

Principios de Sistemas Operativos

Sistema de Archivos

Armando Arce, Escuela de Computación, arce@itcr.ac.cr

Tecnológico de Costa Rica

El sistema de archivos proporciona el mecanismo para el almacenamiento en línea y para el acceso a los contenidos de los archivos, incluyendo datos y programas.

- El sistema de archivos reside permanentemente en almacenamiento secundario.

Estructura de un sistema de archivos

Los discos constituyen el principal tipo de almacenamiento secundario para mantener sistemas de archivos. Tienen dos características que los convierten en un medio conveniente para el almacenamiento de múltiples archivos:

1. El disco puede ser reescrito de manera directa múltiples veces.
2. Se puede acceder directamente a cualquier bloque de datos.

En lugar de transferir un byte cada vez, las transferencias de E/S entre la memoria y el disco se realizan en unidades de bloques, para mejorar la eficiencia de E/S. Cada bloque tiene uno o más sectores.

Estructura de un sistema de archivos

El propio sistema de archivos está compuesto de muchos niveles:

- Controladores de dispositivo y rutinas de tratamiento de interrupción: que se encargan de transferir la información entre la memoria principal y el disco.
- Sistema básico de archivos: que envía comandos genéricos al controlador de dispositivo apropiado, con el fin de leer y escribir bloques físicos en el disco.

Estructura de un sistema de archivos

- El módulo de organización de archivos: que tiene conocimiento acerca de los archivos y de sus bloques lógicos, así como de sus bloques físicos. Incluyendo la administración del espacio libre.
- El sistema lógico de archivos: que gestiona la información de metadatos incluyendo toda la estructura del sistema de archivos.
- Este nivel mantiene la estructura de los archivos mediante *bloques de control de archivo* (FCB, file-control block).

Es importante conocer las diferentes estructuras y operaciones utilizadas para implementar las acciones relativas a los sistemas de archivos.

- Se utilizan varias estructuras en disco y en memoria para implementar un sistema de archivos.
- Estas estructuras varían dependiendo del sistema operativo y del sistema de archivos, aunque hay algunos principios básicos que son de aplicación general.

Algunas de esas estructuras son:

- Un bloque de control de arranque por cada volumen: puede contener la información que el sistema necesita para iniciar un sistema operativo a partir de dicho volumen. Normalmente, es el primer bloque del volumen.
- Un bloque de control de volumen por cada volumen: contiene detalles acerca del volumen (o partición), tal como el número de bloques, el tamaño de los bloques, el número de bloques libres y los punteros a esos bloques libres.

Implementación de sistemas de archivos

- Una estructura de directorios por cada sistema de archivos: el cual incluye los nombres de los archivos y los nombres de los *i-nodos* asociados.
- Un bloque FTB por cada archivo: que contiene numerosos detalles acerca del archivo, incluyendo permisos de acceso, propietario, tamaño y ubicación de los bloques de datos.

La información almacenada en memoria se utiliza tanto para la administración de un sistema de archivos como para la mejora del rendimiento mediante mecanismos de caché.

- Los datos se cargan en el momento del montaje y se descargan cuando el dispositivo se desmonte.

Las estructuras existentes pueden incluir las siguientes:

- Una tabla de montaje en memoria con información de cada volumen.
- Una caché de la estructura de directorios en memoria que almacena las últimas entradas de directorio a las que se ha accedido.
- La *tabla global de archivos abiertos* que contiene una copia del FCB de cada archivo abierto
- La *tabla de archivos abiertos de cada proceso* que contiene un puntero a la entrada apropiada de la entrada global de archivos abiertos

La selección de los algoritmos de asignación de directorios y administración de directorios afecta significativamente a la eficiencia, las prestaciones y la fiabilidad del sistema de archivos.

Generalmente se utilizan dos métodos para implementar directorios:

- Lista lineal
- Tabla de dispersión

Lista lineal

El método más simple para implementar un directorio consiste en utilizar una lista lineal de nombres de archivos, con punteros a los bloques de datos.

