Sistema de Ficheros

Este módulo explica todos esos aspectos relacionados con la gestión y uso de los ficheros. La interfaz que conoce el usuario, tamaño de bloque, asignar espacio a ficheros, estructuras de directorios, gestionar espacio libre, consistencia de los datos y rendimiento.

Tabla de Contenidos

Autor: Lina García Cabrera

Copyright: Copyright by Lina García Cabrera

1 Introducción

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

Todos los **dispositivos de E/S deben ser transparentes a los usuarios**. La mayoría de los sistemas ofrecen una serie de servicios que nos permiten olvidarnos de todos los detalles específicos de los dispositivos de almacenamiento permanente, y que liberan a los usuarios de tener que trasladar datos explícitamente entre memoria principal y memoria secundaria.

Este módulo explica todos esos aspectos relacionados con la administración y manipulación de los ficheros. Primero **la interfaz que conoce el usuario**, aspectos básicos sobre ficheros, sus organizaciones lógicas más comunes, y las formas en las que podemos organizar un sistema de ficheros.

Luego se estudia el ***sistema de ficheros desde el punto de vista del sistema operativo***. Se analiza el parámetro **tamaño de bloque**, las distintas alternativas de **asignar espacio de disco a un fichero** (contigua, enlazada e indexada), las diversas **estructuras de directorios** que

hacen posible la localización de un fichero y, por último, las posibles formas de **gestionar el espacio libre** (mapas de bits, lista enlazada).

También se ocupa de los mecanismos que velan por la **consistencia y la integridad de los datos** en un sistema de ficheros: métodos que permiten recuperar la información en caso de pérdida, y de aquellos que detectan y resuelven las posibles inconsistencias de un sistema de ficheros (comprobación de bloques y directorios).

El último factor de diseño que se examina será el **rendimiento del sistema de ficheros**: técnicas relacionadas con la gestión de los ficheros que consiguen reducir el tiempo que se invierte en transferir un fichero.

El **sistema de ficheros** está compuesto por dos partes diferenciadas: una colección de ficheros, cada uno almacena una serie de datos, y una estructura de directorios, que organiza todos los ficheros del sistema y proporciona información sobre ellos. Un **fichero** es una secuencia de bits, bytes, líneas de texto o registros cuya interpretación está definido por su creador.

OBJETIVOS

- Saber cómo se asigna espacio a los ficheros en disco, las diversas estructuras de directorios y gestionar el espacio libre del disco.
- Conocer los mecanismos que velan por la consistencia y la integridad de los datos del sistema de ficheros.
- Comprender cómo se puede mejorar el rendimiento del sistema de ficheros.

2 Motivación

2º Grado en Ingeniería en Informática

teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

Todas las aplicaciones necesitan almacenar y recuperar algún tipo de información. Al ejecutarse un proceso, éste puede almacenar una cantidad limitada de información dentro de su propio espacio de direcciones. Sin embargo, la capacidad de almacenamiento queda restringida, a lo más, al tamaño del espacio de direcciones virtuales. Para algunas aplicaciones, este **tamaño** es adecuado; pero para otras, como el sistema de reservas de una compañía aérea, es demasiado pequeño.

El segundo problema, relativo al almacenamiento de la información dentro del espacio de direcciones es que, **cuando el proceso termina la información se pierde**.

Un tercer problema, en el caso de varios procesos, es que se requiere con frecuencia el acceso a la información al mismo tiempo. Si tenemos almacenada cierta información dentro del espacio de direcciones de un solo proceso, únicamente ese proceso podrá tener acceso a dicha información. La forma de resolver este problema es hacer que la propia **información sea independiente de todos los procesos**.

Así pues, tres son las condiciones esenciales para el almacenamiento de la información a largo plazo:

1. Debe ser posible almacenar una **cantidad muy grande de información**.
2. La **información debe sobrevivir a la ejecución del proceso** que la utiliza.
3. Debe ser posible que varios procesos tengan **acceso concurrente a la información**.

La solución a estos problemas es el almacenamiento de la información en **discos** y otros medios externos, en unidades denominadas **archivos o ficheros**. Los procesos podrán entonces leerlos y escribir otros nuevos. La información almacenada en los ficheros debe ser **persistente**; es decir, no debe verse afectada por la creación y terminación de un proceso.

Los ficheros son administrados por el sistema operativo. Su estructura, nombre, forma de acceso, uso, protección e implantación son temas fundamentales en el diseño de un sistema operativo. Aquella parte del sistema operativo que trabaja con los ficheros se conoce como el **sistema de ficheros**.

Desde la perspectiva del usuario, el aspecto más importante del sistema de ficheros es su forma de aparecer ante él, esto es, qué constituye un fichero, cómo se nombran y protegen y, qué operaciones se permiten realizar sobre ellos.

Los detalles de si se utilizan listas enlazadas o mapas de bits para controlar el espacio que está libre o cuántos sectores hay en un bloque lógico, solo son de interés para los diseñadores de sistemas de ficheros. Por esta razón se ha estructurado el tema en dos grandes apartados, uno concerniente a la **interfaz de usuario que ofrecen ficheros y directorios**, y otro a **la perspectiva** que tiene del **sistema de ficheros el sistema operativo**.

3 Sistemas de Ficheros desde la Perspectiva del Usuario

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
A continuación se estudiará los ficheros desde el punto de vista del usuario, esto es, cómo se puede hacer uso de ellos y qué propiedades tienen.

3.1 Aspectos Básicos sobre Ficheros

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

Nombre de un fichero

Los **ficheros son un mecanismo de abstracción**. Son una forma de almacenar información en un disco y volver a leerla más adelante. La característica más importante de cualquier mecanismo de abstracción es probablemente la forma de nombrar a los objetos administrados.

Cuando un proceso **crea un archivo**, le da a éste **un nombre**. Cuando el proceso concluye, el archivo sigue existiendo y otros procesos pueden tener acceso a él mediante su nombre. Las reglas exactas para los nombres de ficheros varían un poco con cada sistema.

Muchos sistemas operativos utilizan nombres de archivo con dos partes, separadas por un punto, como en prog.c. La parte posterior al punto es la **extensión de archivo** e indica por lo general algo relativo al archivo (como su **tipo**).

En UNIX, el tamaño de la extensión, si ésta existe, se deja al usuario, incluso un archivo puede tener más de un punto, dando lugar, como tal, a más de una extensión. En ciertos casos, las extensiones de los ficheros son meras convenciones y no están forzadas a un valor concreto, pero no siempre es así.

Tipos de Ficheros

Muchos sistemas operativos permiten varios tipos de ficheros.

En Unix se distinguen 4 tipos de archivos:

- Ficheros regulares u ordinarios: son archivos que contienen información que han introducido un usuario, aplicación o programa de sistema.
- Directorios: una lista de nombres de archivos que organizan la información de forma jerárquica. Son archivos ordinarios pero con una estructura concreta.
- Especiales: se usan para acceder a los dispositivos periféricos pueden ser de carácter (teclado) o de bloque.
- Tubos con nombre para intercambio de información entre las aplicaciones.

En Unix se distinguen 4 tipos de archivos:

- Ficheros regulares u ordinarios: son archivos que contienen información que han introducido un usuario, aplicación o programa de sistema.
- Directorios: una lista de nombres de archivos que organizan la información de forma jerárquica. Son archivos ordinarios pero con una estructura concreta.
- Especiales: se usan para acceder a los dispositivos periféricos pueden ser de carácter (teclado) o de bloque.
- Tubos con nombre para intercambio de información entre las aplicaciones.

Los **ficheros regulares** son, en general, ficheros ASCII o binarios. Los **ficheros en ASCII** constan de líneas de texto de longitud variable. Los **ficheros binarios** son ficheros que contienen datos binarios que no se corresponden con caracteres ASCII imprimibles, además pueden tener una cierta estructura interna.

Atributos Asociados a los Ficheros

Cada archivo tiene su nombre y datos. Además, todos **los sistemas operativos asocian información adicional a cada archivo**; por ejemplo, la fecha y hora de su creación, así como su tamaño. Estos elementos adicionales se llaman **atributos**.
La cantidad de atributos que se registran varía en cada sistema operativo, pueden mantener atributos relacionados con la protección de los ficheros, relacionados con contraseñas, propiedades, visibilidad y temporalidad, atributos que hacen mención al tamaño del archivo, etc.

