Sistema de Archivos

JOAQUÍN ROIZ PAGADOR[1]

[1] *Universidad Pablo de Olavide, Ctra. Utrera 1, 41013, Dos Hermanas, Sevilla, España.*
[*] *Contacto: quiniroiz@gmail.com*

Compiled June 2, 2018

Resumen del temario de la asignatura de Sistemas Operativos del Grado en Ingeniería Informática en Sistemas de Información. © 2018 Joaquín Roiz Pagador

OCIS codes: (130.6750) Systems.

quiniroiz@gmail.com

La capacidad de almacenamiento está restringida por el tamaño del espacio de direcciones virtuales. Para algunas aplicaciones este tamaño es adecuado; para otras, tales como las de reservaciones en aerolíneas, las bancarias o las de contabilidad corporativa, pueden ser demasiado pequeño.

Un segundo problema relacionado con el mantenimiento de la información dentro del espacio de direcciones de un proceso es que cuando el proceso termina, la información se pierde. Para muchas aplicaciones la información se debe retener durante semanas, meses o incluso indefinidamente. Es inaceptable que esta información se desvanezca cuando el proceso que la utiliza termine.

Un tercer problema es que frecuentemente es necesario que varios procesos accedan a (partes de) la información al mismo tiempo.

Tres requerimientos esenciales para el almacenamiento de información a largo plazo:

1. Debe ser posible almacenar una cantidad muy grande de información.
2. La información debe sobrevivir a la terminación del proceso que la utilice.
3. Múltiples procesos deben ser capaces de acceder a la información concurrentemente.

Durante muchos años se han utilizado discos magnéticos para este almacenamiento a largo plazo, así como cintas y discos óptimos, aunque con un rendimiento mucho menor.Un disco sería una secuencia lineal de bloques de tamaño fijo que admite dos operaciones:

1. Leer el bloque k.
2. Escribir el bloque k.

Éstas son operaciones muy inconvenientes, en especial en sistemas extensos utilizados por muchas aplicaciones y tal vez varios usuarios.

1. ¿Cómo encontramos la información?
2. ¿Cómo evitamos que un usuario lea los datos de otro usuario?
3. ¿Cómo sabemos cuáles bloques están libres?

Los **archivos** son unidades lógicas de información creada por los procesos. Un disco contiene miles o incluso millones de archivos independientes.

Los procesos pueden leer los archivos existentes y crear otros si es necesario. La información que se almacena en los archivos debe ser **persistente**, es decir, no debe ser afectada por la creación y terminación de los procesos. Un archivo debe desaparecer sólo cuando su propietario lo remueve de manera explícita.

La parte del sistema operativo que trata con los archivos se conoce como **sistema de archivos** y es el tema de este capítulo.

1. ARCHIVOS

A. Nomenclatura de archivos

Los archivos son un mecanismo de abstracción. Proporcionan una manera de almacenar información en el disco y leerla después. Esto se debe hacer de tal forma que se proteja al usuario de los detalles acerca de cómo y dónde se almacena la información y cómo funcionan los discos en realidad.

La característica más importante de cualquier mecanismo de abstracción es la manera en que los objetos administrados son denominados. Cuando un proceso crea un archivo le proporciona un nombre. Cuando el proceso termina, el archivo continúa existiendo y puede ser utilizado por otros procesos mediante su nombre.

Las reglas exactas para denominar archivos varían un poco de un sistema a otro, pero todos los sistemas operativos actuales permiten cadenas de una a otro letras como nombres de archivos legales.

Muchos sistemas operativos aceptan nombres de archivos en dos partes. La parte que va después del punto se conoce como la **extensión del archivo** y por lo general indica algo acerca de su naturaleza.

B. Estructura de archivos

Los archivos se pueden estructurar en una de varias formas. Tres posibilidades comunes se describen en la Figura **??**. El archivo **??**(a) es una secuencia de bytes sin estructura: el sistema operativo no sabe, ni le importa, qué hay en el archivo. Todo lo que ve son bytes. Cualquier significado debe ser impuesto por los programas a nivel usuario.

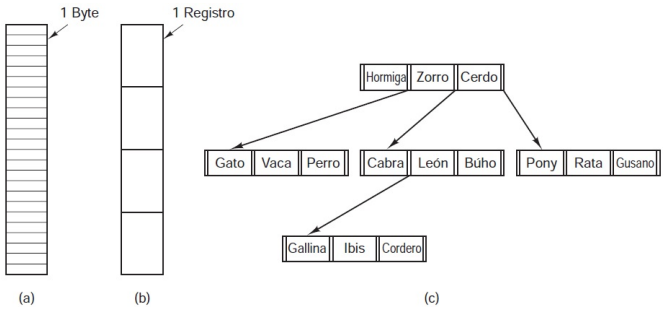

Fig. 1. Tres tipos de archivos. (a)Secuencia de bytes. (b) Secuencia de registros. (c) Árbol

Hacer que el sistema operativo considere los archivos sólo como secuencias de bytes provee la máxima flexibilidad. Los programas de usuario pueden colocar cualquier cosa que quieran en sus carchivo s y denominarlos de cualquier manera conveniente. El sistema operativo no ayuda, pero tampoco estorba.