- Este método es simple de programar, pero requiere mucho tiempo de ejecución.
- Para crear un nuevo archivo, se debe primero explorar el directorio para asegurar de que no haya ningún archivo existente con el mismo nombre.
- Después, se añade una nueva entrada al final del directorio.
- Para borrar un archivo, se explora el directorio en busca del archivo especificado y luego se libera el espacio asignado al mismo.

Para reutilizar una entrada del directorio, se pueden utilizar varios métodos:

- Se puede marcar la entrada como no utilizada (asignándole un nombre especial, p.ej. un nombre completamente en blanco o mediante un bit usado-libre para cada entrada).
- Se puede insertar en una lista de entradas libres de directorio. Una tercera alternativa consiste en copiar la última entrada del directorio en la ubicación que ha quedado libre y reducir la longitud del directorio.
- También puede utilizarse una lista enlazada para reducir el tiempo requerido para borrar un archivo.

Lista lineal

La principal desventaja de una lista lineal es que, para localizar un archivo, se requiere una búsqueda lineal que es muy lenta.

- De hecho, muchos sistemas implementan una caché para almacenar la información de directorio más recientemente utilizada.
- Una lista ordenada permite efectuar una búsqueda ordinaria y reduce el tiempo medio de búsqueda; sin embargo, el requisito de mantener la lista ordenada puede complicar los procesos de creación y borrado de archivos.
- En este caso, se puede utilizar una estructura de datos en árbol más sofisticada, como por ejemplo un árbol-B.
- Una ventaja de la lista ordenada es que puede generarse un listado ordenado del directorio sin ejecutar un paso separado de ordenación.

Tabla de dispersión (hash)

Otro tipo de estructura de datos utilizando para los directorios de archivos son las *tablas de dispersión*.

- Con este método, se almacenan las entradas de directorio en una lista lineal, pero también se utiliza una estructura de datos *hash*.
- La tabla *hash* toma un valor calculado a partir del nombre del archivo y devuelve un puntero a la ubicación de dicho nombre de archivo dentro de la lista lineal.
- Por tanto, puede reducir enormemente el tiempo de búsqueda en el directorio.

Tabla de dispersión (hash)

La inserción y el borrado son también bastante sencillas, aunque es necesario tener en cuenta la posible aparición de *colisiones*, que son aquellas situaciones en las que dos nombres de archivo proporcionan, al aplicar la función *hash*, la misma ubicación dentro de la lista.

Tabla de dispersión (hash)

Las principales dificultades asociadas con las tablas *hash* son que su tamaño es, por regla general, fijo y que la función *hash* depende de dicho tamaño.

- Cuando la tabla se ha llenado y se desea agregar nuevas entradas se debe crear una nueva función *hash* y se deben reordenar todas las entradas de la nueva tabla.

Tabla de dispersión (hash)

Alternativamente, se puede usar una tabla *hash* con desbordamiento encadenado.

- Cada entrada *hash* puede ser una lista enlazada en lugar de un valor individual y se puede resolver las colisiones añadiendo la nueva entrada a esa lista enlazada.
- Este mecanismo puede retardar algo las búsquedas, pero de todas formas, este método será mucho más rápido que una búsqueda lineal a través de todo el directorio.

Para asignar el espacio a los archivos de forma rápida hay tres métodos principales de asignación de espacio de disco que se utilizan de manera común:

- Asignación contigua,
- Asignación enlazada, y
- Asignación indexada.

La *asignación contigua* requiere que cada archivo ocupe un conjunto de bloques contiguos en el disco.

- Las direcciones de disco define una ordenación lineal del disco.
- Una ventaja de este método es que el número de reposicionamientos del cabezal del disco requeridos para acceder a los archivos que están asignados de modo contiguo es mínimo, al igual que lo es el tiempo de búsqueda cada vez que hace falta reposicionar la cabeza.

Asignación contigua

Asignación contigua

directorio

archivo	inicio	longitud
count	0	4
tr	6	3
mail	9	3
list	13	4

Ventajas:

- * Acceso secuencial eficiente
- * Acceso directo eficiente

Desventajas:

- * Fragmentación externa
- * Difícil determinar espacio necesario
- * Se complica cuando un archivo crece (hay que buscarle otro hueco más grande)

bloques en disco

#	datos	
0	XXX	}
1	XXX	
2	XXX	
3	XXX	
4	...	}
5	...	
6	YYY	
7	YYY	
8	YYY	}
9	ZZZ	
10	ZZZ	
11	ZZZ	
12	...	}
13	WWW	
14	WWW	
15	WWW	
16	WWW	}

© 2012 Armando Arce

Figure 1:

Acceder a un archivo que haya sido asignado de manera contigua resulta sencillo.

