TEMA 7: EL SISTEMA DE ARCHIVOS

1. Objetivos y funciones del sistema de archivos

Sistema de archivos: Subsistema del S.O. encargado de la gestión de la memoria secundaria (concretamente del almacenamiento de la información en dispositivos de memoria secundaria).

Se encuentra en los niveles más externos del sistema operativo (más próximos al usuario). Este nivel suministra al usuario el concepto de archivo (una de las abstracciones fundamentales que genera un sistema operativo).

El sistema de archivos es el subsistema que suministra los medios para la organización y el acceso a los datos almacenados en dispositivos de memoria secundaria (disco).

Concepto de archivo: Agrupación de datos que el usuario ve como una entidad (por ejemplo: programa, conjunto de rutinas, resultados de un cálculo, ...). Es la unidad que almacena y manipula el sistema de archivos.

El medio sobre el que se almacenan los archivos se divide en bloques de longitud fija, siendo el sistema de archivos el encargado de asignar un número adecuado de bloques a cada archivo.

Funciones del sistemas de archivos:

- Crear y borrar archivos
- Permitir el acceso a los archivos para que sean leídos o escritos
- Automatizar la gestión de la memoria secundaria
- Permitir referenciar un archivo por su nombre simbólico
- Proteger los archivos frente a fallos del sistema
- Permitir el uso compartido de archivos a usuarios autorizados

2. El sistema de archivos desde el punto de vista del usuario

2.1. Organización de archivos

Los usuarios pueden definir objetos con nombre llamados archivos y que están constituidos por una secuencia de bits, bytes, líneas o registros. Se referencian mediante su nombre.

El sistema operativo suministra una serie de operaciones especiales (llamadas al sistema) para la manipulación de los archivos:

- Como unidad: Apertura, cierre, creación, borrado, copia, ...
- Accediendo al contenido: Lectura y escritura, actualización, inserción, ...

Las organizaciones comunes de archivos (estructuras de archivos) varían de un S.O. a otro. Por ejemplo, en UNIX los archivos son secuencias de bytes y la interpretación de esos bytes no la hace el S.O.

Los tipos de archivos que podemos encontrar en UNIX son:

- Regulares: contienen datos (programa, ...). Se dividen en diferentes tipos según su uso (normalmente se identifican por la extensión).
- Directorios: contienen información sobre los archivos contenidos en el mismo.
- Especiales: dan nombres a los dispositivos de E/S.

2.2. Directorios

Un directorio es una tabla o un archivo (según el sistema) que contiene una entrada por cada archivo contenido en el mismo.

Estructura de directorios:

- Más simple: Un único directorio que contiene todos los archivos de todos los usuarios. Problema: conflictos cuando 2 usuarios utilizan el mismo nombre.
- Mejora: Un directorio por usuario.
- Generalización: Estructura jerárquica del sistema de archivos: árbol de directorios.

3. Diseño del sistema de archivos

3.1. Gestión del espacio de disco

Dos estrategias posibles para el almacenamiento de un archivo de n bytes en un dispositivo de memoria secundaria:

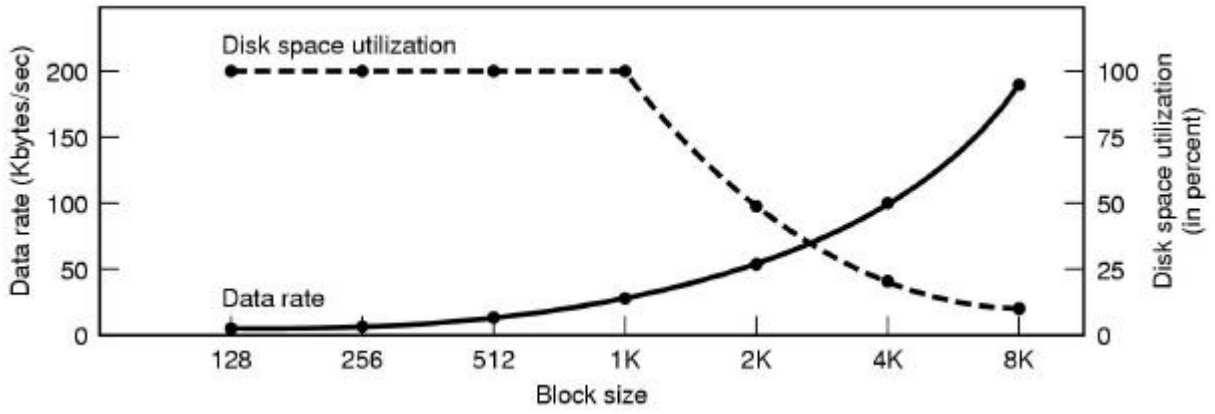
- *Asignación contigua*: Al archivo se le asignan n bytes consecutivos de espacio de disco. Si el archivo crece (hay que tener en cuenta que los archivos son estructuras de datos con una alta volatilidad) probablemente tendrá que ser movido en el disco (análogo a mover segmentos en memoria central, con la diferencia de que esta última operación es más rápida).
- *Asignación no contigua*: El archivo se divide en m bloques de tamaño fijo que se almacenan en disco en bloques no necesariamente contiguos. (Nota: obsérvese la analogía con el caso de la paginación).

La segunda estrategia es la más usual, pero nos encontramos con un primer problema en la gestión del espacio de disco: El establecimiento del tamaño del bloque, es decir, de la unidad mínima de asignación del espacio de disco. Para elegir el tamaño del bloque, se van a tener en cuenta los siguientes datos:

- *Organización física del disco*: Dada la forma en que se organizan los discos, el sector, la pista y el cilindro son candidatos obvios para la unidad de asignación.
- *Aprovechamiento del espacio de disco*: Si se elige una unidad de asignación grande (cilindro) y el tamaño medio de los ficheros es pequeño, se desaprovecha gran cantidad de espacio de disco por fragmentación interna.
- *Ritmo de transferencia de datos*: Si se elige una unidad de asignación pequeña, cada archivo estará dividido en muchos bloques. Como la lectura de cada bloque requiere un tiempo de búsqueda, un tiempo de retraso rotacional y un tiempo de transferencia, la lectura de un archivo (dividido en muchos bloques) será lenta.

Los dos últimos criterios están en contraposición, por lo que la elección del tamaño del bloque deberá ser un compromiso entre los mismos.

En la siguiente figura, se muestra, por un lado, el ritmo de transferencia de datos frente a T (tamaño del bloque), observándose que este es mayor a medida que aumenta T. También se ha representado en la misma gráfica el porcentaje de utilización del espacio de disco, suponiendo un tamaño de medio de archivo de 1 KB. Se observa que, a medida que aumenta el tamaño del bloque, el porcentaje de utilización del espacio de disco disminuye (mayor fragmentación interna).