Operaciones con Ficheros

Los distintos sistemas proporcionan diversas operaciones para permitir el almacenamiento y la recuperación de información de los ficheros, las llamadas al sistema más comunes relacionadas con los ficheros se enumeran a continuación:

1. **CREATE** (Crea un archivo).
2. **DELETE** (Elimina el archivo del dispositivo de almacenamiento).
3. **OPEN** (Conecta el archivo al proceso).
4. **CLOSE** (Desconecta el archivo del proceso).
5. **READ** (Lee información del archivo).
6. **WRITE** (Escribe información en el archivo).
7. **APPEND** (Escribe información al final).
8. **SEEK** (Se posiciona en un archivo de acceso aleatorio).

Ficheros Mapeados en Memoria

Muchas personas piensan que el acceso a ficheros es poco eficiente, en especial si se compara con el acceso a la memoria ordinaria. Por esta razón, ciertos sistemas operativos proporcionan una forma de asociar los ficheros con un espacio de direcciones de un proceso en ejecución. Podemos imaginar la existencia de dos nuevas llamadas al sistema: MAP y UNMAP. La primera utiliza un nombre de archivo y una dirección virtual y hace que el sistema operativo asocie el archivo con la dirección virtual en el espacio de direcciones.
Aunque el mapeo de ficheros elimina la necesidad de E/S y por lo tanto facilita la programación, introduce sus propios problemas:

- Es difícil que el sistema conozca la longitud exacta del archivo de salida.
- Si un proceso asocia un archivo con memoria, y otro proceso abre dicho archivo para efectuar modificaciones se puede llegar a situaciones de inconsistencia.
- Puede que el archivo sea mayor que un segmento de memoria, incluso mayor que todo el espacio de direcciones virtuales, será necesario poder asociar sólo parte del archivo y no todo.

3.2 Organización Lógica de los Ficheros

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Los ficheros se pueden estructurar de varias maneras:

- **Una serie no estructurada de bytes**. Al sistema operativo no le interesa ni se preocupa por el contenido de cualquier archivo. Lo único que ve son los bytes; cualquier significado debe ser impuesto por los programas a nivel de usuario. Tanto UNIX como Windows utilizan este enfoque. Esta estructura proporciona una gran flexibilidad, ya que son los programas de usuario los que se encargan de organizar la información.
- **Una secuencia de registros de longitud fija**. La operación de lectura obtiene un registro y las operaciones de escritura escriben sobre un registro o añaden uno nuevo.
- **Un árbol de registros (ISAM)**. No necesariamente todos los registros tienen por que tener la misma longitud; cada uno de ellos tiene un campo clave en una posición fija del registro. El árbol se ordena mediante este campo, con el fin de optimizar las búsquedas. Esta organización se utiliza ampliamente en grandes computadores *mainframe*, usadas en el procesamiento de datos comerciales.

Acceso a un Fichero
Los primeros sistemas operativos tenían un solo tipo de acceso a los ficheros: el **acceso secuencial**. Los ficheros secuenciales son convenientes cuando el medio de almacenamiento son las cintas magnéticas y no los discos.
Cuando se comenzaron a utilizar los discos para el almacenamiento de ficheros, fue posible acceder a los bytes o registros de un archivo en un orden cualquiera, hablamos así de ficheros de **acceso** aleatorio o **directo**.

3.3 Estructura del Sistema de Ficheros

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Para seguir la pista de los ficheros, el sistema ofrece normalmente **directorios**, que en muchos sistemas **son en sí mismos ficheros**. Los directorios suelen contener una serie de entradas, una por cada fichero.
En su forma más sencilla, el sistema tiene un solo directorio que incluye todos los ficheros de todos los usuarios. Si hay muchos usuarios que escogen los mismos nombres para sus ficheros, los conflictos y la confusión surgidos convierte el sistema muy pronto en un caos. Este modelo de sistema se usa sólo en los sistemas operativos de los microcomputadores más primitivos.
Una mejora de la idea de tener un solo directorio para todos los ficheros es la de tener un directorio por usuario. Este diseño elimina los conflictos de nombres entre usuarios, pero no satisface a usuarios que tengan muchos ficheros. Es normal que los usuarios quieran agrupar sus ficheros de maneras lógicas. Hace falta un mecanismo razonable para poder agrupar todos los ficheros.
En general lo que se necesita es disponer de una **jerarquía como la que puede establecer un**

árbol de directorios. Con este mecanismo, cada usuario puede tener tantos directorios como necesite para poder agrupar los ficheros de forma natural.
Al organizar el sistema de ficheros como un árbol de directorios, hace falta una forma de **especificar nombres de fichero**. Se emplean normalmente para ello dos métodos distintos:

- Dar a cada fichero un nombre de **ruta absoluta**, formado por la **ruta que hay que seguir desde el directorio raíz hasta ese fichero**. Los nombres de ruta absoluta comienzan siempre en el directorio raíz, y son únicos.
- Utilizar nombres de **ruta relativa**. Éstos s**e usan junto con el concepto de directorio de trabajo** o directorio actual en el que se encuentra el usuario. Los usuarios pueden designar un directorio como el directorio de trabajo actual, en cuyo caso se considera que todos los nombres de ruta que no comiencen en el directorio raíz son relativos al directorio de trabajo.

Operaciones con Directorios
Damos un ejemplo (próximo a UNIX) de llamadas al sistema para gestión de directorios:

1. **CREATE** (Crea un directorio).
2. **DELETE** (Elimina un directorio).
3. **OPENDIR** (Abre un directorio para leer sus entradas).
4. **CLOSEDIR** (Cierra el directorio y lo libera).
5. **READDIR** (Lee la siguiente entrada del directorio).
6. **RENAME** (Cambia el nombre).
7. **LINK** (Crea un enlace).
8. **UNLINK** (Elimina un enlace).

4 El Sistema de Ficheros desde la perspectiva del Sistema Operativo

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Al usuario sólo le preocupa la interfaz del sistema de ficheros: cómo se puede hacer referencia a un fichero, qué operaciones le están permitidas, qué puede saber de la estructura arborescente de directorios.
El **Sistema Operativo debe concentrarse en aspectos de diseño:**

- cómo se almacenan ficheros y directorios,
- cuánto espacio de disco queda libre, y
- qué puedo hacer para conseguir que la gestión del sistema de ficheros sea más eficiente y, por supuesto, más fiable.

4.1 El Tamaño de Bloque

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Los ficheros suelen almacenarse en discos, y es por esto que la gestión del espacio en disco es de importancia capital para los diseñadores de sistemas de ficheros. Existen dos métodos para almacenar un fichero de n bytes:

1. Utilizar n bytes consecutivos de espacio en disco.
2. Subdividir el fichero en un cierto número de bloques. Cada bloque es uno o varios sectores, una potencia de dos.

El almacenamiento de ficheros como secuencias contiguas de bytes tiene el inconveniente de que si crecen, lo cual es frecuente, habrá que moverlos a otra área del disco, resultando una tarea sustancialmente costosa. Por esta razón, casi todos los sistemas de ficheros los dividen en bloques de tamaño fijo, que no tienen por qué estar contiguos.

Una vez que se ha decidido almacenar los ficheros en bloques de tamaño fijo, la cuestión es elegir el **tamaño del bloque**. Dada la forma en que los discos están organizados, el sector, la

pista y el cilindro son candidatos obvios a convertirse en unidades de asignación.