- Para el acceso secuencial, el sistema recuerda la dirección del último bloque referenciado y lee el siguiente bloque cuando sea necesario.
- Para el acceso directo al bloque i de un archivo que comience en el bloque b , se puede acceder inmediatamente al bloque $b+i$.
- Así, la asignación contigua permite soportar tanto el acceso directo como el secuencial.

Sin embargo, la asignación contigua también tiene sus problemas.

- Una de las dificultades estriba en encontrar espacio para un nuevo archivo.
- Este problema puede verse como otro caso del problema general de *asignación dinámica del almacenamiento* y que se refiere a cómo satisfacer una solicitud de tamaño n a partir de una lista de huecos libres, p.ej. primer ajuste, mejor ajuste y peor ajuste.

Asignación contigua con previsión

directorio

archivo	inicio	ocupado	longitud
count	0	4	6
tr	6	3	4
mail	10	2	3
list	13	2	3

Ventajas:

- * Acceso secuencial eficiente
- * Acceso directo eficiente
- * Archivo puede crecer

Desventajas:

- * Fragmentación externa e interna
- * Difícil determinar espacio necesario

#	datos
0	XXX
1	XXX
2	XXX
3	XXX
4	...
5	...
6	YYY
7	YYY
8	YYY
9	...
10	ZZZ
11	ZZZ
12	...
13	WWW
14	WWW
15	...
16	...

Diagram illustrating contiguous allocation with reservation. The table shows indices 0 to 16. Brackets on the right group indices into four categories: 'count' (indices 0-5), 'tr' (indices 6-9), 'mail' (indices 10-12), and 'list' (indices 13-16).

Figure 2:

Todos estos algoritmos sufren del problema de la fragmentación externa.

- A medida que se asignan y borran archivos, el espacio libre del disco se descompone en fragmentos muy pequeños que no podrán ser utilizados.
- Una estrategia que se acostumbra utilizar para evitar esta fragmentación es la compactación de huecos, aunque para discos grandes esta puede durar un tiempo considerable.

Otro problema en la asignación contigua es determinar cuánto espacio se necesita para un archivo.

- Si se asigna demasiado poco espacio, se puede presentar la situación que el archivo no puede ampliarse.
- En dicho caso se deberá localizar un hueco de mayor tamaño, copiar el contenido del archivo al nuevo espacio y liberar el espacio anterior.

Asignación contigua con extensiones

directorio

archivo	ext1	ext2	ext3
count	0-1	4-5	
tr	2-3	6-8	15-16
mail	10-11		
list	13-14		

Ventajas:

- * Acceso secuencial eficiente
- * Acceso directo eficiente
- * Archivo puede crecer

Desventajas:

- * Fragmentación externa
- * Difícil determinar espacio necesario
- * Cuando se usan muchas extensiones es necesaria la compactación

#	datos
0	XXX
1	XXX
2	YYY
3	YYY
4	XXX
5	XXX
6	YYY
7	YYY
8	YYY
9	...
10	ZZZ
11	ZZZ
12	...
13	WWW
14	WWW
15	YYY
16	YYY

© 2012 Armando Arce

Figure 3:

Para minimizar estos problemas, algunos sistemas utilizan un esquema modificado de asignación contigua.

- En este mecanismo se asigna inicialmente un bloque contiguo de espacio y, posteriormente, si dicho espacio resulta insuficiente, se añade otro área de espacio contiguo, denominado *extensión*.
- La ubicación de los bloques de un archivo se registra entonces mediante una dirección y un número de bloques, junto con un enlace al primer bloque de la siguiente extensión.

El mecanismo de asignación enlazada resuelve todos los problemas de la asignación contigua.

- Con la asignación enlazada, cada archivo es una lista enlazada de bloques de disco, pudiendo estar dichos bloques dispersos por todo el disco.
- El directorio contiene un puntero al primer y al último bloque de cada archivo.