Los programas leían la entrada en unidades de 80 caracteres y la escribían en unidades de 132 caracteres, aunque los últimos 52 podían ser espacios.

El tercer tipo de estructura de archivos se muestra en la figura **??**(c). En esta organización, un archivo consiste de un árbol de registros,, donde no todos son necesariamente de la misma longitud; cada uno de ellos contiene un campo **llave** en una posición fija dentro del registro. El árbol se ordena con base de campo llave para permitir una búsqueda rápida por una llave específica.

La operación básica aquí no es obtener el "siguiente" registro, aunque eso también es posible, sino obtener el registro con una llave específica. Se pueden agregar nuevos registros al archivo, con el sistema operativo, y no el usuario, decidiendo dónde colocarlos. Evidentemente, este tipo de archivos es bastante distinto de los flujos de bytes sin estructura que se usan en UNIX y Windows, pero se utiliza de manera amplia en las grandes computadoras mainframe que aún se emplean en algún procesamiento de datos comerciales.

C. Tipos de archivos

Los **archivos regulares** son los que contienen información del usuario. Todos los archivos de la figura **??** son archivos regulares. Los **directorios** son sistemas de archivos para mantener la estructura del sistema de archivos. Estudiaremos los directorios un poco más adelante. Los **archivos especiales de caracteres** se relacionan con la entrada/salida y se utilizan para modelar dispositivos de E/S en serie, tales como terminales, impresoras y redes. Los archivos **especiales de bloques** se utilizan para modelar discos.

Un archivo binario ejecutable siempre tomado de una de las primeras versiones de UNIX consta de cinco secciones: encabezado, texto, datos, bits de reubicación y tabla de símbolos. El encabezado empieza con un supuesto **número mágico.**

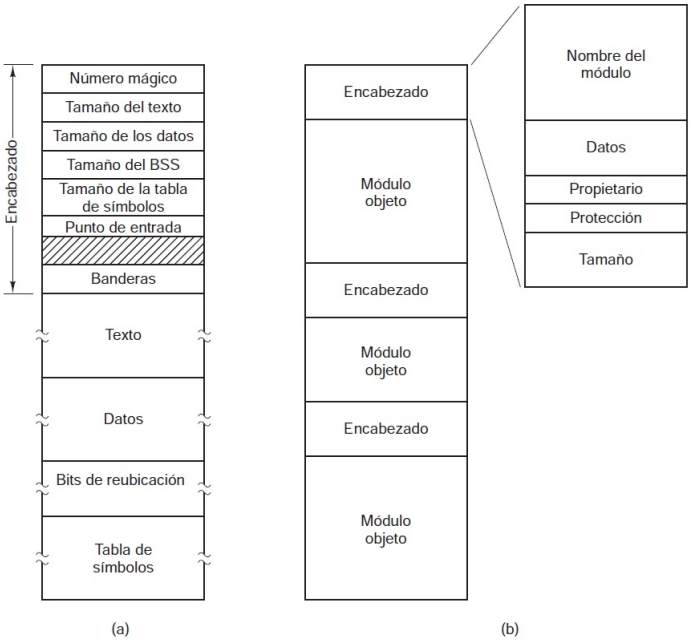

Fig. 2. (a) Un archivo ejecutable. (b) Un archivo.

D. Acceso a archivos

Los primeros sistemas operativos proporcionaban sólo un tipo de acceso: **acceso secuencial**. En estos sistemas, un proceso podía leer todos los bytes o registros en un archivo en orden, empezando desde el principio, pero no podía saltar algunos y leerlos fuera de orden.

Cuando se empezó a usar discos para almacenar archivos, se hizo posible leer los bytes o registros de un archivo fuera de orden, pudiendo acceder a los registros por llave en vez de posición. Los archivos cuyos bytes o registros se pueden leer en cualquier orden se llaman **archivos de acceso aleatorio**. Son requeridos por muchas aplicaciones.

E. Atributos de archivos

Todo archivo tiene un nombre y sus datos. Todos los sistemas operativos asocian otra información con cada archivo; por ejemplo, la fecha y hora de la última modificación del archivo y su tamaño. A estos elementos adicionales les llamaremos **atributos** del archivo. Algunas personas los llaman **metadatos**.