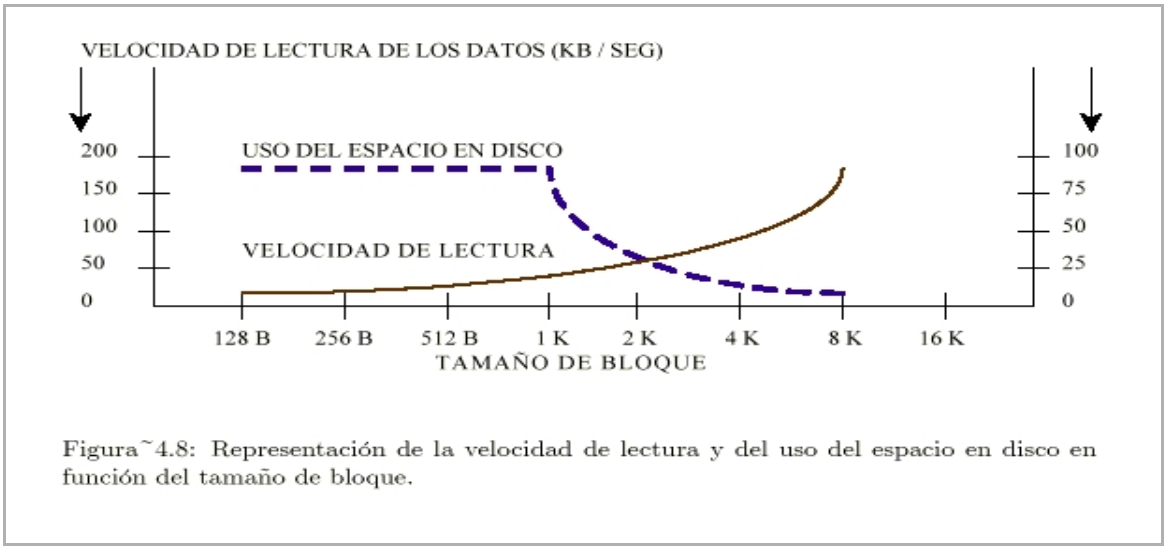
Elegir **una unidad de asignación muy grande**, como el cilindro, significa que cada fichero, incluso los que tengan un sólo byte, ocupará un cilindro completo, generando una **enorme fragmentación interna** del espacio.

> Estudios realizados por Mullender y Tanenbaum, en 1984, demostraron que el tamaño medio del fichero en entornos UNIX es de alrededor de 1K, de forma que reservar cilindros de 32K para cada fichero desperdiciaría 31/32, es decir, el 97% del espacio total del disco.

Por otro lado, si se selecciona **una unidad de asignación pequeña**, cada fichero tendrá muchos bloques. Leer un bloque suele requerir una operación de búsqueda y un tiempo de latencia, por lo que **leer ficheros con muchos bloques pequeños resultará lento**.

> Como ejemplo, podemos considerar un disco con 32768 bytes por pista, un tiempo de rotación de 8.3 mseg. y un tiempo de búsqueda de 3 mseg. El tiempo de lectura en milisegundos de un bloque de k bytes en una posición aleatoria es la suma del tiempo de búsqueda, el tiempo de latencia y el tiempo de transferencia: $3 + 4,15 + k/32768 * 8,3$

La curva sólida de la Figura 1 muestra la velocidad de transferencia de datos de este disco en función del tamaño de bloque, si consideramos todos los ficheros de 1K, la curva discontinua da la utilización del espacio en disco. Lo malo es que utilizaciones elevadas del espacio conducen a velocidades de transferencia bajas, y viceversa. El uso eficaz del espacio está inherentemente en conflicto con el tiempo que se tarda en llevar a cabo las transferencias.



Figura~4.8: Representación de la velocidad de lectura y del uso del espacio en disco en función del tamaño de bloque.

Figura 1. Velocidad de transferencia y utilización del espacio en disco.

El compromiso más normal consiste en escoger tamaños de bloque de 512, 1K, 2K ó 4K bytes. Si se elige un bloque de 1K en un disco de sectores de 512 bytes, el sistema de ficheros siempre leerá o escribirá dos sectores consecutivos, tratándolos como una sola unidad indivisible.

4.2 Asignación de Espacio a Ficheros

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Uno de los aspectos clave relacionados con el diseño del sistema de ficheros es decidir cómo asignamos espacio a nuestros ficheros. Este almacenamiento deberá hacerse de tal forma que se consiga **usar el espacio de disco lo más eficientemente** posible al mismo tiempo que se mantienen **mínimos los tiempos de acceso a los ficheros**.

4.2.1 Asignación Contigua

2º Grado en Ingeniería en Informática

teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

El esquema de asignación más simple es el que almacena cada fichero como un conjunto de bloques de datos adyacentes en el disco.

Este esquema tiene dos ventajas significativas:

- su **fácil implementación**, ya que solamente se ha de registrar el bloque de comienzo del fichero, y
- su **excelente rendimiento**, ya que todo bloque del fichero se puede leer con sólo un acceso a disco. Admite tanto acceso secuencial como directo. Si se sabe el bloque de comienzo, sólo hay que sumarle el número de bloque al que queremos acceder.

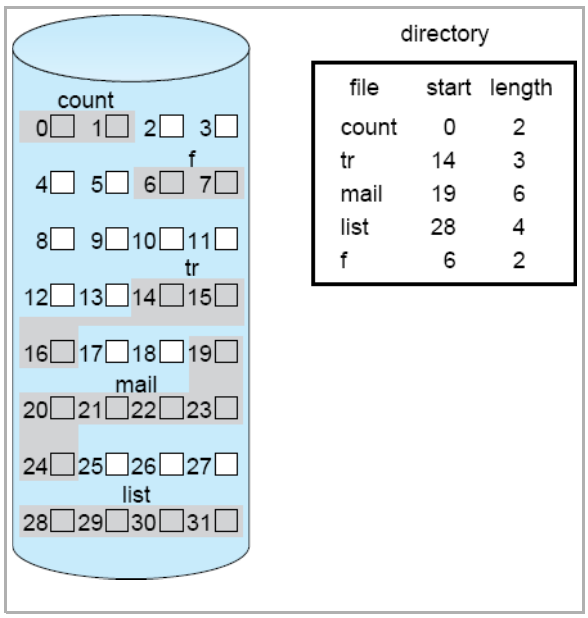


Figura 2. Asignación Contigua [Silberschatz2010].

Por desgracia, este método tiene también dos defectos:

- El primero es que no es realizable, porque el **sistema operativo no sabe cuánto espacio en disco debe reservar para cada fichero**. Sin embargo, en los sistemas donde hay que escribir los ficheros de un solo golpe, puede utilizarse con grandes ventajas.
- La segunda desventaja es la **fragmentación externa del disco**. Se desperdicia el espacio que podría utilizarse de otra forma.

Tenemos también el problema de **cómo encontrar espacio para un nuevo fichero**. Se puede gestionar la memoria de forma parecida a la memoria y se podrían aplicar las estrategias *First-Fit*, *Best-Fit* y *Worst-Fit*.

El problema de la fragmentación externa se puede solucionar con la **compactación**. Pero eso supone un **gran coste**. Además, los **ficheros crecen**, es difícil saber el espacio que necesitarán. Se puede retrasar el problema:

- Asignar más espacio.
- Reubicarlo en un espacio mayor.

4.2.2 Asignación Enlazada

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

Un método posible es almacenar los bloques de un fichero como una **lista enlazada**.

> Estableciendo bloques de disco de 1024 bytes, cada uno puede contener 1022 bytes de datos y un puntero de 2 bytes al siguiente bloque de la cadena.

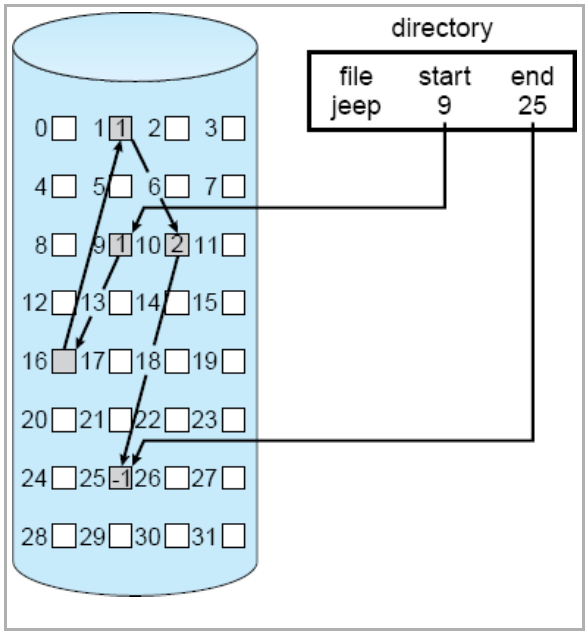


Figura 3. Asignación Enlazada [Silberschatz2010].

