. Solución: contador de enlaces: cuando llega a 0 se borran los atributos

Distribución del sistema de archivos

Los sistemas de archivos se almacenan en discos que pueden dividirse en una o más 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, lista de bloques libres, espacio para atributos de archivo...

Bloque de arranque para inicializar el sistema localizado por *bootstrap*.

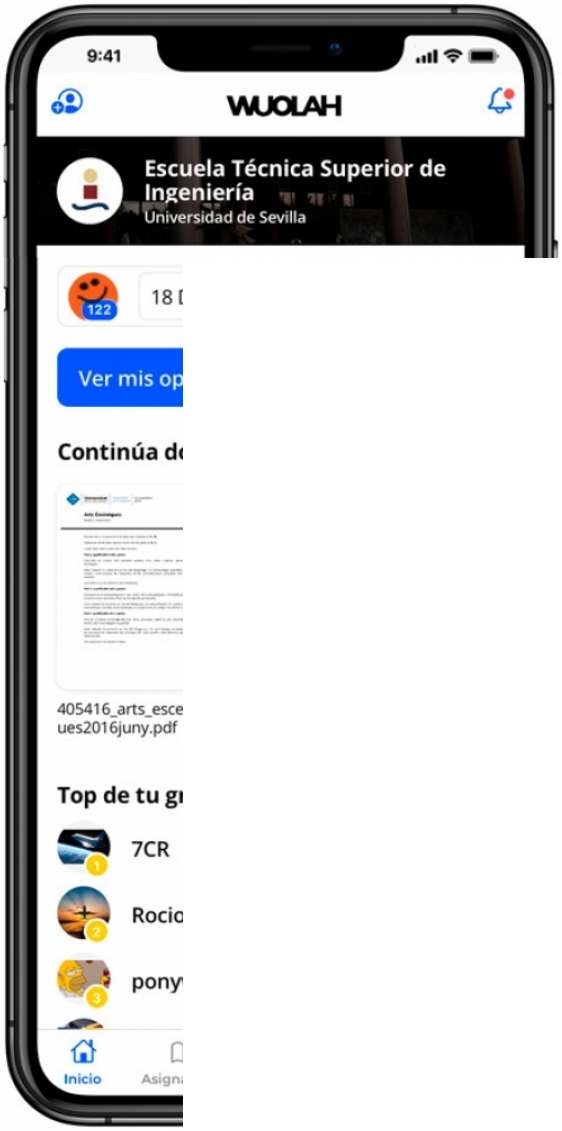
Métodos necesarios para detectar y manejar bloques dañados.

Recuperación de información

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.

Distintos enfoques:

- **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 lista enlazadas que con bloques índices
- 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.



Descarga la APP de Wuolah.

Ya disponible para el móvil y la tablet.

Available on the App Store

GET IT ON Google Play