Atributo	Significado
Protección	Quién puede acceso al archivo y en qué forma
Contraseña	Contraseña necesaria para acceder al archivo
Creador	ID de la persona que creó el archivo
Propietario	El propietario actual
Bandera de sólo lectura	0 para lectura/escritura; 1 para sólo lectura
Bandera oculto	0 para normal; 1 para que no aparezca en los listados
Bandera del sistema	0 para archivos normales; 1 para archivo del sistema
Bandera de archivo	0 si ha sido respaldado; 1 si necesita respaldarse
Bandera ASCII/binario	0 para archivo ASCII; 1 para archivo binario
Bandera de acceso aleatorio	0 para sólo acceso secuencial; 1 para acceso aleatorio
Bandera temporal	0 para normal; 1 para eliminar archivo al salir del proceso
Banderas de bloqueo	0 para desbloqueado; distinto de cero para bloqueado
Longitud de registro	Número de bytes en un registro
Posición de la llave	Desplazamiento de la llave dentro de cada registro
Longitud de la llave	Número de bytes en el campo llave
Hora de creación	Fecha y hora en que se creó el archivo
Hora del último acceso	Fecha y hora en que se accedió al archivo por última vez
Hora de la última modificación	Fecha y hora en que se modificó por última vez el archivo
Tamaño actual	Número de butes en el archivo
Tamaño máximo	Número de butes hasta donde puede crecer el archivo

Fig. 3. Algunos posibles atributos de archivos.

F. Operaciones de archivos

1. Create. El archivo se crea sin datos. El propósito de la llamada es anunciar la llegada del archivo y establecer algunos de sus atributos.

2. Delete. Cuando el archivo ya no se necesita, se tiene que eliminar para liberar espacio en el disco. Siempre hay una llamada al sistema para este propósito.

3. Open. Antes de usar un archivo, un proceso debe abrirlo. El propósito de la llamada a open es permitir que el sistema lleve los atributos y la lista de direcciones de disco a memoria principal para tener un acceso rápido a estos datos en llamadas posteriores.

4. Close. Cuando terminan todos los accesos, los atributos y las direcciones de disco ya no son necesarias, por lo que el archivo se debe cerrar para liberar espacio en la tabla interna. Muchos sistemas fomentan esto al imponer un número máximo de archivos abiertos en los procesos. Un disco se escribe en bloques y al cerrar un archivo se obliga a escribir el último bloque del archivo, incluso aunque ese bloque no esté lleno todavía.

5. Read. Los datos se leen del archivo. Por lo general, los bytes provienen de la posición actual. EL llamador debe especificar cuántos datos se necesitan y también se debe proporcionar un búfer para colocarlos.

6. Write. Los datos se escriben en el archivo una y otra vez, por lo general en la posición actual. Si la posición actual es la final del archivo, aumenta su tamaño. Si la posición actual está en medio del archivo, los datos existentes se sobrescriben y se pierden para siempre.

7. Append. Esta llamada es una forma restringida de write. Sólo puede agregar datos al final del archivo. Los sistemas que proveen un conjunto mínimo de llamadas al sistema por lo general no tienen append; otros muchos sistemas proveen varias formas de realizar la misma acción y algunas veces ésos tienen append.

8. Seek. Para los archivos de acceso aleatorio, se necesita un método para especificar de dónde se van a tomar los datos. Una aproximación común es una llamada al sistema de nombre seek, la cual reposiciona el apuntador del archivo en una posición específica del archivo. Una vez que se completa esta llamada, se pueden leer o escribir datos en esa posición.

9. Get attributes. A menudo, los procesos necesitan leer los atributos de un archivo para realizar su trabajo. Por ejemplo, el programa make de UNIX se utiliza con frecuencia para administrar proyectos de desarrollo de software que consisten en muchos archivos fuente. Cuando se llama a make, este programa examina los tiempos de modificación de todos los archivos funte y objeto, con los que calcula el mínimo número de compilaciones requeridas para tener todo actualizado. Para hacer su trabajo, debe analizar los atributos, a saber, los tiempos de modificación.

10. Set attributes. Algunos de los atributos puede establecerlos el usuario y se pueden modificar después de haber creado el archivo. Esta llamada al sistema hace eso posible. La información del modo de protección es un ejemplo obvio. La mayoría de las banderas también caen en esta categoría.

11. Rename. Con frecuencia ocurre que un usuario necesita cambiar el nombre de un archivo existente. Esta llamada al sistema lo hace posible. No siempre es estrictamente necesaria, debido a que el archivo por lo general se puede copiar en un nuevo archivo con el nuevo nombre, eliminando después el archivo anterior.

2. DIRECTORIOS

Para llevar el registro de los archivos, los sistemas de archivos por lo general tienen **directorios** o **carpetas**, que en muchos sistemas son también archivos. En esta sección hablaremos sobre los directorios, su organización, sus propiedades y las operaciones que pueden realizarse con ellos.

A. Sistemas de directorios de un solo nivel

Algunas veces se le llama **directorio raíz**, pero es el único, el nombre no importa mucho. En las primeras computadoras personales, este sistema era común, en parte debido a que sólo había un usuario.