Este método aunque presenta ventajas:

- la **ausencia de fragmentación externa** y
- posibilidad de que los **ficheros crezcan mientras tengamos espacio** (asignándole bloques libres),

sin embargo, presenta serios inconvenientes:

- el **espacio requerido por los punteros** provoca el que el **número de bytes de datos de un bloque deje de ser potencia de dos**, lo que puede ser molesto en algunos casos,
- el **acceso aleatorio es costoso de implementar** (es necesario recorrer la lista).
- podemos tener **problemas de fiabilidad** debido a la facilidad con la que un puntero se puede perder o dañar produciéndose referencias cruzadas entre los ficheros.

Aun así, la idea de representar ficheros como listas enlazadas puede considerarse como eficiente si mantenemos los punteros en memoria.

4.2.3 Asignación Indexada

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

La **asignación enlazada no permite el acceso directo** porque tenemos bloques directos por todo el disco y punteros dispersos por todo el disco.

Las desventajas de la asignación mediante lista enlazada se pueden eliminar si se toma la palabra del **puntero de cada bloque del disco y se le coloca en una tabla o índice de la memoria**.

> La Figura 4 muestra la apariencia de la tabla. El fichero X utiliza los bloques 217, 618, 339 y 2 del disco, en ese orden, mientras que el fichero B utiliza los bloques 5, 9 y 12, en ese orden.

Mediante esta organización, **todo el bloque está disponible para los datos**. Además, el **acceso aleatorio es mucho más sencillo**. Aunque hay que seguir la cadena para encontrar un bloque determinado del fichero, esto se realiza sobre la tabla que se encuentra en memoria, y no mediante accesos a disco.

MS-DOS utiliza este método para la asignación de espacio a los ficheros en discos y para ello utiliza la denominada **FAT** (*File Allocation Table*).

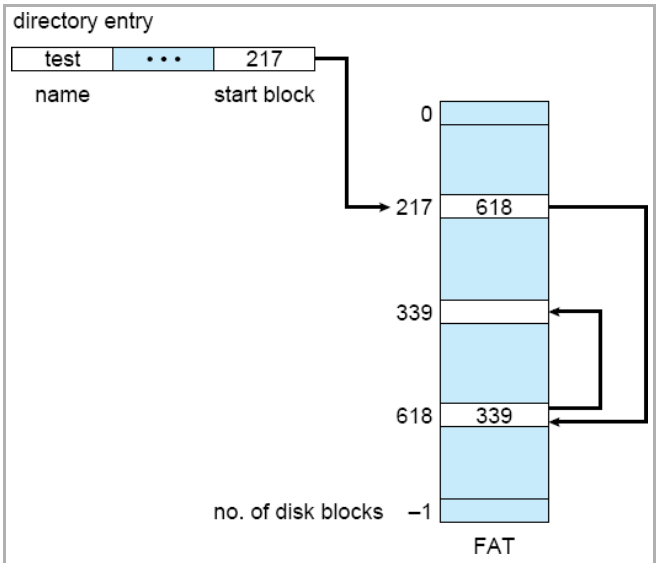


Figura 4. Asignación indexada FAT [Silberschatz2010].

La principal desventaja de este método es que **toda la tabla debe estar en memoria todo el tiempo para que funcione**.

Con un solo disco de gran tamaño, digamos de 500000 bloques de 1K (500M), la tabla tendría 500000 entradas, cada una de las cuales tendría al menos tres o cuatro bytes. Así, la tabla ocuparía 1.5 o 2 megabytes todo el tiempo.

UNIX utiliza una variante de la asignación indexada. Cada fichero tiene una pequeña estructura de datos asociada (en disco), llamada **i-nodo**, como muestra la Figura 5, esta estructura de datos que contiene **información de protección y de facturación**, además de **información que permite acceso directo** a los 10 primeros **bloques de datos** de un fichero, mientras que al resto de bloques hay que acceder de forma **indirecta**.

Cuando un fichero tiene más de 10 bloques de disco, se adquiere un bloque libre y se pone un **puntero indirecto** simple apuntándole. Este **bloque se usa para contener punteros a bloques de disco**.

Para bloques de 1K y direcciones de disco de 32 bits, el **bloque indirecto simple** puede contener hasta 256 direcciones de disco. Este método es suficiente para ficheros de hasta 266 bloques.

Por encima de 266 bloques, se emplea el puntero **indirecto doble**, que apunta a un bloque de disco de hasta 256 punteros, que apuntan a 256 bloques indirectos simples, en vez de hacerlo a bloques de datos. El bloque indirecto doble cubre hasta ficheros con 266+2562=65.802 bloques.

Para ficheros de más de 64M hay que emplear el **puntero indirecto triple**. Con este planteamiento los ficheros mayores de 16 gigabytes no se pueden procesar.

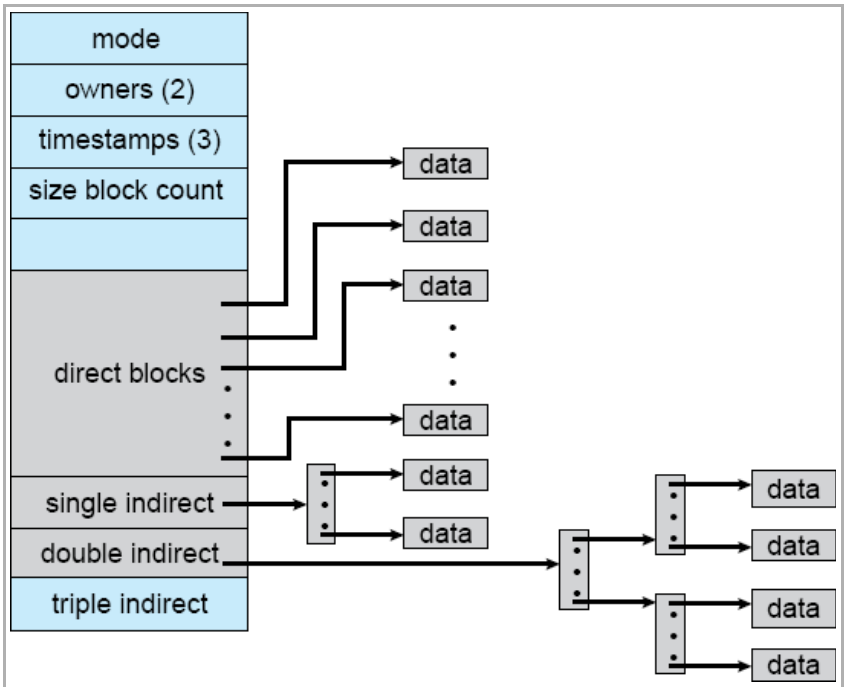


Figura 5. Asignación Indexada i-nodo en Unix [Silberschatz2010].

La potencia del esquema de UNIX reside en que los **bloques indirectos sólo se utilizan cuando hacen falta**. Para ficheros menores de 10K no hace falta ningún bloque indirecto. Incluso **para ficheros más grandes, se necesitan como mucho tres referencias a disco** para localizar la dirección de cualquier byte del fichero, sin contar la lectura del nodo-i, que se lee al abrir el fichero y se deja en memoria hasta que se cierra.

4.3 Estructura Interna de los Directorios

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

Antes de utilizar un fichero hay que abrirlo, para tal fin el usuario proporciona el nombre de ruta correspondiente al sistema operativo, éste último lo emplea para localizar sus correspondientes bloques de datos en el disco, de manera que el fichero propiamente dicho pueda leerse o escribirse posteriormente. La **transformación de nombre de ruta en nodos-i**, o equivalente, nos lleva a la cuestión de cómo organizar ficheros y directorios.

Los **directorios de ficheros** representan la organización lógica de los ficheros sobre los dispositivos. Cada entrada, denominada **bloque de control del fichero**, describe las propiedades lógicas que definen un fichero (nombre, información de acceso y protección, etc.). Puede que cada entrada contenga un puntero a la entrada correspondiente en el directorio de dispositivo. Nuestro interés se centra, principalmente, en la estructura de directorios de ficheros.

Los sistemas de directorios se organizan en **sistemas con árboles de directorios jerárquicos**. En este tipo de organización cada entrada contiene, entre otras cosas, el nombre del fichero y el número de su primer bloque (o el número de su nodo-i en el caso de UNIX).

Éste puede utilizarse como un índice de una tabla de asignación de ficheros (FAT), para encontrar el número del siguiente bloque, y así sucesivamente, localizando así todos los bloques de un fichero dado.