Surge pues un conflicto inherente entre la eficiencia del tiempo y del espacio (una buena utilización del espacio de disco implica ritmos de transferencia de datos bajos y viceversa). Compromiso normal: Se elige el tamaño de bloque de 512 B, 1 KB o 2 KB. Si se elige un tamaño de bloque de 1 KB sobre un disco con un tamaño de sector de 512 Bytes, el sistema de archivos leerá o escribirá dos sectores consecutivos y los tratará como una unidad simple e indivisible (hay que tener en cuenta que el sector es la unidad más pequeña que se puede leer o escribir en el disco).

Conversión de un número de bloque en una dirección de disco

Podemos ver un disco como un espacio tridimensional de sectores (referenciados mediante una dirección de disco que se establece mediante el cilindro, la pista dentro del cilindro y el sector dentro de la pista). El sistema de

archivos (software de E/S independiente del dispositivo) trabaja con números de bloque lineales (correspondientes a un espacio unidimensional de bloques de disco), entendiéndose que la transformación de la dirección de bloque lineal en tridimensional es realizada en otros niveles de software inferiores (drivers).

En el caso de que el bloque fuera del mismo tamaño que el sector, la transformación se realizaría de la siguiente manera: Las direcciones de bloque aumentan a través de todos los sectores de una pista, luego a través de todas la pistas (superficies) de un cilindro y, por último, del cilindro 0 al ultimo cilindro.

De esta forma, el número de bloque de un sector **s** en la pista **p** del cilindro **c** es:

$$b = s + ns * p + ns * np * c$$

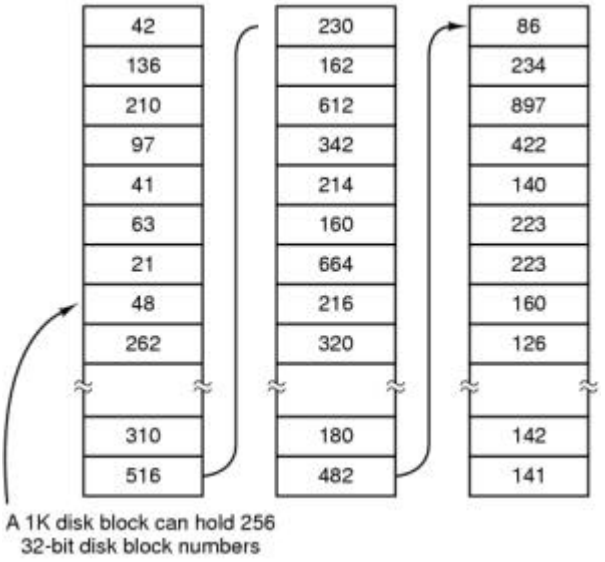
donde np es el número de superficies (pistas por cilindro) y ns es el número de sectores por pista.

Ejercicio: Dado un disco con 4 platos de doble cara y 20 sectores/pista, calcular la dirección física de disco (tridimensional) correspondiente al número de bloque lineal 850.

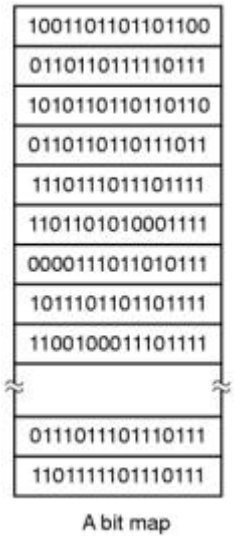
Métodos para llevar registro de los bloques libres

La primera operación que implica la gestión de un recurso consiste en llevar registro de la utilización del recurso. Para el caso de la memoria secundaria (disco), se deberá llevar el registro de los bloques de disco que están libres y de los que están asignados a los diferentes archivos. Vamos a empezar por el registro de los bloques de disco libres, para lo cual hay dos estrategias posibles:

- Usar una lista encadenada de bloques de disco: Cada bloque de la lista libre contiene direcciones de bloques de disco que están libres. Por ejemplo, si los bloques son de 1 KB y las direcciones de bloques son de 32 bits, cada bloque de la lista libre podrá contener hasta 256 direcciones de bloques libres. Por tanto, un disco de 20 MB necesitaría 80 bloques para la lista libre en el caso de que no contuviese ningún archivo.



- Usar un mapa de bits: Un disco con n bloques requiere un mapa de n bits. Los bloques libres se representan con un 0 y los asignados con un 1. Por ejemplo, un disco de 20 MB, con bloques de 1 KB y direcciones de disco de 32 bits, requiere 3 bloques de disco para el mapa de bits.



El mapa de bits suele resultar más conveniente si hay suficiente memoria central. Si sólo se dispone de un bloque de memoria central para llevar el registro de los bloques libres y el disco está casi lleno, la lista libre es más conveniente (con un mapa de bits puede suceder que en ese bloque no haya ningún bit a 0, que identifique un bloque libre, por lo que habrá que acceder al disco para leer el resto del mapa de bits).

3.2. Almacenamiento de archivos

El sistema de archivos debe llevar registro no sólo de los bloques de disco libres sino también de los bloques asignados a cada archivo. Un archivo se divide en bloques de tamaño fijo que se almacenan en bloques de disco no necesariamente contiguos.

El sistema de archivos debe registrar la dirección de disco donde se encuentra almacenado cada bloque lógico de cada archivo.

Algunas estrategias posibles son las que se indican a continuación (para cada una se presenta una discusión del rendimiento de las mismas en accesos aleatorios a un byte concreto):

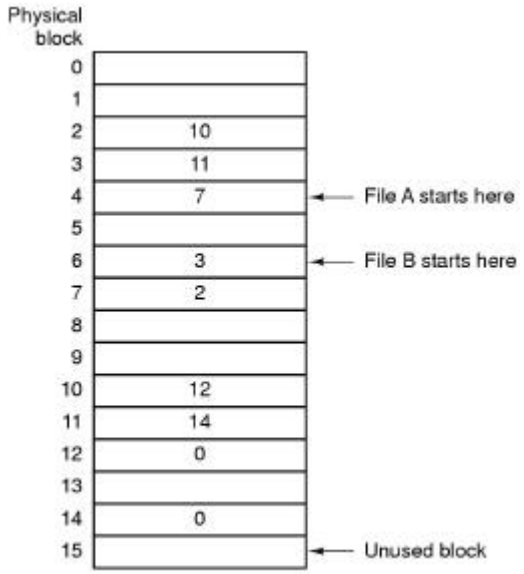
- *Lista encadenada de bloques*: En la entrada del directorio correspondiente al archivo, se guarda la dirección del primer bloque de disco. Cada bloque contiene (al final) la dirección del siguiente bloque de la cadena o una marca de fin de archivo (EOF).