B. Sistemas de directorios jerárquicos

Tener un solo nivel es adecuado para aplicaciones dedicadas simples, pero para los usuarios modernos con miles de archivos, sería imposible encontrar algo si todos los archivos estuvieran en un solo directorio.

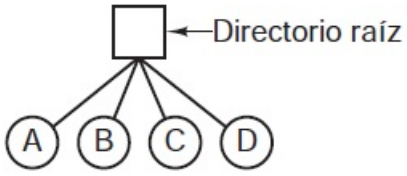

Fig. 4. Un sistema de directorio de un solo nivel que contiene cuatro archivos.

En consecuencia, se necesita una forma de agrupar los archivos relacionados.

Lo que se necesita es una jerarquía. Con este esquema, puede haber tantos directorios como se necesite para agrupar los archivos en formas naturales. Además, si varios usuarios comparten un servidor de archivos común, como se da el caso en muchas redes de empresas, cada usuario puede tener un directorio raíz privado para su propia jerarquía.

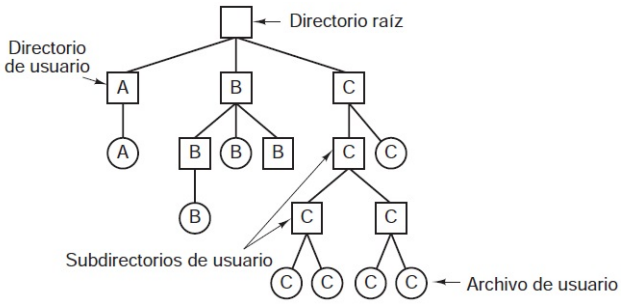

Fig. 5. Un sistema de directorios jerárquico

La capacidad de los usuarios para crear un número arbitrario de subdirectorios provee una poderosa herramienta de estructuración para que los usuarios organicen su trabajo. Por esta razón, casi todos los sistemas de archivos modernos se organizan de esta manera.

C. Nombres de rutas

Cuando el sistema de archivos está organizado como un árbol de directorios, se necesita cierta forma de especificar los nombres de los archivos. Por lo general se utilizan dos métodos distintos. En el primero método, cada archivo recibe un **nombre de ruta absoluto** que consiste en la ruta desde el directorio raíz al archivo.

El otro tipo de nombre es el **nombre de ruta relativa**. Éste se utiliza en conjunto con el concepto del **directorio de trabajo**. Un usuario puede designar un directorio como el directorio de trabajo actual, en cuyo caso todos los nombres de las rutas que no empiecen en el directorio raíz se toman en forma relativa al directorio de trabajo.

Cada proceso tiene su propio directorio de trabajo, por lo que cuando éste cambia y después termina ningún otro proceso se ve afectado y no quedan rastros del cambio en el sistema de archivos.

D. Operaciones de directorios

1. Create. Se crea un directorio.
2. Delete. Se elimina un directorio.
3. Opendir. Los directorios se pueden.
4. Closedir. Cerrar el directorio.
5. Readdir. Devuelve la siguiente entrada en un directorio abierto.
6. Rename. Los directorios son sólo como archivos y se les puede cambiar el nombre de la misma forma que a los archivos.
7. Link. La vinculación es una técnica que permite a un archivo aparecer en más de un directorio. Se le llama algunas veces **vínculo duro** (o **liga dura**).
8. Unlink. Se elimina una entrada de directorio. Si el archivo que se va a desvincular sólo está presente en un directorio, se quita del sistema de archivos. Si está presente en varios directorios, se elimina sólo el nombre de la ruta especificado. Los demás permanecen.

Una variante sobre la idea de vincular archivos es el **vínculo simbólico** (**liga simbólica**). En vez de tener dos nombres que apunten a la misma estructura de datos interna que presenta un archivo, se puede crear un nombre que apunte a un pequeño archivo que nombre a otro.

3. IMPLEMENTACIÓN DE SISTEMA DE ARCHIVOS

La forma en que se almacenan los archivos y directorios, cómo se administra el espacio en el disco y cómo hacer que todo funcione con eficiencia y confiabilidad.

A. Distribución del sistema de archivos

La mayoría de los discos se pueden dividir en una o más particiones, con sistemas de archivos independientes en cada partición. El sector 0 del disco se conoce como el **MBR** y se utiliza para arrancar la computadora. El final del MBR contiene la tabla de particiones, la cual proporciona las direcciones de inicio y fin de cada partición. Una de las particiones en una tabla se marca como activa. Cuando se arranca la computadora, el BIOS lee y ejecuta el MBR. Lo primero que hace el programa MBR es localizar la partición activa, leer su primer bloque, conocido como **bloque de arranque**, y ejecutarlo. El programa en el bloque de arranque carga el sistema operativo contenido en esa partición. Cada partición inicia con un bloque de arranque que no contenga un sistema operativo que se pueda arrancar.