En UNIX toda la información sobre el tipo, tamaño, tiempos, propietario y bloques en disco del fichero está contenida en su nodo-i, luego los **directorios en UNIX son simples ficheros** que almacenan entradas que **contienen el nombre del fichero y su número de nodo-i**.

A la hora de estudiar cuestiones de implementación es de carácter obligado tener presentes algunas consideraciones:

- **¿Dónde deben guardarse los directorios, en disco o en memoria?** Los más interesante es almacenar **en memoria principal sólo las entradas activas**, mientras que

el resto permanecen en el dispositivo. La forma de conocer cuales son las entradas activas es prestando atención a los momentos en los que se solicita la apertura de los ficheros para su procesamiento.

- **¿Cómo se tiene acceso al directorio raíz?** La forma de conocer la posición física del directorio raíz es colocarlo, siempre, en una **posición conocida del volumen**, de esta forma, el sistema operativo puede localizarlo accediendo a esa posición.

4.3.1 Ejemplos de Organización Real de Directorios

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

Directorios en MS-DOS

En MS-DOS cada entrada de directorio consta de 32 bytes, de los cuales 11 se utilizan para el nombre y extensión del fichero, 1 byte para atributos, 4 bytes para hora y fecha, 10 bytes reservados, 4 bytes para reflejar el tamaño del fichero y 2 bytes que se utilizan como puntero al primer bloque.

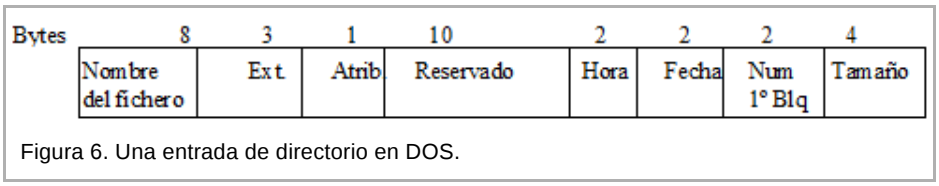


Figura 6. Una entrada de directorio en DOS.

MS-DOS divide el disco físico en **zona de datos y la tabla de asignación de ficheros** (FAT). La **FAT es una lista enlazada**, donde cada entrada referencia a los bloques de datos del dispositivo. Las dos primeras posiciones especifican el tamaño del disco, en el resto de posiciones pueden aparecer códigos como:

- EOF: último bloque de datos de un fichero,
- FREE: bloque de datos libre,
- BAD: bloque de datos defectuoso.

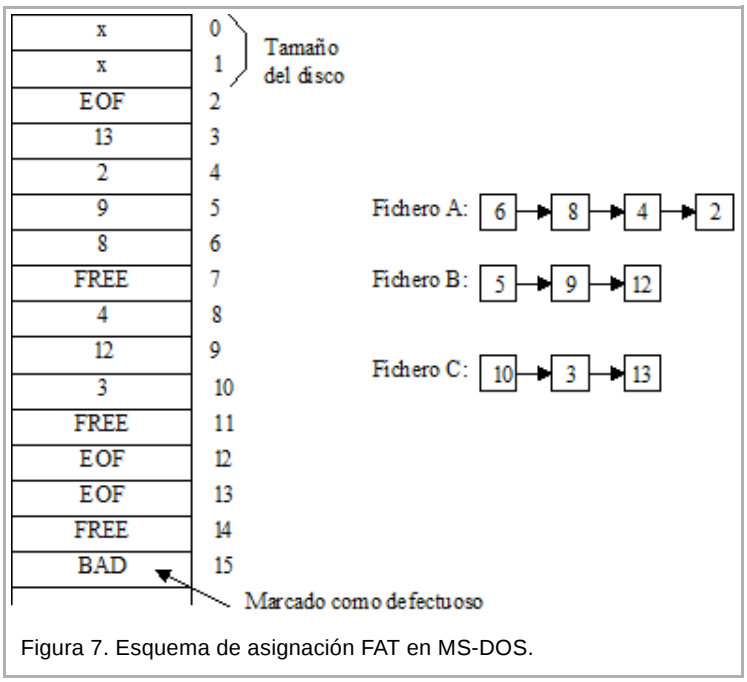


Figura 7. Esquema de asignación FAT en MS-DOS.

Directorios en UNIX

En UNIX, toda la información sobre el fichero está contenida en su nodo-i. Los directorios son

ficheros donde cada entrada consta de 16 bytes, 14 de ellos para almacenar el nombre del archivo y los otros 2 almacenan el número de nodo-i asociado al fichero (ver la Figura 8).

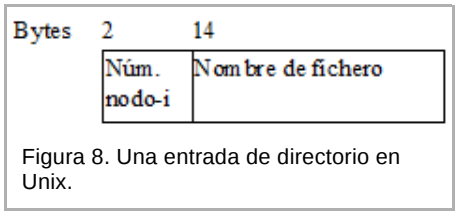


Figura 8. Una entrada de directorio en Unix.

Cuando se pretende acceder a un fichero concreto, utilizando un nombre de ruta absoluta, primero el sistema de ficheros **localiza el directorio raíz** (el nodo-i de la raíz se encuentra en un lugar fijo del disco), a continuación, **se busca en el directorio** (es decir, fichero correspondiente al directorio raíz) **el primer elemento de la ruta**, **accediendo posteriormente a su nodo-i** correspondiente, y así sucesivamente hasta llegar a localizar el nodo-i del fichero referenciado, a partir de aquí una vez cargado este nodo-i en memoria ya se conoce la posición de todos los bloques de datos del fichero.

Consideremos el ejemplo de búsqueda de la ruta de acceso `/usr/ast/mbox`.

En primer lugar, el sistema de ficheros localiza el fichero raíz. En Unix, su nodo-i se localiza en un lugar fijo del disco. Después se busca la primera componente de la ruta de acceso, usr, en el directorio raíz, para encontrar el nodo-i del fichero `/usr`. La localización de un nodo-i a partir de su número es directa, puesto que cada uno de ellos tiene una posición fija dentro del disco. A partir de este nodo-i, el sistema localiza el directorio de `/usr` y busca la siguiente componente (`ast`) en él. Al encontrar el dato de `ast`, obtiene el nodo-i de `/usr/ast`. Con este nodo-i puede encontrar el propio directorio y buscar `mbox`. Puede leer entonces el nodo-i de este fichero y recuperar a través de los apuntadores directos y, en su caso indirectos, todos sus bloques de información.

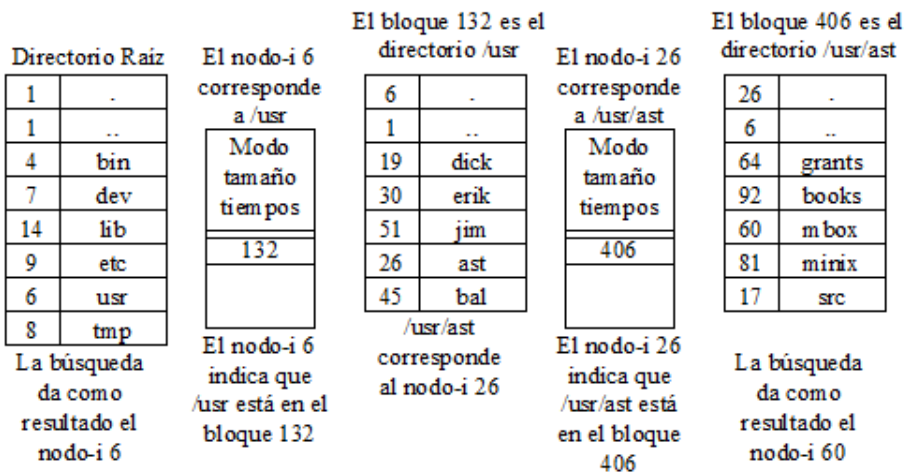


Figura 9. Los pasos en la búsqueda de /usr/ast/mbox.

Los **nombres relativos de las rutas de acceso** se buscan de la misma manera que los nombre absolutos, sólo que se parte del directorio de trabajo en vez de partir del directorio raíz.

Cada directorio tiene entradas para **.** y **..**, colocadas al momento de crearse el directorio. La **entrada . tiene el número de nodo-i para el directorio activo** y l**a entrada .. tiene el número de nodo-i del directorio padre**.