  Ejemplo: si tenemos bloques de 1 KB y direcciones de disco de 32 bits, cada bloque contiene 1020 bytes de datos y un puntero de 4 bytes al siguiente bloque.

  Problema: el acceso al archivo debe ser secuencial. Si un programa busca el byte 32768 del archivo (bloque lógico 32, desplazamiento 128), el S.O. tiene que acceder a 33 bloques de disco (del 0 al 32) para encontrar el dato solicitado (las búsquedas directas son ineficientes).

- *Tabla de asignación de archivos (FAT)*: A cada disco se asocia una tabla (FAT) con una entrada por cada bloque de disco. Cada entrada de la FAT puede contener: una dirección de disco, una marca de bloque libre (indicativo de que el bloque no está asignado), una marca de bloque defectuoso o una marca de fin de archivo (EOF).

  Ejemplo: En la siguiente figura se ilustra la forma de registrar el almacenamiento de dos archivos A y B mediante una FAT. En la entrada al directorio para el archivo A tenemos un 4 (dirección de disco donde está almacenado el bloque lógico 0) y para el archivo B tenemos un 6.



Este esquema de la FAT es originario del sistema operativo MS-DOS. En los disquetes de 320 KB, con tamaño de bloque de 1 KB y números de bloque de 12 bits, la FAT tenía 320 entradas y ocupaba 480 bytes (se podía almacenar en un sector de 512 bytes).

A partir de la versión 2.0, los disquetes se pueden formatear a 360 KB, ocupando la FAT 540 bytes (ya se necesitan dos sectores).

En discos duros de más de 4096 bloques, el número de bloque de 12 bits resulta insuficiente para poder direccionar todos los bloques del disco (hay que aumentar el tamaño de cada entrada de la FAT).

Para discos grandes este esquema es poco atractivo. Por ejemplo, en un disco de 64 MB con bloques de 1 KB y direcciones de disco de 2 bytes, la FAT ocupa 128 KB. Vamos a calcular el número de accesos a disco necesarios para acceder al byte 32768 (bloque lógico 32, desplazamiento 0), en los casos siguientes:

- La FAT se guarda entera en memoria: Se sigue la secuencia de bloques en memoria hasta localizar la dirección de disco correspondiente al bloque lógico 32. El problema es que se gasta mucha memoria central para almacenar la FAT.

- La FAT se guarda en disco (sólo se carga en memoria central cuando se necesita):

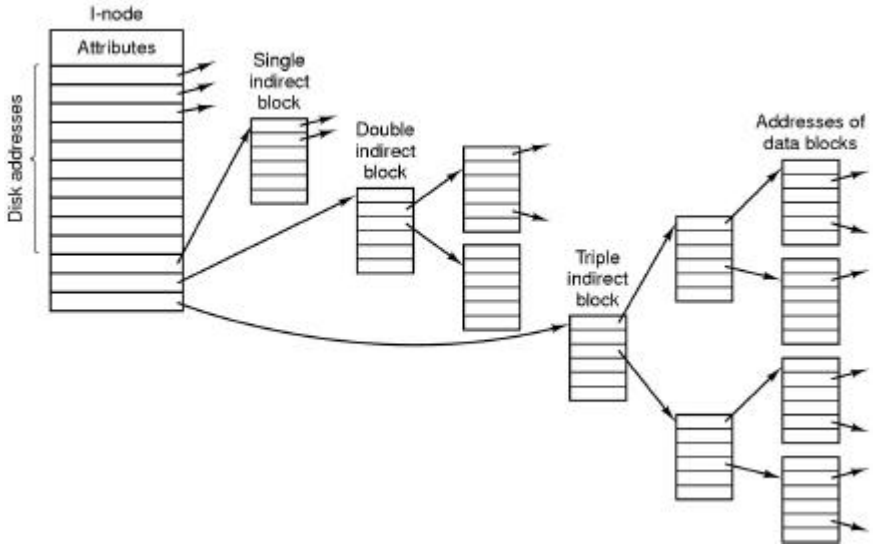
    - Situación más favorable: Las entradas de la FAT correspondientes a los bloques lógicos del 0 al 31 se encuentran en el mismo bloque de la FAT (hay que tener en cuenta que en cada bloque de la FAT puede haber hasta 512 entradas). Por tanto, se necesita un único acceso al bloque correspondiente de la FAT (nos permite localizar la dirección de disco en la que se encuentra el bloque lógico 32), mas otro acceso al bloque de datos.

    - Situación más desfavorable: Las entradas de la FAT correspondientes a los bloques lógicos del 0 al 31 se encuentran en bloques diferentes de la FAT. Se necesitan, por tanto, 32 accesos a disco (uno para cada bloque de la FAT) para localizar la dirección de disco en la que se encuentra el bloque lógico 32, mas el acceso al bloque de datos.

  Por tanto, se requieren entre 1 y 33 accesos a disco, según el grado de fragmentación del archivo en el disco.

  El problema de la FAT radica en que los punteros para todos los archivos almacenados en el disco están mezclados al azar en la misma tabla, requiriéndose toda la tabla aún cuando sólo se quiera abrir un archivo.

- *Bloques de índices (nodos-índices)*: A diferencia de la FAT, se guardan las listas de bloques de los diferentes archivos en estructuras de datos independientes. Esta estrategia es la que se utiliza en el S.O. UNIX, donde a cada archivo se le asigna una tabla llamada nodo índice (nodo-i) que contiene la siguiente información: tipo de nodo, número de enlaces al archivo, identificación de usuario y de grupo del propietario, tamaño del archivo, fecha/hora de creación, del último acceso y de la última modificación, las 10 primeras direcciones de bloques de disco, y punteros indirectos simple, doble y triple.



Para un archivo que ocupe menos de 10 bloques, no son necesarios los punteros indirectos, puesto que en el nodo-i se registran las direcciones de disco de todos los bloques de datos. Si el archivo crece por encima de los 10 bloques, se adquiere un bloque de disco (bloque indirecto simple) al cual apunta el P.I.S. Dicho bloque contiene direcciones de disco de bloques de datos.

Ejemplo: tamaño de bloque 1 KB, direcciones de disco de 32 bits (el B.I.S. almacena hasta 256 punteros a bloques de datos). Tamaño máximo del archivo que utilice el P.I.S.: 266 bloques. Para archivos de mayor tamaño se necesita el P.I.D., que apunta a un bloque de disco (bloque indirecto doble), que a su vez apunta a bloques indirectos simples (estos últimos apuntan a bloques de datos). El tamaño máximo de un archivo que utilice el P.I.D. es $10 + 256 + 256^2 = 65802$ bloques (64,2 MB). Para archivos mayores se utiliza el P.I.T. (Ejercicio: Calcular el tamaño máximo de un archivo que utilice el puntero indirecto triple).