A menudo el sistema de archivos contendrá algunos de los elementos, como el **superbloque**. Contiene todos los parámetros clave acerca del sistema de archivos y se lee en la memoria cuando se arranca la computadora o se entra en contacto con el sistema de archivos por primera vez.

B. Implementación de archivos

B.1. Asignación contigua

El esquema de asignación más simple es almacenar cada archivo como una serie contigua de bloques de disco. Así, e un disco con bloques de 1 KB, a un archivo de 50 KB se le asignarían 50 bloques consecutivos. Con bloques de 2 KB, se le asignarían 25 bloques consecutivos.

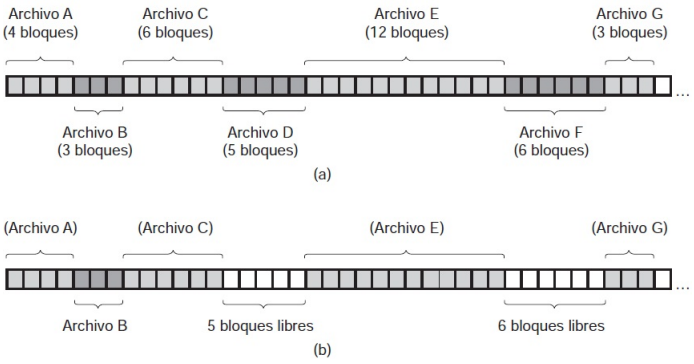

Fig. 6. (a) Asignación contigua de espacio de disco para siete archivos. (b) El estado del disco después de haber removido los archivos D y F.

Dos ventajas significativas. En primer lugar es simple de implementar, ya que llevar un registro de una ubicación de los bloques de un archivo se reduce a recordar dos números: la dirección de disco del primer bloque y el número de bloques en el archivo. Dado el número del primer bloque, se puede encontrar el número de cualquier otro bloque con una simple suma.
En segundo lugar, el rendimiento de lectura excelente debido a que el archivo completo se puede leer del disco en una sola operación. Sólo se necesita una búsqueda. Después de eso, no son necesarias más búsquedas ni retrasos por rotación, por lo que los datos llegan con el ancho de banda completa del disco. Por ende, la asignación contigua es simple de implementar y tiene un alto rendimiento.

Por desgracia, la asignación contigua también tiene una desventaja ligeramente significante: con el transcurso del tiempo, los discos se fragmentan.

Al principio esta fragmentación no es un problema, ya que cada nuevo archivo se puede escribir al final del disco, después del anterior. En un momento dado el disco se llenará y será necesario compactarlo, lo cual es un extremo costoso o habrá que reutilizar el espacio libre de los huecos. Para reutilizar el espacio hay que mantener una lista de huecos, lo cual se puede hacer. Sin embargo, cuando se cree un nuevo archivo será necesario conocer su tamaño final para poder elegir un hueco del tamaño correcto y colocarlo.

Hay una situación en la que es factible la asignación contigua: en los CD-ROMs.

B.2. Asignación de lista enlazada (ligada)

Mantener cada uno como una lista enlazada de bloques de disco, como se muestra en la figura **??**. La primera palabra de cada bloque se utiliza como apuntador al siguiente.

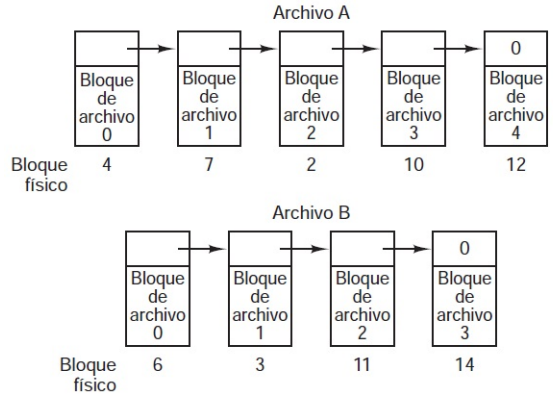

Fig. 7. Almacenamiento de un archivo como una lista enlazada de bloques de disco.

En este método se puede utilizar cada bloque del disco. No se pierde espacio debido a la fragmentación del disco. Para la entrada del directorio sólo le basta con almacenar la dirección del disco del primer bloque. El resto se puede encontrar a partir de ella.

Aunque la lectura secuencial de un archivo es directa, el acceso aleatorio es un extremo lento. Para llegar al bloque n, el sistema operativo tiene que empezar desde el principio, leer los n-1 bloques anteriores, uno a la vez. Es claro que tantas lecturas serán demasiado lentas.