Asignación enlazada

Cuando se agregan datos al archivo el sistema de administración de espacio libre localizará un bloque libre y los datos se escribirán en este nuevo bloque y se enlazará al final del archivo.

- Para leer un archivo, simplemente se leen los bloques siguiendo los punteros de un bloque a otro.
- No hay ninguna fragmentación externa con la asignación enlazada y puede utilizarse cualquiera de los bloques de la lista de bloques libres para satisfacer una solicitud.
- No es necesario declarar el tamaño del archivo al momento de crearlo y el archivo puede continuar creciendo mientras existan bloques libres. En consecuencia, nunca es necesario compactar el espacio de disco.

Asignación enlazada

Asignación enlazada

directorio

archivo	inicio	fin
count	0	9
tr	1	7
mail	2	10
list	11	16

Ventajas:

- * Un archivo puede crecer fácilmente
- * No hay fragmentación externa

Desventajas:

- * Problemas con acceso directo
- * Punteros requieren espacio
- * Se pueden romper los enlaces

bloques en disco

#	datos	sig
0	XXX	3
1	YYY	13
2	ZZZ	8
3	XXX	6
4	...	
5	...	
6	XXX	9
7	YYY	-
8	ZZZ	10
9	XXX	-
10	ZZZ	-
11	WWW	14
12	...	
13	YYY	7
14	WWW	15
15	WWW	16
16	WWW	-

© 2012 Armando Arce

Figure 4:

Sin embargo, el mecanismo de asignación enlazada también tiene sus desventajas.

- El principal problema es que sólo se lo puede utilizar de manera efectiva para los archivos de acceso secuencial.
- Cada acceso a un puntero requiere una lectura de disco y algunos de ellos requerirán un reposicionamiento de cabezales.
- En consecuencia, resulta muy poco eficiente tratar de soportar una funcionalidad de acceso directo para los archivos de asignación enlazada.

Otra desventaja es el espacio requerido para los punteros. Cada archivo requerirá un poco más de espacio de lo que se requeriría con otro mecanismo.

- La solución usual consiste en consolidar los bloques en grupos, denominados *clusters*, y asignar clusters en lugar de bloques.
- Los punteros utilizarán entonces un porcentaje mucho más pequeño del espacio de disco del archivo.
- Este mecanismo mejora la tasa de transferencia del disco porque utiliza menos reposicionamientos de cabezales.
- El coste asociado con esta técnica es un incremento en el grado de fragmentación interna.

Otro problema de la asignación enlazada es la fiabilidad.

- Si uno de los punteros se pierde o resulta dañado se perdería el acceso al resto del archivo.
- Una solución parcial a este problema consiste en utilizar listas de elementos enlazados, mientras que otra consiste en almacenar el nombre del archivo y el número de bloque relativo en cada bloque; sin embargo, estos esquemas requieren desperdiciar todavía más espacio en cada archivo.

Asignación enlazada

Tabla de asignación de archivos - FAT

directorio

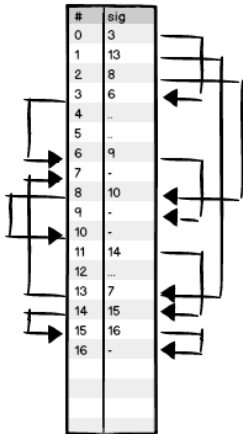
archivo	inicio	longitud
count	0	4
tr	6	3
mail	9	3
list	13	4

Ventajas:

- * Acceso secuencial eficiente
- * Acceso directo eficiente
- * Archivo puede crecer

Desventajas:

- * FAT debe estar en cache



#	datos
0	XXX
1	YYY
2	ZZZ
3	XXX
4	...
5	...
6	XXX
7	YYY
8	ZZZ
9	XXX
10	ZZZ
11	WWW
12	...
13	YYY
14	WWW
15	WWW
16	WWW

© 2012 Armando Arce

Figure 5:

Una variante importante del mecanismo de asignación enlazada es la que se basa en el uso de una *tabla de asignación de archivos* (FAT, file-allocation table).