4.3.2 Ficheros Compartidos

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros

Cuando varios usuarios trabajan juntos en un proyecto, necesitan a menudo **compartir ficheros**. Por ello es conveniente contar con la posibilidad de que los ficheros compartidos

aparezcan simultáneamente en directorios diferentes, pertenecientes a usuarios distintos.

La conexión que se establece entre el fichero compartido y el directorio donde se pretende que aparezca dicho fichero se denomina **enlace**. Los sistemas de ficheros dejan así de ser árboles para convertirse en grafos acíclicos dirigidos, o DAG (Directed Acyclic Graph).

Aunque compartir ficheros es una herramienta conveniente, también puede introducir algunos problemas. Para empezar, en sistemas cuyos directorios contengan direcciones de disco, el enlace de un fichero a otro directorio implica la copia de las direcciones de disco del fichero. Si posteriormente se añaden nuevos bloques al fichero compartido, estos bloques aparecerán solamente en el directorio del usuario que realiza la operación. Los demás usuarios no verán ningún cambio, lo que invalida el propósito de tener ficheros compartidos.

Este problema se puede solucionar de dos formas:

- Listar los bloques de disco en una pequeña estructura de datos asociada al propio fichero, en lugar de hacerlo en directorios. Entonces los **directorios apuntarán a la estructura de datos**. Este es el método que utiliza UNIX, donde la estructura de datos mencionada es el nodo-i. A este método se le llama **enlace duro** o físico.
- Introduciendo la **creación de un nuevo fichero de tipo enlace**, y creando una entrada al mismo en su directorio. El nuevo fichero contiene simplemente el nombre de ruta del fichero con el que se quiere enlazar. Cuando se quiere acceder al fichero compartido el sistema operativo ve que el fichero que se quiere leer es de tipo enlace, obtiene el nombre del fichero al que apunta y finalmente lee este último. Este método se conoce como **enlace simbólico**.

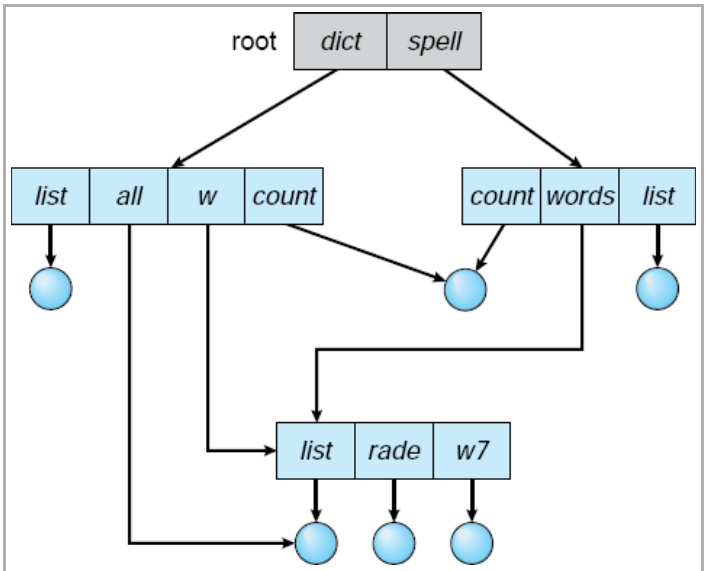


Figura 10. Sistema de ficheros con ficheros compartidos [Silberschatz2010].

En los **enlaces duros**, se nos va a plantear el siguiente dilema: **cuando un usuario quiera borrar un fichero sobre el que se mantiene dos o más enlaces**, ¿se debe borrar el fichero?. La respuesta es no y veamos porqué.

Cuando se establece un enlace, no se modifica en el nodo-i del fichero ningún tipo de dato referente al propietario del fichero, solamente se incrementa la cuenta de enlaces, de forma que el sistema registra el número de entradas de directorio que están apuntando al fichero.

Si posteriormente, uno de los usuarios que están compartiendo el fichero, intenta borrarlo y el sistema operativo así lo hace, se libera el nodo-i asociado al fichero. Por tanto, en uno de los directorios quedará una entrada apuntando a un nodo-i no válido. Incluso si posteriormente, se asigna este nodo-i a un nuevo fichero, el enlace anterior quedaría apuntando a un fichero que no es el correcto.

El problema queda solucionado impidiendo que un fichero sea borrado mientras su contador de enlaces no sea 1. En otro caso, lo único que debe hacer el sistema operativo es eliminar la entrada del directorio que apunta al fichero compartido pero dejar intacto el nodo-i.

Los enlaces simbólicos no tienen este problema porque sólo el propietario del fichero tiene un puntero al nodo-i. Los usuarios que se han enlazado al fichero tienen nombre de ruta, no punteros al nodo-i. Cuando el propietario borra el fichero, éste se destruye. Los intentos

posteriores de acceder al fichero usando los enlaces simbólicos fallarán, al ser incapaz el sistema de encontrarlo. Borrar un enlace simbólico no tiene ningún efecto sobre el fichero.

El inconveniente de los enlaces simbólicos es la cantidad extra de recursos que consumen. En efecto, se debe leer el fichero que contiene el nombre de ruta, que debe interpretarse y recorrerse, elemento a elemento, hasta alcanzar el nodo-i. Toda esta actividad puede requerir un número considerable de accesos extra al disco.

Además, se necesita un nodo-i adicional por cada enlace simbólico y un bloque de disco extra para almacenar el nombre de ruta, aunque si es corto el sistema puede almacenarlo en el propio nodo-i.

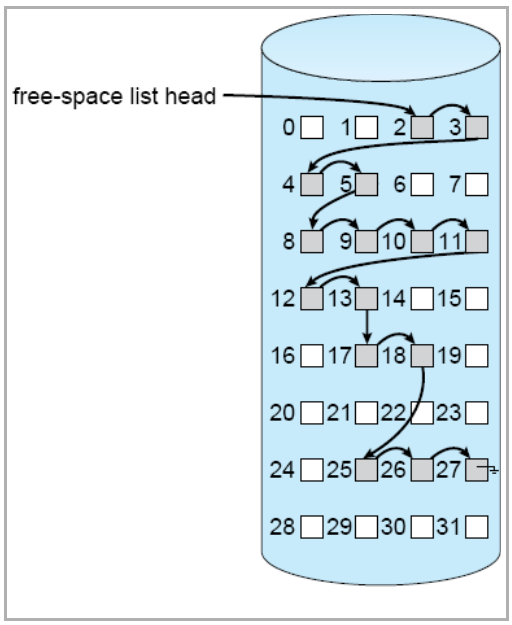
Los enlaces simbólicos tienen la ventaja de poder utilizarse para enlazar ficheros en máquinas de cualquier parte del mundo, dando simplemente la dirección de red de la máquina donde reside el fichero, además de su nombre de ruta en dicha máquina.

Existe también otro problema introducido por los enlaces, sean simbólicos o no. En los programas que empezando en un directorio dado encuentren todos los ficheros en el mismo y en sus subdirectorios, acabarán localizando varias veces cada fichero enlazado (buscar un fichero, realizar estadísticas o copias de seguridad).

4.4 Gestión del Espacio Libre

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Una vez elegido el tamaño de bloque, el siguiente paso es plantearse **cómo llevar la cuenta de los bloques libres**. Consideremos dos posibilidades:

- Utilización de una lista enlazada.
- Mapa de bits.