La cantidad de almacenamiento de datos en un bloque ya no es una potencia de dos debido a que el apuntador ocupa unos cuantos bytes. Aunque no es fatal, tener un tamaño peculiar es menos eficiente debido a que muchos programas leen y escriben en bloques, cuyo tamaño es una potencia de dos.

B.3. Asignación de lista enlazada utilizando una tabla en memoria

Ambas desventajas de la asignación de lista enlazada se pueden eliminar si tomamos la palabra del apuntador de cada bloque de disco y la colocamos en una tabla en memoria.

El archivo A utiliza los bloques de disco 4, 7, 2, 10 y 12, en ese orden y el archivo B utiliza los bloques de disco 6, 3, 11 y 14, en ese orden. Utilizando la tabla de la figura **??**, podemos empezar con el bloque 4 y seguir toda la cadena hasta el final. Lo mismo se puede hacer empezando con el bloque 6. Ambas cadenas se terminan con un marcador especial que no sea un número de bloque válido. Dicha tabla en memoria principal se conoce como **FAT**.

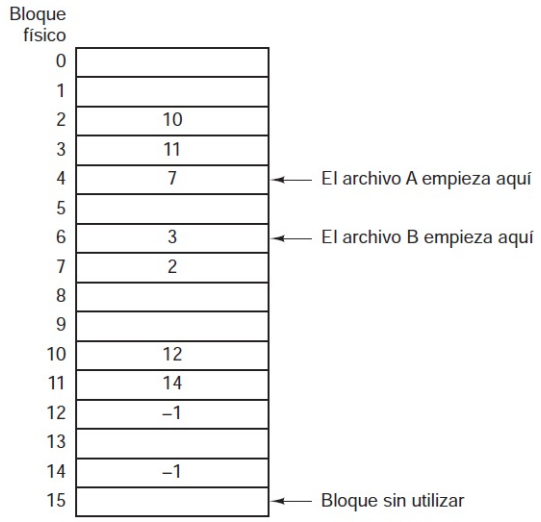

Fig. 8. Asignación de lista enlazada que utiliza una tabla de asignación de archivos en la memoria principal.

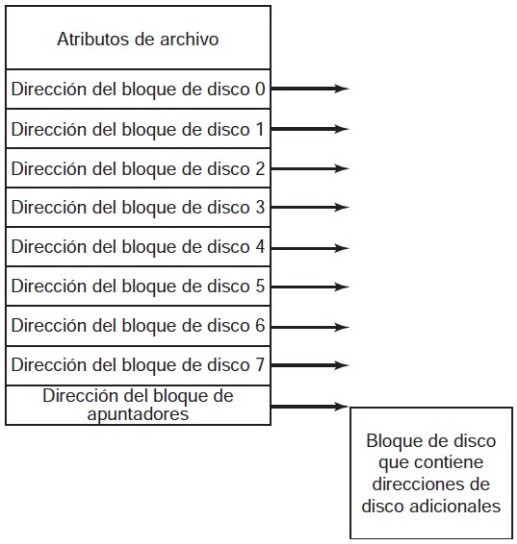

Fig. 9. Un nodo-i de ejemplo

Utilizando esta organización, el bloque completo está disponible para los datos. Además, el acceso aleatorio es mucho más sencillo. Aunque aún se debe seguir la cadena para encontrar un desplazamiento dado dentro del archivo, la cadena está completamente en memoria y se puede seguir sin necesidad de hacer referencias al disco. Al igual que el método anterior, la entrada de directorio necesita mantener sólo un entero y aún así puede localizar todos los bloques, sin importar qué tan grande sea el archivo.

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

B.4. Nodos-i

Cuál archivo es asociar con cada archivo una estructura de datos conocida como **nodo-i**(**nodo-índice**), la cual lista los atributos y las direcciones de disco de los bloques del archivo. Dado el nodo-i, entonces es posible encontrar todos los bloques del archivo. La gran ventaja de este esquema, en comparación con los archivos vinculados que utilizan una tabla en memoria, es que el nodo-i necesita estar en memoria sólo cuando está abierto el archivo correspondiente. Si cada nodo-i ocupa n bytes y puede haber un máximo de k archivo abiertos a la vez, la memoria total ocupada por el arreglo que contiene los nodos-i para los archivos abiertos es de sólo kn bytes. Sólo hay que reservar este espacio por adelantado.

Este arreglo es mucho más pequeño que el espacio ocupado por la tabla de archivos descrita en la sección anterior. La razón es simple: la tabla para contener la lista enlazada de todos los bloques de disco es proporcional en tamaño al disco en sí. Si el disco tiene n bloques, la tabla necesita n entradas. A medida que aumenta el tamaño de los discos, esta tabla aumenta linealmente con ellos. En contraste, el esquema del nodo-i requiere un arreglo en memoria cuyo tamaño sea proporcional al número máximo de archivos que pueden estar abiertos a la vez. No importa si el disco es de 10 GB, de 100 GB ó de de 1000 GB.