- Una sección del disco al principio de cada volumen se reserva para almacenar la tabla, que tiene una entrada por cada bloque del disco y está indexada según el número de bloque.

La FAT se utiliza de forma bastante similar a una lista enlazada.

- Cada entrada de directorio contiene el número de bloque del primer bloque del archivo.
- La entrada de la tabla indexada según ese número de bloque contiene el número de bloque del siguiente bloque del archivo.
- Esta cadena continua hasta el último bloque, que tiene un valor especial de fin de archivo.
- Los bloques no utilizados se indican mediante un valor de tabla igual a 0.
- El problema de la tabla FAT es que no resulta muy eficiente a menos que se cargue completa en memoria.

El método de asignación indexada agrupa todos los punteros, a los diferentes bloques de un archivo, en una única ubicación: el bloque de índice.

- Cada archivo tiene su propio bloque de índice, que es una matriz de direcciones de bloques de disco.
- El directorio contiene la dirección del bloque de índice.
- Para localizar y leer el bloque i , se utiliza el puntero contenido en la entrada i del bloque de índice.

Asignación indexada

Asignación indexada

directorio

archivo	índice
count	0
tr	1
mail	2
list	11

Ventajas:

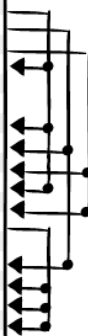
- * Acceso secuencial eficiente
- * Acceso directo eficiente

Desventajas:

- * Desperdicio espacio
- * Difícil establecer tamaño de bloque índice

bloques en disco

#	datos
0	3:6:9
1	7:13
2	8:10
3	XXX
4	...
5	...
6	XXX
7	YYY
8	ZZZ
9	XXX
10	ZZZ
11	14:15:16
12	...
13	YYY
14	WWW
15	WWW
16	WWW



© 2012 Armando Arce

Figure 6:

El mecanismo de asignación indexada soporta el acceso directo, sin sufrir el problema de la fragmentación externa, ya que puede utilizarse cualquier bloque del disco para satisfacer una solicitud de espacio.

- Sin embargo, este mecanismo sí sufre del problema del desperdicio de espacio.
- El espacio adicional requerido para almacenar los punteros del bloque de índice es, generalmente, mayor que el que se requiere en caso de la asignación enlazada.

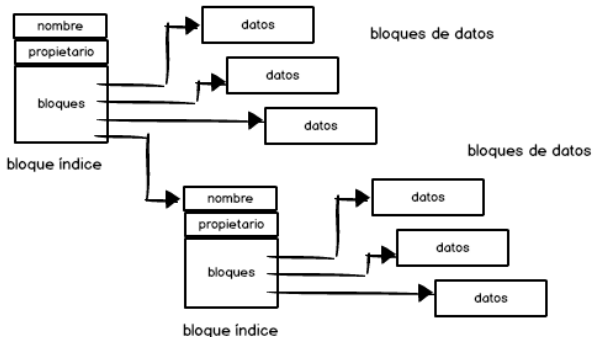
El problema anterior plantea la cuestión de cuál debe ser el tamaño del bloque de índice.

- Generalmente lo preferible sería que el bloque índice sea lo más pequeño posible.
- Sin embargo, si este bloque es demasiado pequeño, no podrá almacenar suficientes punteros para un archivo de gran tamaño y será necesario implementar un mecanismo para resolver este problema.
- Entre los mecanismos que pueden utilizarse para ello se encuentran: esquema enlazada, índice multinivel y esquema combinado.

En este mecanismo cada bloque de índice ocupa normalmente un bloque de disco.

- Por tanto, puede leerse y escribirse directamente.
- Para que puedan existir archivos de gran tamaño, se puede enlazar varios bloques de índice.
- El último puntero del bloque será el valor nulo para un archivo de tamaño pequeño o un puntero a otro bloque de índice para un archivo de tamaño grande.