La ventaja del esquema UNIX radica en que los bloques indirectos sólo se usan cuando se necesitan (no son necesarios para archivos menores de 10 KB). Incluso para archivos grandes, como mucho se necesitan 3 referencias a disco para localizar la dirección de disco de cualquier byte del archivo (sin incluir la referencia a disco para obtener el nodo-i que se guarda en memoria al abrir el archivo y permanece ahí hasta que se cierra).

3.3. Estructuras de directorio

Antes de acceder a un archivo, éste tiene que ser abierto. En la apertura, el S.O. carga en memoria central la información que permite localizar los bloques de disco asignados al archivo.

Los archivos se organizan en directorios, siendo un directorio una tabla o archivo que contiene información acerca de los archivos contenidos en el mismo. Ejemplos de estructuras de directorios:

MS-DOS ⇒ Los directorios son archivos que contienen un número arbitrario de entradas. Cada entrada tiene 32 bytes repartidos de la siguiente forma:

- Nombre del archivo: 8 bytes
- Extensión del archivo: 3 bytes
- Atributos (archivo de lectura, oculto, del sistema, etiqueta de volumen, subdirectorio, ...): 1 byte
- Reservado: 10 bytes
- Hora: 2 bytes
- Fecha: 2 bytes
- Nº del primer bloque se usa como índice de la FAT para localizar los demás bloques de disco): 2 bytes
- Tamaño en bytes: 4 bytes

UNIX ⇒ Los directorios son archivos y en cada entrada al directorio, que tiene un tamaño de 16 bytes, se almacena la siguiente información:

- Nº del nodo-i: 2 bytes
- Nombre del archivo: 14 bytes

Toda la información del tipo de archivo, tamaño, propietario, bloques de disco, ... está contenida en el nodo-i.

Organización del directorio

En el directorio se deben poder insertar entradas, eliminar entradas, buscar una entrada concreta y listar todas las entradas. Por tanto, se pueden utilizar las siguientes estructuras de datos para la organización del directorio:

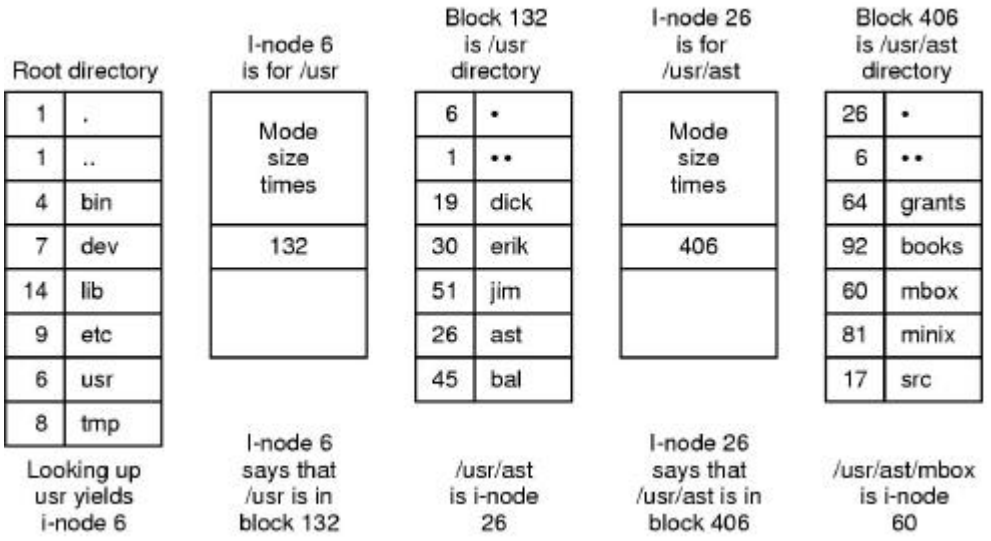
- Lista lineal de entradas:
  - Se requiere una búsqueda lineal para localizar una entrada particular (fácil de programar, pero consume mucho tiempo)
  - Creación de un archivo:
    1. Se busca en la lista para ver si hay otro archivo con el mismo nombre.
    2. Se añade una nueva entrada al final del directorio.
  - Eliminación de un archivo:
    1. Se busca en la lista hasta encontrar la entrada y se libera el espacio que tenía asignado.
    2. Reutilización de la entrada. Posibilidades:
       - Marcarla como no usada (asignarle un nombre especial o activar un bit de la entrada).
       - Colocarla en una lista de entradas de directorio libres.
       - Copiar la última entrada del directorio en la localización liberada y decrementar la longitud del directorio.
- Lista ordenada de entradas:
  - Búsqueda binaria para encontrar un archivo (algoritmo más complejo de programar, pero más eficiente).
  - La lista debe estar ordenada (se complica la creación y borrado de archivos puesto que hay que mover mucha información para mantener un directorio ordenado).

- Tabla “hash”:
  - Mejora el tiempo de búsqueda.
  - Las inserciones y las eliminaciones son directas, aunque hay que tratar el problema de las colisiones o sinónimos propio de las tablas hash.
  - Inconvenientes: tamaño fijo de la tabla hash.

Apertura de un archivo

Al abrir un archivo, el sistema de archivos debe tomar el nombre del archivo suministrado y localizar sus bloques de disco. En sistemas tipo UNIX, se lee el nodo-i de ese archivo y se guarda en memoria central hasta que se cierra el archivo.

Ejemplo: Pasos seguidos para la apertura del archivo /usr/ast/mbox:



- El nodo-i del directorio raíz (nodo-i número 1) se localiza en una posición fija del disco.
- Se lee un bloque del directorio raíz.
- Se busca en dicho bloque la entrada correspondiente a /usr, localizándose su número de nodo-i (en este caso, el 6).

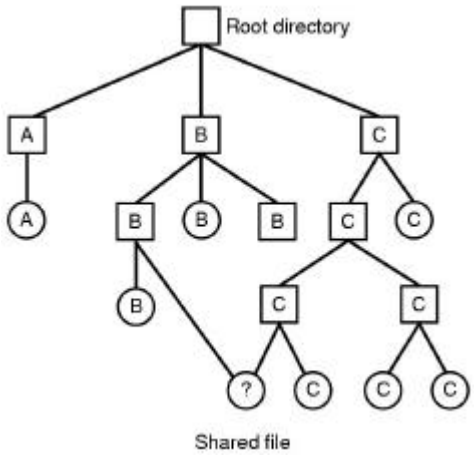
- A continuación, se lee el nodo-i número 6. Entre otra información, éste contiene el o los bloques de disco asignados a /usr (bloque número 132).
- Se accede a disco para leer el bloque número 132. En dicho bloque se localiza la entrada correspondiente a /usr/ast y su nodo-i asociado (el 26).
- Se lee el nodo-i número 26, donde se registran los bloques de disco asignados a /usr/ast (en este caso, el 406).
- A continuación, se lee el bloque de disco 406 y, dentro de éste, se busca la entrada correspondiente a /usr/ast/mbox, localizándose su nodo-i (el 60).
- Se lee el nodo-i número 60, que es el asociado al archivo que se quiere abrir.