Un problema es que si cada uno tiene espacio para un número fijo de direcciones de disco, ¿qué ocurre cuando un archivo crece más allá de ese límite? Una solución es reservar la última dirección de discos no para un bloque de datos, sino como para la dirección de un bloque que contenga más direcciones de bloques de disco. Algo aun más avanzado sería que dos o más de esos bloques contuvieran direcciones de disco o incluso bloques de disco apuntando a otros bloques de disco llenos de direcciones.

C. Implementación de directorios

Antes de poder leer un archivo, éste debe abrirse. Cuando se abre un archivo, el sistema operativo utiliza el nombre de la ruta suministrado por el usuario para localizar la entrada de directorio. Esta entrada provee la información necesaria para encontrar los bloques de disco. Dependiendo del sistema, esta información puede ser la dirección puede ser la dirección de todo el archivo o el número del nodo-i. En todos los casos, la función principal del sistema de directorios es asociar el nombre ASCII del archivo a la información necesaria para localizar los datos.

Cada sistema de archivos mantiene atributos de archivo, como el propietario y a la hora de creación de cada archivo, debiendo almacenarse en alguna parte. Una posibilidad obvia es almacenarlos directamente en la entrada de directorio. Un directorio consiste en una lista de entradas de tamaño fijo, una por archivo, que contienen un nombre de archivo, una estructura de los atributos del archivo y una o más direcciones de disco que indique en dónde se encuentran los bloques de disco.

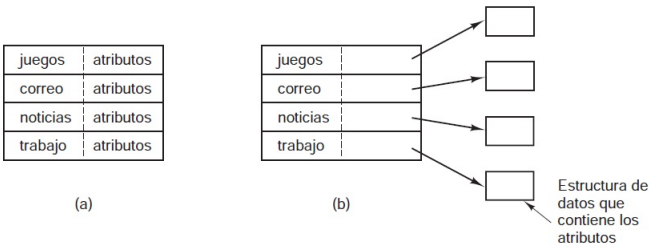

Fig. 10. (a) Un directorio simple que contiene entradas de tamaño fijo, con las direcciones de disco y los atributos en la entrada de directorio. (b) Un directorio en el que cada entrada sólo hace referencia a un nodo-i.

Para los sistemas que utilizan nodos-i, existe otra posibilidad para almacenar los atributos en los nodos-i, en vez de hacerlo en las entradas de directorio. En este aso, la entrada de directorio puede ser más corta: sólo un nombre de archivo y un número de nodo-i.

D. Archivos compartidos

Cuando hay varios usuarios trabajando en conjunto en un proyecto, a menudo necesitan compartir archivos. Como resultado, con frecuencia es conveniente que aparezca un archivo compartido en forma simultánea en distintos directorios que pertenezcan a distintos usuarios. La conexión entre el directorio de B y el archivo compartido se conoce como un **vínculo**. El sistema de archivos en sí es ahora un **Gráfico acíclico dirigido** en vez de un árbol.

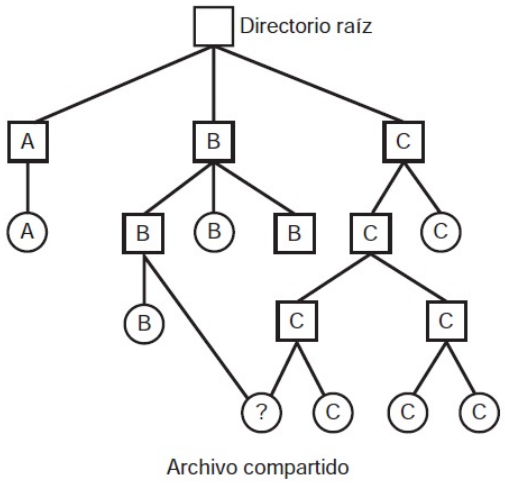

Fig. 11. Sistema de archivos que contiene un archivo compartido.

Compartir archivos es conveniente, pero también introduce ciertos problemas.. Para empezar, si los directorios en realidad contienen direcciones de disco, entonces habrá que realizar una copia de las direcciones de disco en el directorio de B cuando se ligue el archivo. Si B o C agregan posteriormente al archivo, los nuevos bloques se listarán sólo en el directorio del usuario que agregó los datos. Los cambios no estarán visibles para el otro usuario, con lo cual fracasa el propósito de la compartición.

Este problema se puede resolver de dos formas. En la primera solución, los bloques de disco no se listan en los directorios, sino en una peque na estructura de datos asociada con el archivo en sí. Entonces, los directorios apuntarían sólo a la pequeña estructura de datos. Éste es el esquema que se utiliza en UNIX.