Asignación indexada enlazada



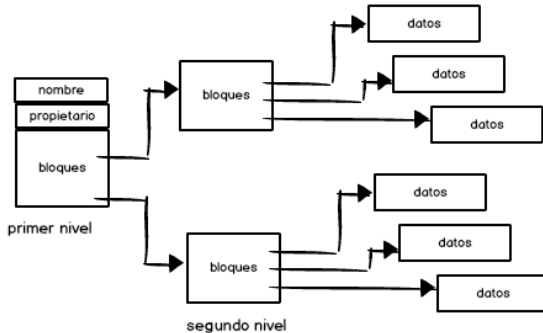
© 2012 Armando Arce

Figure 7:

Este mecanismo es una variante de la representación enlazada y consiste de utilizar un bloque de índice de primer nivel para apuntar a un conjunto de bloques de índice de segundo nivel, que a su vez apuntarán a los bloques del archivo.

- Para acceder a un bloque, el sistema utiliza el índice del primer nivel para encontrar un bloque de índice de segundo nivel y luego emplea dicho bloque para hallar el bloque de datos deseado.
- Esta técnica podría ampliarse hasta un tercer o cuarto nivel, dependiendo del tamaño máximo del archivo que se desee.

Asignación indexada multinivel



© 2012 Armando Arce

Figure 8:

Otra alternativa, consiste en mantener, los primeros punteros del bloque de índice en el nodo del archivo.

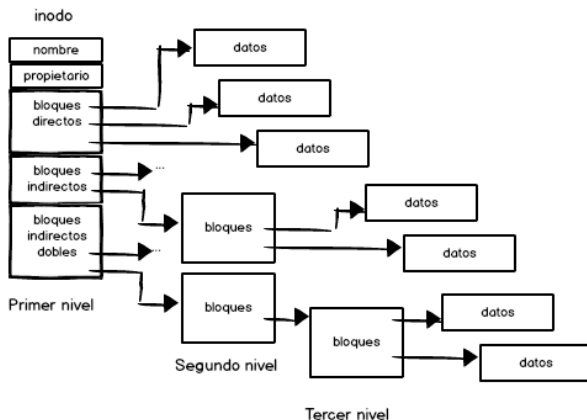
- De estos punteros los primeros harán referencia a bloques directos, es decir, contienen la dirección de una serie de bloques que almacenan los datos del archivo.
- De este modo, los datos para los archivos pequeños no necesitan un bloque de índice separado.

Esquema combinado

Los últimos tres punteros harán referencia a bloques indirectos.

- El primero de ellos apuntará a un *bloque indirecto de un nivel*, que es bloque de índice que no contiene datos sino las direcciones de otros bloques que almacenan datos.
- El segundo de los últimos punteros hace referencia a un *bloque doblemente indirecto*, que contiene la dirección de un bloque que almacena las direcciones de otras series de bloques que contienen punteros a los propios bloques de datos.
- El último puntero contiene la dirección de un *bloque triplemente indirecto*.
- Con este método, el número de bloque que pueden asignarse a un archivo sean extremadamente grande.
- Los *i-nodos* de Unix utilizan precisamente este mecanismo.

Asignación indexada combinado



© 2012 Armando Arce

Figure 9:

Gestión del espacio libre

Para controlar el espacio libre del disco, el sistema mantiene una *lista de espacio libre*.

- La lista de espacio libre indica todos los bloques de disco libres, aquellos que no están asignados a ningún archivo o directorio.
- Para crear un archivo, se explora la lista de espacio libre hasta obtener la cantidad de espacio requerida y se asigna ese espacio al nuevo archivo.
- Luego, ese espacio se elimina de la lista de espacio libre.
- Cuando se borra un archivo, su espacio de disco se añade a la lista de espacio libre.
- La lista de espacio libre puede implementarse de diferentes formas: vector de bits, lista enlazada, agrupamiento o recuento.

La lista de espacio libre se implementa como un mapa de bits o vector de bits. Cada bloque está representado por 1 bit.

- Si el bloque está libre, el bit será igual a 1; si el bloque está asignado, el bit será 0.

Administración de espacio libre

Mapa de bits

1110
1100
1000
1101

Ventajas:

- * Eficiente para fusión de huecos
- * Útil combinado con asignación contigua

Desventajas:

- * Debe estar presente en memoria
- * Para discos grandes ocupa mucho espacio

#	datos
0	AAA
1	AAA
2	AAA
3	—
4	BBB
5	BBB
6	—
7	—
8	CCC
9	CCC
10	—
11	—
12	—
13	DDD
14	DDD
15	—
16	EEE

© 2012 Armando Arce

Figure 10:

La principal ventaja de este enfoque es su relativa simplicidad y la eficiencia que permite a la hora de localizar el primer bloque libre o n bloques libres consecutivos en el disco.