Las búsquedas de nombres relativos se realizan de forma similar, sólo que partiendo del directorio actual en lugar del directorio raíz.

Nota: Las entradas . y .. son añadidas al crear el directorio y no se pueden eliminar. La entrada . hace referencia al directorio actual y la .. al directorio padre.

3.4. Archivos compartidos

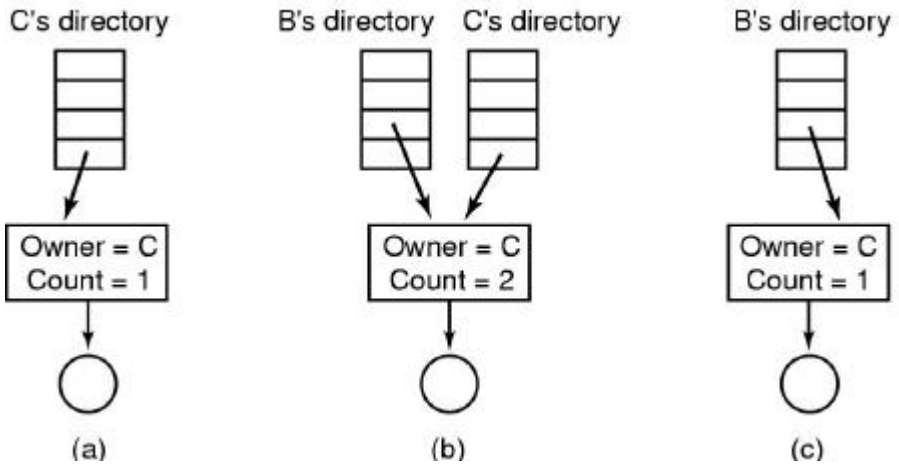
Cuando varios usuarios trabajan en un mismo proyecto, puede ser necesario que compartan archivos. Cada archivo compartido debe aparecer de forma simultánea en diferentes directorios. (Nota: un archivo compartido no es lo mismo que dos copias del archivo). Ejemplo:



A la conexión entre el directorio B y el archivo compartido se le llama enlace (link). De esta forma, el sistema de archivos deja de ser un árbol para pasar a ser un grafo acíclico dirigido.

Para compartir archivos en UNIX se pueden plantear dos opciones:

1) Como los bloques de disco se listan en una pequeña estructura de datos asociada al archivo, llamada nodo-i, podemos hacer que más de un directorio apunte a la misma estructura de datos. Ejemplo:



Inicialmente, el directorio del usuario C contiene una entrada al nodo-i correspondiente a un archivo cuyo propietario es C y la cuenta de enlace es 1.

Si se crea un enlace desde el directorio del usuario B a ese mismo archivo (compartido), el nodo-i sigue manteniendo a C como propietario e incrementa en 1 la cuenta de enlace (pasa a ser 2).

Problema: El propietario decide eliminar el archivo y la cuenta de enlace es mayor que 1:

- Si el sistema de archivos destruye el nodo-i, B tendrá un puntero a un nodo-i erróneo, que puede ser reasignado posteriormente a otro archivo.

- Si el propietario elimina el archivo y éste no se borra, surgen problemas si el S.O. asigna cuotas de espacio de disco a los usuarios.

2) Establecer un enlace simbólico: B se enlaza a un archivo de C haciendo que el sistema cree un nuevo archivo de tipo LINK y lo introduzca en el directorio B. Ese archivo contendrá la ruta del archivo enlazado.

   Sólo el propietario tendrá un puntero al nodo-i, de forma que si éste decide eliminar el archivo, se destruirá su nodo-i (el sistema debe tener en cuenta la posibilidad de que, al eliminar un archivo compartido, se cree otro con el mismo nombre).

   Inconvenientes de los enlaces simbólicos:

   - El archivo que contiene la ruta debe ser leído, requiriéndose un número extra de accesos a disco.
   - Se necesita un nodo-i extra y un bloque de disco para cada enlace simbólico.
   - Complican el diseño de programas que procesan todo el árbol de directorios (copias de seguridad, localización de un archivo, estadísticas acumulativas de archivos, ...), puesto que un archivo con múltiples enlaces se considerará varias veces.

3.5. Integridad del sistema de archivos

La destrucción de una computadora puede resultar desastrosa por el coste que conlleva, aunque sería fácilmente reemplazable.

Sin embargo, la destrucción de un sistema de archivos suele tener consecuencias catastróficas puesto que la restauración de la información puede resultar difícil, consumir mucho tiempo o, sencillamente, ser imposible.

Es evidente que el sistema de archivos no puede ofrecer protección contra la destrucción física del equipo (terremoto, accidente, ...), pero sí puede ayudar a proteger la información almacenada.

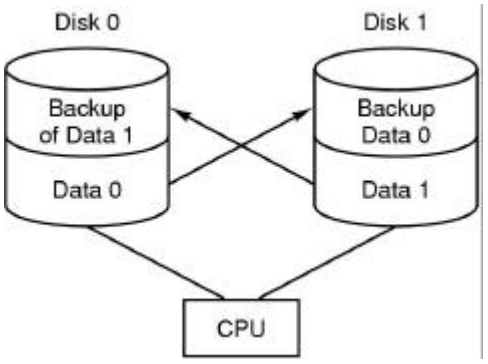
A menudo, los discos tienen sectores defectuosos (a veces, incluso, desde su creación). Dos formas de gestionar este problema:

- Hardware: Dedicar un bloque del disco a contener la lista de bloques defectuosos. Al inicializar el controlador, éste lee la lista de bloques defectuosos.
- Software: El sistema de archivos construye un archivo que contenga todos los bloques defectuosos. Al sacarse estos bloques de la lista libre, no podrán ser asignados a ningún otro archivo. Hay que tener cuidado de no leer este archivo especial de bloques defectuosos al hacer una copia de seguridad.

Copias de seguridad (backups)

Son mecanismos de salvaguarda y recuperación de la información frente a fallos del hardware o del software. Evolución de las estrategias de copias de seguridad:

- Sistemas de archivos sobre disquetes: Se copia el disco entero en uno en blanco.
- Discos Winchester pequeños: Se copia el disco entero en cinta magnética.
- Discos Winchester grandes (500 MB): La copia de seguridad de la unidad entera en cinta es difícil y consume mucho tiempo $\Rightarrow$ no se puede hacer muy a menudo $\Rightarrow$ puede contener archivos que estén obsoletos. Soluciones:
    - La computadora dispone de dos unidades de disco, cada una de ellas dividida en dos mitades, una para datos y otra para la copia de seguridad de la otra unidad.