Es la segunda solución, B se vincula a uno de estos archivos de C haciendo que el sistema cree un archivo, de tipo LINK e introduciendo ese archivo en el directorio de B. El nuevo archivo contiene sólo el nombre de la ruta del archivo al cual está vinculado. Cuando B lee el archivo vinculado, el sistema operativo ve que el archivo del que se están leyendo datos es de tipo LINK, busca el nombre del archivo y lee el archivo. A este esquema se le conoce como **vínculo simbólico** (**liga simbólica**), para contrastarlo con el tradicional vínculo.

Cada uno de estos métodos tiene sus desventajas. En el primer método, al momento en que B se vincula al archivo compartido, el nodo-i registra al propietario del archivo como C. Al crear un vínculo no se cambia la propiedad, sino incrementa la cuenta de vínculos en el nodo-i, por lo que el sistema sabe cuántas entradas de directorio actualmente apuntan al archivo.

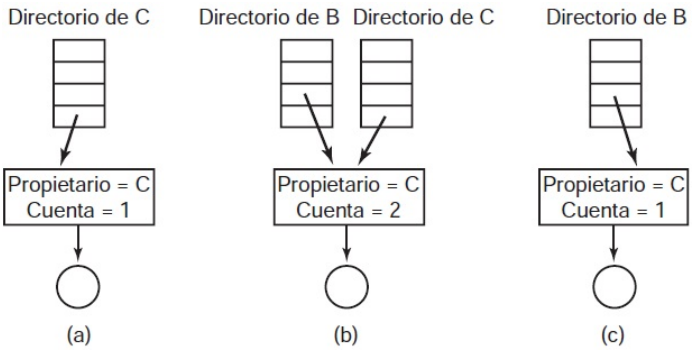

Fig. 12. (a) Situación previa a la vinculación. (b) Después de crear el vínculo. (c) Después de que el propietario original elimina el archivo.

Si C posteriormente trata de eliminar el archivo, el sistema se enfrenta a un problema. Si elimina el archivo y limpia el nodo-i, B tendrá una entrada de directorio que apunte a un nodo-i inválido. Si el nodo-i se reasigna más tarde a otro archivo, el vínculo de B apuntará al archivo incorrecto. El sistema puede ver de la cuenta en el nodo-i que el archivo sigue todavía en uso, pero no hay manera sencilla de que encuentre todas las entradas de directorio para el archivo, para que pueda borrarlas. Los apuntadores a los directorios no se pueden almacenar el nodo-i, debido a que puede haber un número ilimitado de directorios.

4. ADMINISTRACIÓN Y OPTIMIZACIÓN DE SISTEMA DE ARCHIVOS

A. Administración del espacio en disco

Por lo general los archivo se almacenan en disco, así que la administración del espacio en disco es una cuestión importante para los diseñadores de sistemas de archivos. Hay dos estrategias generales posibles para almacenar un archivo de n bytes: se asignan n bytes consecutivos de espacio en disco o el archivo se divide en varios bloques contiguos.

A.1. Tamaño de bloque

Una vez que se ha decidido almacenar archivos en bloques de tamaño fijo, surge la pregunta acerca de qué tan grande debe ser el bloque. Dada la forma en que están organizados los discos, el sector, la pista y el cilindro son candidatos obvios para la unidad de asignación. En un sistema de paginación, el tamaño de la página también es uno de los principales contendientes.

A.2. Registro de bloques libres

Una vez que se ha elegido un tamaño de bloque, la siguiente cuestión es cómo llevar registro de los bloques libres. Hay dos métodos utilizados ampliamente. El primero consiste en utilizar una lista enlazada de bloques de disco, donde cada bloque contiene tantos números de bloques de disco libres como pueda. Con un bloque de 1 KB y un número de bloque de disco de 32 bits, cada bloque en la lista de bloques libres contiene los números de 255 bloques libres.

La otra técnica de administración del espacio libre es el mapa de bits. Un disco con n bloques requiere un mapa de bits con n bits. Los bloques libres se representan mediante 1s en el mapa, los bloques asignados mediante 0s. Para nuestro disco de 500 GB de ejemplo, necesitamos 488 millones de bits para el mapa, que requieren poco menos de 60,000 bloques de 1KB para su almacenamiento. No es sorpresa que el mapa de bits requiera menos espacio, ya que utiliza 1 bit por bloque, en comparación con 32 bits en el modelo de la lista enlazada. Sólo si el disco está casi lleno es cuando el esquema de la lista enlazada requiere menos bloques que el mapa de bits.

REFERENCES

1. Andrew S. Tanenbaum, Sistemas Operativos Modernos, Tercera Edición, Ed. Pearson (2009).
2. Silberschatz Galvin, Sistemas Operativos, Quinta Edición, Ed. Addison Wesley (1999).

SOBRE MI

Joaquín Roiz Pagador

Estudiante del Grado en Ingeniería Informática en Sistemas de Información en la universidad *Pablo de Olavide*, promoción de 2016.