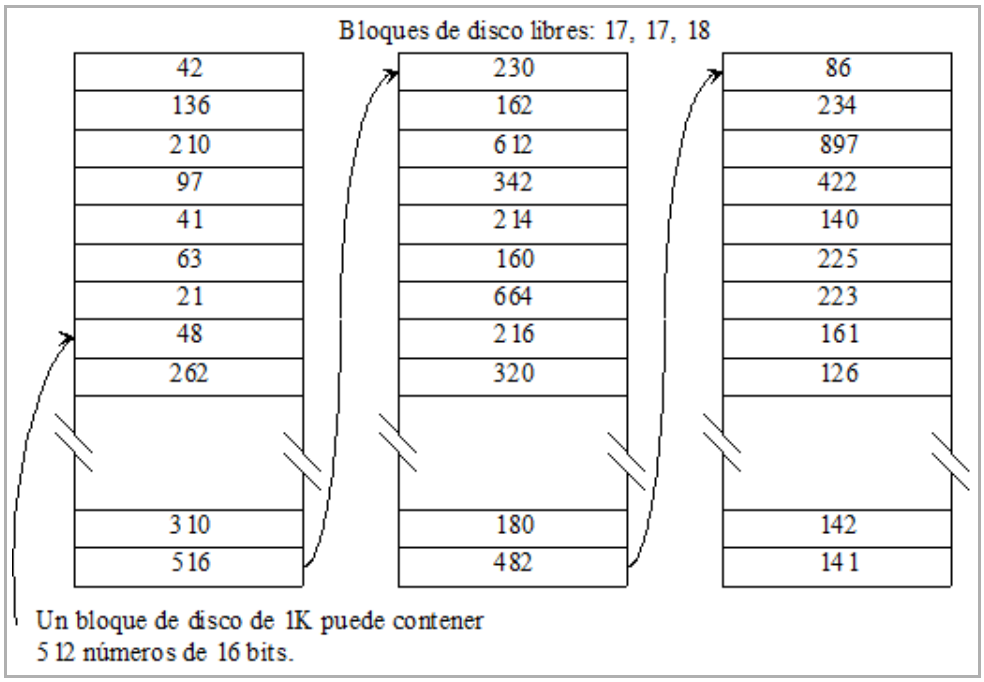


Figura 11. Bloques libres en forma de lista enlazada.
Podemos utilizar una lista enlazada de bloques en disco, donde cada bloque contiene tantos números de bloque de disco libres como le quepan.
Con bloques de 1K y números de bloque de disco de 16 bits, cada bloque en la lista de libres contendrá los números de 511 bloques libres. Un disco de 20M necesitaría una lista de libres de no más de 40 bloques para contener los 20K números de bloques en disco.
Utilizando una lista enlazada obtenemos varias cuestiones positivas:

- no produce fragmentación externa,
- no es costoso de implementar en términos de estructuras de datos,
- facilita las inserciones de nuevos bloques,
- facilita el acceso secuencial,
- facilita la gestión de los bloques defectuosos.

Como contrapartida, también hay que tener en cuenta el precio que hay que pagar:

- produce una pequeña fragmentación interna de medio bloque en promedio,
- el acceso directo es lento, hay que pasar por todos los bloques anteriores,
- un puntero de enlace dañado produce errores graves (un conjunto de bloques de información puede ser inaccesible).

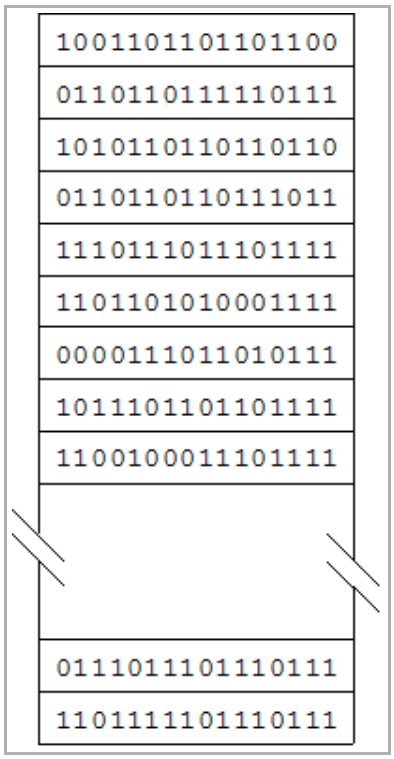


Figura 12. Gestión de bloques libres con mapa de bits.
Utilizando la técnica de **mapa de bits** se pretende mejorar el acceso directo, reuniendo los punteros a bloques de datos asignados en bloques índice, a su vez esta técnica también elimina las diferencias de tiempo entre acceso directo y secuencial.
Si consideramos un disco de n bloques se requieren mapas de bits con n bits. Los bloques libres están representados por 1 en el mapa, mientras los ya asignados se representan por 0 (o viceversa). Un disco de 20M requeriría 20K bits para el mapa, que cabrían en 3 bloques.
No es sorprendente que haga falta menos espacio para el mapa de bits, ya que éste usa sólo 1 bit por bloque, frente a los 16 empleados por el modelo de lista enlazada. Solo si el disco está casi lleno, hará falta menos espacio en el modelo de lista enlazada que en el de mapa de bits.
En general, si hay memoria principal suficiente para contener el mapa de bits, ésta es la técnica de gestión del espacio libre preferida por los diseñadores.

4.5 Fiabilidad del Sistema de Ficheros

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
La **destrucción de un sistema de ficheros suele ser un desastre mucho mayor que la destrucción de un computador**. Si un sistema de ficheros está irremediablemente perdido, sea por el hardware, el software o cualquier otra circunstancia, recuperar la información será difícil, llevará mucho tiempo y, en muchos casos, sencillamente no será posible.
Por otro lado, **los discos suelen tener algunos bloques defectuosos**, concretamente los discos fijos suelen tener bloques ya defectuosos de fabricación: es demasiado caro fabricarlos libres de todo defecto. De hecho, casi todos los fabricantes proporcionan con cada unidad una lista de los bloques defectuosos que han descubierto durante las pruebas.
Existen dos soluciones al problema de los bloques defectuosos, una hardware y otra software.

- La consiste **solución hardware** dedica un sector en el disco a almacenar la lista de bloques defectuosos. Al inicializar el controlador por primera vez, éste lee la lista de bloques defectuosos y selecciona un bloque (o pista) de repuesto para sustituir cada uno de ellos, anotando la correspondencia en la lista de bloques con defecto. A partir de ese momento, todas la peticiones dirigidas a los bloques defectuosos utilizarán los sustitutos respectivos.
- La **solución software** requiere que el usuario o el sistema de ficheros construyan un fichero que contenga todos los bloques defectuosos. De esta forma se eliminarán de la lista de bloques disponibles, por lo que no se usarán nunca en ficheros de datos. Mientras no se lea o escriba en el fichero de bloques defectuosos, no habrá ningún problema. Se

debe poner especial cuidado durante las copias de seguridad del disco para evitar la lectura de este fichero.

4.5.1 Copias de Seguridad

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Aunque se usen métodos sofisticados para evitar los bloques defectuosos, es importante hacer frecuentemente copias preventivas (o de seguridad) de los ficheros.
Realizar **copias de unidades completas de discos fijos** muy grandes es tedioso y hace falta mucho tiempo. Un método fácil de implementar, pero que desperdicia la mitad de la capacidad de almacenamiento, consiste en dotar a cada computador de dos unidades de disco en lugar de una. Ambas unidades están divididas en dos partes: datos y copias de seguridad. Todas las noches, la porción de datos de la unidad 0 se copia en el área de copia de la unidad 1, y viceversa. De esta forma, aunque se estropee completamente una unidad, no se pierde la información.

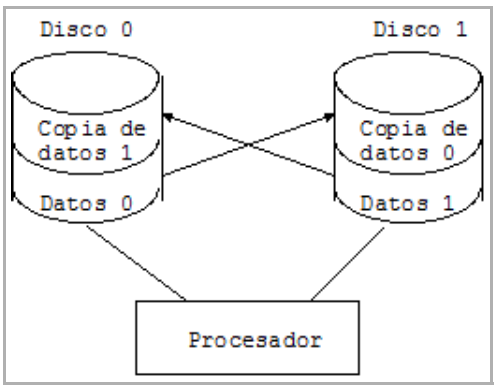


Figura 13. Copias de seguridad.
Una solución al volcado del sistema de ficheros completo puede ser hacer **volcados incrementales**. Es decir, partiendo de un volcado completo, periódicamente se lleva a cabo un volcado de aquellos ficheros que se han modificado desde el último volcado completo. La implementación de este método exige tener una lista en el disco con las fechas de los volcados de cada fichero.

4.5.2 Coherencia del Sistema de Ficheros

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Es fundamental mantener la coherencia del sistema de ficheros. Muchos sistemas de ficheros leen bloques, los modifican y posteriormente los vuelven a escribir en disco. Si el sistema se para antes de que todos los bloques modificados se hayan vuelto a escribir en el disco, el sistema de ficheros puede quedar en un **estado inconsistente**. Este problema es especialmente **grave** cuando los bloques que no se han actualizado son **bloques de nodos-i**, de **directorios** o los que contienen la **lista de bloques libres**.
Para tratar este problema, casi todos los sistemas disponen de un **programa de servicio que comprueba la coherencia interna del sistema de ficheros**. Puede ejecutarse cada vez que se autoarranque el sistema, especialmente después de una caída. Estos comprobadores de sistemas de ficheros verifican cada sistema de ficheros independientemente de los demás. Generalmente se hacen siempre dos tipos de comprobaciones de coherencia: de bloques y de directorios.