Cada noche, los datos de una unidad se copian a la parte correspondiente a la copia de seguridad de la otra unidad, de forma que, si una unidad se estropea, no se pierde la información. El inconveniente es que se desaprovecha la mitad del almacenamiento.

- Copias incrementales: Se hace una copia completa periódicamente (normalmente, por semana o por mes) y, diariamente, se hace una copia de los archivos que hayan sido modificados desde la última vez que fueron salvados (evidentemente, incluye los archivos nuevos).

  Esta estrategia es la que suele utilizarse con mayor frecuencia en la actualidad (necesita conocerse la fecha de grabación y de última modificación de cada archivo, y además el proceso de copia se puede realizar en baja prioridad y en paralelo con otras tareas habituales).

Consistencia del sistema de archivos

Los sistemas de archivos leen bloques, los modifican y, a continuación, los reescriben en el disco. Si se produce un fallo del sistema antes de que todos los bloques modificados hayan sido grabados en el disco, el sistema de archivos puede quedar en un estado inconsistente. El problema es crítico si los bloques no grabados son bloques de nodos-i, bloques de directorios o bloques que contienen la lista libre.

Muchas computadoras disponen de un programa que comprueba la consistencia del sistema de archivos y que se ejecuta después de producirse un fallo en el sistema.

Ejemplo: Verificador de la consistencia de sistemas de archivos de UNIX (**fsck**). Este programa verifica cada sistema de archivos efectuando dos comprobaciones de consistencia (de bloques y de archivos):

Consistencia de bloques

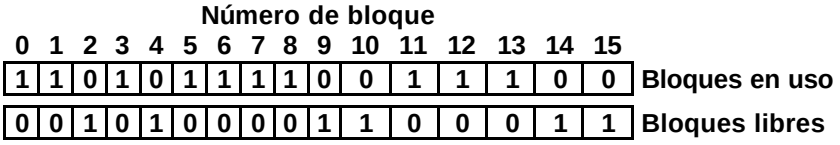
El programa construye una tabla con dos entradas (contadores) por bloque (ambas inicializadas a cero).

A continuación, el programa lee todos los nodos índices para averiguar los números de bloques usados en el correspondiente archivo (cada vez que se lee un bloque, se incrementa su primer contador).

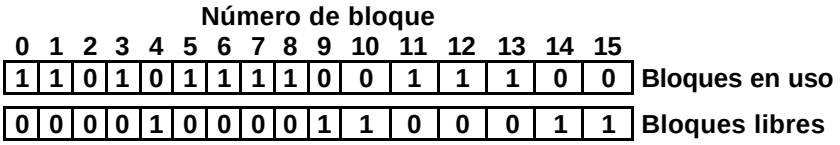
Luego, el programa examina la lista libre para encontrar todos los bloques que no están en uso (por cada ocurrencia de un bloque en la lista libre, se incrementa su segundo contador).

Al examinar esta tabla, el programa ve si el sistema de archivos está en un estado consistente o no. Los estados posibles del sistema de archivos son:

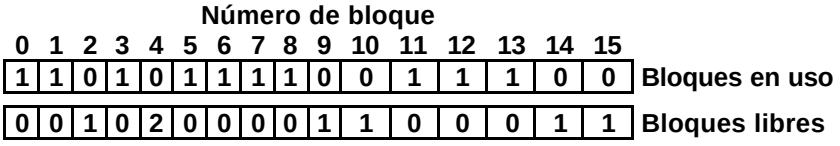
- Consistente: Cada bloque o está libre o está en uso sólo una vez.



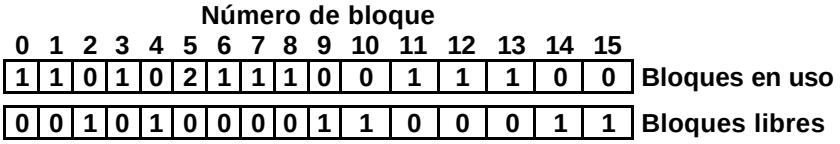
- Bloque perdido: El bloque 2 ni aparece en la lista libre ni está asignado. Los bloques perdidos gastan espacio ⇒ el verificador los añade al final de la lista libre.



- Bloque duplicado en la lista libre: El segundo contador del bloque 4 tiene un valor mayor que 1 ⇒ el programa verificador reconstruye la lista libre para eliminar una de las dos entradas. (No ocurre con un mapa de bits).



- Bloque de datos duplicado: El primer contador del bloque 5 tiene un valor mayor que 1 ⇒ el programa verificador coge un bloque de la lista libre, copia en él los contenidos del bloque duplicado e inserta la copia en uno de los archivos (el contenido permanece inalterable).



Consistencia de archivos

El programa construye una tabla con dos contadores por archivo (el primero cuenta los directorios que apuntan al nodo-i asociado al archivo y el segundo contiene la cuenta de enlace almacenada en el nodo-i).

El programa verificador comienza en el directorio raíz y, de forma recursiva, desciende por todo el árbol inspeccionando cada directorio del sistema de archivos. Para cada archivo en cualquier directorio, incrementa el primer contador para el nodo-i de ese archivo.

Al finalizar, tiene una lista (indexada por número de nodo-i) que indica el número de directorios que apuntan al nodo-i. Posibles estados del sistema de archivos:

- Consistente: La cuenta de enlace es igual al número de entradas de directorio que apuntan a ese nodo-i.
- La cuenta de enlace es mayor que el número de entradas de directorio ⇒ Si se eliminan los archivos de todos los directorios, la cuenta de enlace no será nula y el nodo-i no será eliminado (se desaprovecha espacio de disco con archivos que no están en ningún directorio). El programa verificador iguala la cuenta de enlace con el valor correcto.
- La cuenta de enlace es menor que el número de entradas de directorio. Cuando la cuenta de enlace sea 0, el sistema de archivos libera los bloques de disco asignados al archivo, pero tendremos directorios apuntando a un nodo-i no usado, cuyos bloques pueden ser reasignados a otros archivos. En este caso, el programa verificador también iguala la cuenta de enlace con el valor correcto.

Ambas comprobaciones de consistencia (de bloques y de archivos) se realizan de forma conjunta por razones de eficiencia, realizándose junto con otra serie adicional de comprobaciones.

3.6. Rendimiento del sistema de archivos

Dos de las diferencias más importantes entre la memoria principal y la memoria secundaria son:

- El acceso a disco es mucho más lento que el acceso a memoria.
- El tiempo de acceso a memoria principal es constante.

Muchos sistemas de archivos se diseñan de modo que intenten reducir el número de accesos a disco necesarios. La técnica del búfer o bloque oculto ("block cache" o "buffer cache") se fundamenta en que el tiempo que se tarda en leer un sector y una pista completa es prácticamente el mismo (varía en ½ rotación).

Para ello, se utiliza una caché, que estará formada por una colección de bloques que pertenecen lógicamente al disco, pero que se guardan en memoria por razones de eficiencia.

Gestión de la caché: Se comprueban todas las demandas de lectura para ver si el bloque requerido está en la "caché". Si está ahí, no se accede al disco. Si el bloque no está en la "caché", primero se trae a la caché y luego se copia a donde se necesite. Si la "caché" está llena, hay que sacar algún bloque y reescribirlo en disco (se pueden utilizar los algoritmos de reemplazamiento vistos en la gestión de memoria en sistemas paginados).

Hay que procurar que los bloques de la caché se actualicen a disco para prevenir fallos del sistema y evitar la inconsistencia del sistema de archivos (principalmente si son bloques de nodos-i, de directorios, ...). Dos soluciones:

- UNIX: Mediante una llamada al sistema (la llamada hace un sync y el proceso que la ejecuta se llama update) que fuerza a todos los bloques modificados a que se graben en disco inmediatamente. Para ello, se crea un proceso que realiza, mediante un bucle infinito, esas llamadas al sistema cada 30 segundos aproximadamente.
- MS-DOS: Se escribe cualquier bloque modificado a disco tan pronto como haya sido escrito.

Otras técnicas para reducir la cantidad de movimientos del brazo del disco son:

- Los bloques que se van a acceder en secuencia se asignan cerca (dentro del mismo cilindro). De esta forma se consigue el acceso secuencial a un archivo leyendo bloques numerados consecutivamente, lo que reduce muy sensiblemente el tiempo entre búsquedas.

- Se puede registrar el almacenamiento del disco en grupos de bloques consecutivos. Por ejemplo, el sistema de archivos puede usar bloques de 1 KB, pero el almacenamiento de disco se asigna en unidades de 2 bloques. Tanto la “caché” como la transferencia de disco usarán bloques de 1 KB.

- Un cuello de botella importante que se produce en sistemas que utilizan nodos-i consiste en que la lectura de un archivo pequeño requiere dos accesos a disco (uno para el nodo-i y otro para el bloque de datos). Por este motivo, se buscan estrategias que permitan reducir el tiempo de búsqueda entre el acceso al nodo-i y el acceso al primer bloque de datos.

Apéndice A. Sistema de archivos de Windows

Windows da soporte a varios sistemas de archivos, incluyendo el sistema FAT que ejecutan en Windows 95, MS-DOS y OS/2. Pero los desarrolladores de Windows también diseñaron un nuevo sistema de archivos, NTFS, que está pensado para alcanzar requisitos de altas prestaciones en estaciones de trabajo y servidores.

Ejemplos de aplicaciones de altas prestaciones incluyen las siguientes:

- Aplicaciones cliente/servidor tales como los servidores de archivos, servidores de computación y servidores de bases de datos.
- Ingeniería intensiva de recursos y aplicaciones científicas.
- Aplicaciones de red para grandes sistemas corporativos.

Características clave de NTFS

NTFS es un sistema de archivos flexible y potente, construido en un modelo de sistema de archivos elegantemente simple.

Las características más notables de NTFS incluyen las siguientes:

Recuperación. Uno de los requisitos más importantes de NTFS es la capacidad de recuperarse frente a errores en el sistema y los fallos de disco. En el caso de dichos fallos, NTFS es capaz de reconstruir volúmenes de disco y devolverlos a un estado consistente. Para ello, utiliza un modelo de procesamiento de transacciones para los cambios en el sistema de archivos (cada cambio significativo se trata como una acción atómica que se realiza de forma completa o no se realiza en absoluto, de forma que, las transacciones que estén en proceso en el momento del fallo, son completadas posteriormente o se deshacen para dejar el sistema como antes de su ejecución). Además, NTFS utiliza almacenamiento redundante para los datos críticos del sistema de archivos, de forma que un fallo en un sector del disco no suponga pérdida de datos correspondientes a la estructura y el estado del sistema de archivos.

Seguridad. NTFS utiliza el modelo de objetos de Windows para forzar la seguridad. Un archivo abierto se implementa como un objeto archivo con un descriptor de seguridad que define sus atributos de seguridad.

Discos y archivos grandes. NTFS soporta discos y archivos muy grandes de forma más eficiente que la mayoría del resto de los sistemas de archivos.

Múltiples flujos de datos. Los contenidos reales de un archivo se tratan como un flujo de bytes. En NTFS es posible definir múltiples flujos de datos para un único archivo. Ejemplo: permite que sistemas Macintosh remotos utilicen Windows para almacenar y recuperar archivos.

Facilidad general de indexación. NTFS asocia una colección de atributos con cada archivo. El conjunto de descripciones de archivo en el sistema de gestión de archivos se organiza como una base de datos relacional, de forma que los archivos se pueden indexar por cualquier atributo.

Volúmenes NTFS y estructura de archivos

NTFS hace uso de los siguientes conceptos de almacenamiento de disco:

- ***Sector.*** La unidad física de almacenamiento más pequeña en el disco. El tamaño de los datos en bytes es una potencia de 2 y casi siempre es 512 bytes.
- ***Agrupación.*** Uno o más sectores contiguos (próximos entre sí en la misma pista). El tamaño de la agrupación en sectores es una potencia de 2.
- ***Volumen.*** Una partición lógica de un disco, formada por una o más agrupaciones y utilizada por un sistema de archivos para asignar espacio. En cualquier momento, un volumen está formado por información del sistema de archivos, una colección de archivos y cualquier espacio restante que se pueda asignar a los archivos. El tamaño máximo de un volumen en NTFS es de 2^{64} bytes.

La agrupación es la unidad fundamental de asignación en NTFS, que no trabaja con sectores. Actualmente, el tamaño máximo de archivo soportado por NTFS es 2^{32} agrupaciones, lo que equivale a un máximo de 2^{48} bytes (una agrupación puede tener como máximo 2^{16} bytes).

El uso de agrupaciones para la asignación hace a NTFS independiente del tamaño de sector físico.

Tamaños de agrupación por omisión para NTFS (dependen del tamaño del volumen):

Tamaño de volumen	Sectores por agrupación	Tamaño de agrupación
= 512 Mbytes	1	512 bytes
512 Mbytes – 1 Gbyte	2	1 K
1 Gbyte – 2 Gbytes	4	2 K
2 Gbyte – 4 Gbytes	8	4 K
4 Gbyte – 8 Gbytes	16	8 K
8 Gbyte – 16 Gbytes	32	16 K
16 Gbyte – 32 Gbytes	64	32 K
> 32 Gbytes	128	64 K

El tamaño de agrupación que se utiliza para un volumen particular lo establece NTFS cuando el usuario solicita dar formato a un volumen.