Para **comprobar la consistencia de los bloques**, el programa construye una tabla con dos contadores para cada bloque, con valor inicial 0. El primer contador mantiene un registro del número de veces que el bloque está presente en un fichero; el segundo registra el número de veces que está presente en la lista de bloques libres (o

en el mapa de bits de los bloque libres).

El programa lee entonces todos los nodos-i e incrementa el contador correspondiente por cada bloque perteneciente al fichero. El programa examina a continuación la lista o el mapa de bits de los bloque libres. Cada aparición de un bloque hace que se incremente su contador en la segunda tabla.

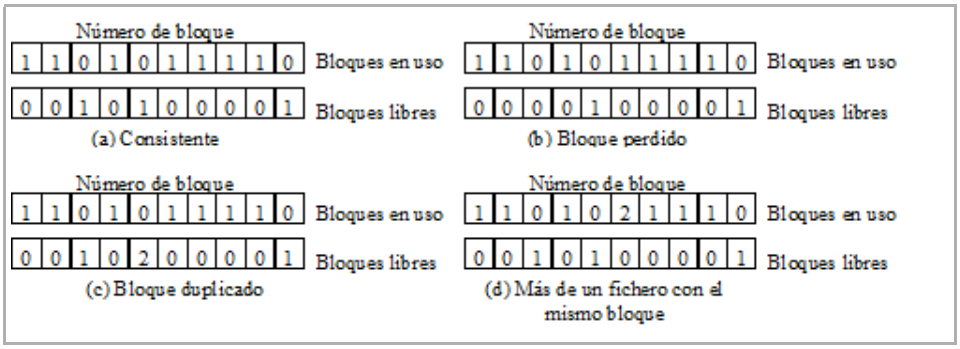


Figura 14. Comprobación de bloques.

Si el sistema de ficheros es consistente, cada bloque tiene un 1 en la primera o en la segunda tabla (ver Figura 14 (a)).

En la Figura 14 (b), el bloque 2 no aparece en ninguna de los contadores. Se debe informar de que ese bloque está perdido (se está desperdiciando su espacio) y se añade a la lista de bloques libres.

Otra situación es la mostrada en la Figura 14 (c). Aquí tenemos un bloque, el número 4, que aparece dos veces en la lista de bloques libres. La solución de este caso es simple: se reconstruye la lista de bloques libres.

Lo peor que puede ocurrir es que el mismo bloque de datos aparezca en uno o más ficheros (Figura 14 (d) bloque 5). Si se elimina cualquiera de estos ficheros, el bloque 5 se coloca en la lista de libres, lo que produce una situación en la que el mismo bloque está en uso y libre al mismo tiempo. Si se eliminan ambos ficheros, el bloque se coloca dos veces en la lista de libres. La acción adecuada en este caso es que el verificador del sistema de ficheros asigne un bloque libre, copie en él el contenido del bloque 5 e inserte la copia en uno de los ficheros. Debe informarse del error para que el usuario pueda inspeccionar el daño.

Por último, otra posibilidad es que un bloque esté presente en un fichero y en la lista de bloques libres. La solución de este caso también es sencilla: se elimina de la lista de bloques libres.

Además de verificar si cada bloque está contado correctamente, el verificador del sistema de ficheros debe **comprobar también el sistema de directorios**. También en este caso utiliza una tabla de contadores, pero en este caso son por directorio y no por bloque. Inicia en el directorio raíz y desciende de forma recursiva el árbol, para inspeccionar cada uno de los directorios del sistema de ficheros. Para cada uno de los ficheros de cada directorio, incrementa el contador del nodo-i de dicho fichero.

Al terminar de hacer esto, se tiene una lista en la que el índice es el número de nodo-i, que indica el **número de directorios que apuntan a ese nodo-i**. Compara entonces estos números con los **contadores de enlaces almacenados** en los propios nodos-i. En un sistema consistente de ficheros, ambos contadores deben coincidir. Sin embargo, pueden aparecer dos tipos de errores: el contador de enlaces del nodo-i puede ser superior o inferior al número de entradas del directorio.

Si el **contador de enlaces es mayor que el número de entradas del directorio**, entonces, aunque se eliminaran todos los ficheros de los directorios, el contador sería distinto de cero y no se podría eliminar el nodo-i. Este error no es serio, pero desperdicia el espacio en el disco con ficheros que no se encuentran en ningún directorio. Esto debe arreglarse haciendo que el contador de enlaces en el nodo-i tome el valor correcto. Si es 0, el fichero debe eliminarse.

El otro tipo de error es catastrófico en potencia. Si dos entradas de un directorio se enlazan a un fichero, pero el nodo-i indica que sólo existe un enlace, entonces, al

eliminar cualquiera de estas entradas de directorio, el contador del nodo-i tomará el valor 0. Cuando el contador de un nodo-i toma el valor 0, el sistema de ficheros lo señala como no utilizado y libera todos sus bloques. Esto hará que uno de los directorios apunte hacia un nodo-i no utilizado, cuyos bloques se podrían asignar entonces a otros ficheros. De nuevo, la solución consiste en forzar que el contador de enlaces del nodo-i sea igual al número de entradas del directorio.

Estas dos operaciones se integran en una sola por razones de eficiencia. Es posible hacer otras verificaciones tales como: que el número de un nodo-i no sea mayor al número de nodos-i en el disco, que los nodos-i aparezcan con atributos extraños (por ejemplo, permiso 0007), directorios con muchas entradas, etc.

4.6 Rendimiento del Sistema de Ficheros

2º Grado en Ingeniería en Informática
teoría de Sistemas Operativos
Bloque Entrada Salida y Ficheros: Sistemas de Ficheros
Un punto crítico que afecta al rendimiento de las operaciones de entrada/salida es el acceso a disco. El método común para reducir estos accesos es el empleo de espacios de almacenamiento intermedios, mantenidos por el sistema operativo: **Caché de Disco**. Este tipo de técnica consiste en una estructura software y no tiene nada que ver con las memoria caches empleadas para agilizar las referencias a memoria principal.
La utilización de caché de disco proporcionará ciertas posibilidades añadidas, se pueden citar las siguientes:

- Operaciones de E/S solapadas, en distintas unidades de disco y controladores.
- Acceso rápido a ficheros cuando los bloques de índices se encuentran en memoria.
- Asignación rápida de bloques al mantener porciones de la lista de bloques libres en la memoria.

El empleo de una caché de disco introduce el problema de la escritura diferida y con ello el de mantener la integridad de los datos (UNIX emplea la llamada al sistema SYNC para provocar la escritura periódica a disco de los bloques modificados).
Existen otras técnicas para aumentar el rendimiento de los sistemas de ficheros:

- Planificación del Disco: ordenar la solicitudes de E/S para reducir el tiempo de búsqueda.
- Agrupar el mayor número de bloques de un fichero, preferentemente dentro de una pista o cilindro, para minimizar el movimiento del brazo del disco.
- Distribuir los nodos-i, o equivalentes, de forma que los bloques de nodos-i y los bloques de datos que referencia se encuentren situados en lugares próximos. Esto también minimiza el movimiento del brazo del disco, ya que para la lectura o escritura de la información contenida en un fichero se necesita un acceso al nodo-i y otro al bloque de datos.

Mejoras de este tipo son fáciles de conseguir cuando se definen varios discos lógicos (file systems) en un mismo disco físico.
Otra posibilidad sería llevar un **registro de la posición rotacional**. Al asignar bloques, el sistema intenta colocar los bloques consecutivos en un fichero en el mismo cilindro pero separados para un máximo rendimiento. Así, si un disco tiempo de rotación de 16,67 mseg y un proceso de usuario tarda 4 mseg en solicitar y obtener un bloque del disco, cada bloque debe colocarse al menos a un cuarto de vuelta de su predecesor. A esto se le llama **técnicas de entrelazado**.