Estructura de un volumen NTFS

NTFS utiliza un enfoque notablemente simple pero potente para organizar la información de un volumen en el disco. Cada elemento del volumen es un archivo, y cada archivo está formado por una colección de atributos. Incluso el contenido de un archivo se trata como un atributo. Con esta estructura sencilla, bastan unas pocas funciones de propósito general para organizar y gestionar un sistema de archivos.

Estructura de un volumen NTFS:

Sector de arranque de la partición	Tabla maestra de archivos	Archivos del sistema	Área de archivos

- ***Sector de arranque de la partición*** (aunque se llame sector, puede estar formado por un máximo de 16 sectores): Contiene información sobre la estructura del volumen, las estructuras del sistema de archivos, y la información de arranque de inicio y el código.
- ***Tabla maestra de archivos*** (Master File Table, MFT): Contiene información sobre todos los archivos y carpetas (directorios) del volumen NTFS, y la información sobre el espacio disponible no asignado. En esencia, la MFT es una lista de todos los contenidos de este volumen NTFS, organizada como un conjunto de filas en una estructura de base de datos relacional.
- ***Archivos del sistema*** (suele tener 1 Mbyte de longitud): Entre los archivos de esta región se encuentran los siguientes:
  - *MFT2*. Un espejo de las tres primeras filas de la MFT, utilizado para garantizar el acceso a la MFT en caso de un fallo de un único sector.
  - *Archivo registro.* Una lista de pasos de transacciones utilizadas para la recuperación en NTFS.
  - *Mapa de bits de las agrupaciones.* Una representación del volumen, mostrando qué agrupaciones están en uso.
  - *Tabla de definición de atributos.* Define los tipos de atributos soportados en este volumen e indica si se pueden indexar o si se pueden recuperar durante una operación de recuperación del sistema.

Tabla maestra de archivos

El corazón del sistema de archivos Windows es la MFT. La MFT se organiza como una tabla de registros de longitud variable (entre 1 y 4 KB). Cada registro (fila) describe un archivo o una carpeta de este volumen, incluyendo la propia MFT, que se trata como un archivo.

Si un archivo es suficientemente pequeño, se localiza completamente en una fila de la MFT. En otro caso, la fila para dicho archivo contiene información parcial y el resto del archivo se encuentra en otras agrupaciones del volumen.

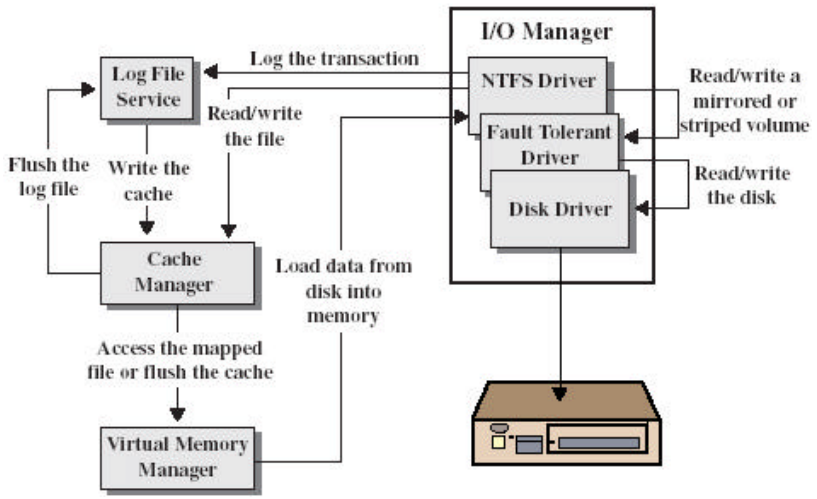
Cada registro de la MFT está formado por un conjunto de atributos que sirve para definir las características del archivo (o carpeta) y sus contenidos.

Atributos que se deben encontrar obligatoriamente en una fila de la MFT:

- Información estándar (permisos, fechas de creación y modificación, …)
- Lista de atributos (utilizado cuando no caben todos los atributos en un registro de la MFT)
- Nombre de archivo (un archivo o carpeta debe tener uno o más nombres)
- Descriptor de seguridad (propietario del archivo y quién puede acceder)
- Datos (contenidos del archivo. Puede haber varios para el mismo archivo)

Recuperación

NTFS hace posible llevar el sistema de archivos a un estado consistente cuando ha habido un error de sistema o un fallo de disco. Los elementos clave que soportan la recuperación son los siguientes:



- ***Gestor de E/S.*** Incluye el controlador NTFS, que gestiona las funciones básicas de apertura, cierre, lectura y escritura de NTFS.
- ***Servicio de archivo de registro.*** Mantiene un registro de las escrituras de disco. El archivo de registro se utiliza para recuperar un volumen con formato NTFS en el caso de un fallo del sistema.

- ***Gestor de caché.*** Responsable del caching de las lecturas y escrituras de archivo para incrementar el rendimiento.
- ***Gestor de memoria virtual.*** NTFS accede a archivos en caché mediante la proyección de las referencias al archivo a las referencias a memoria virtual, leyendo y escribiendo en la memoria virtual.

Los procedimientos de recuperación utilizados por NTFS se diseñan para recuperar los datos del sistema de archivos, no el contenido de los archivos ⇒ un error nunca debe hacer perder un volumen o la estructura de directorios.

El sistema de archivos no garantiza los datos del usuario, puesto que proporcionar una recuperación completa supondría el uso de una herramienta de recuperación más elaborada y con un mayor tiempo de ejecución.

La esencia de la capacidad de recuperación de NTFS se encuentra en el uso de registros (logs). Cada operación que altera un sistema de archivos se trata como una transacción. Cada suboperación de una transacción que altera estructuras de datos importantes del sistema de archivos se graba en un archivo de registro antes de grabarse en el volumen del disco. Mediante este registro, una transacción parcialmente completada en el momento del error se puede rehacer posteriormente o deshacer cuando el sistema se recupere.

En términos generales, estos son los pasos tomados para asegurar la recuperación:

1) NTFS llama al registro del sistema de archivos para grabar en el registro de la caché cualquier transacción que vaya a modificar la estructura del volumen.
2) NTFS modifica el volumen (en la caché).
3) El gestor de la caché llama al registro del sistema de archivos para volcar el archivo de registro al disco.
4) Una vez que el registro se actualiza de forma segura en el disco, el gestor de la caché vuelca los cambios al disco.