- Desafortunadamente, los vectores de bit son ineficientes a menos que se mantenga el vector completo en memoria principal.

Otra técnica para la administración de espacio libre consiste en enlazar todos los bloques de disco libres, manteniendo un puntero al primer bloque libre en una ubicación especial del disco y almacenarlo en la memoria caché.

- El primer bloque contendrá un puntero al siguiente bloque libre, etc.

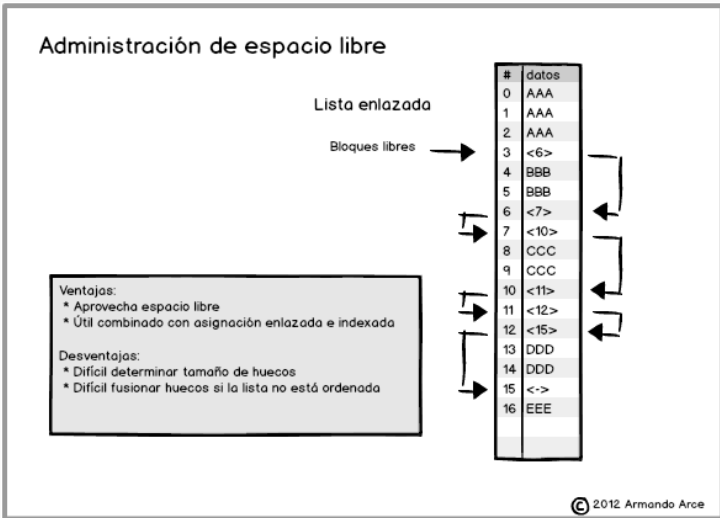


Figure 11:

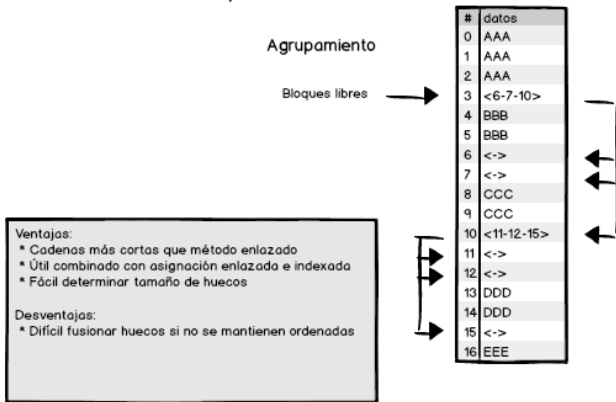
Sin embargo, este esquema es poco eficiente; pues para recorrer la lista, primero se debe leer cada bloque, lo que requiere un tiempo sustancial de E/S.

- Afortunadamente, no resulta frecuente tener que recorrer la lista de espacio libre.
- Generalmente, el sistema únicamente necesitará un bloque libre para poder asignar dicho bloque a un archivo, por lo que se utiliza el primer bloque de la lista.
- De hecho, el método FAT incorpora este mecanismo de control de bloques libres dentro de la estructura de datos utilizada para el algoritmo de asignación.

Una modificación de la técnica basada en una lista de espacio libre consiste en almacenar las direcciones de n bloques libres en el primer bloque libre.

- Los primeros $n-1$ de estos bloques estarán libres.
- El último bloque contendrá las direcciones de otros n bloques libres, etc.
- De este modo, podrán encontrarse rápidamente las direcciones de un gran número de bloques libres, a diferencia del caso en que se utilice la técnica estándar de lista enlazada.

Administración de espacio libre



© 2012 Armando Arce

Figure 12:

Otra técnica consiste en aprovechar el hecho que, generalmente, puede asignarse o liberarse simultáneamente varios bloques contiguos, particularmente cuando se asigna el espacio mediante el algoritmo de asignación contigua o mediante un mecanismo de agrupamiento en *cluster*.

- Así, en lugar de mantener una lista de n direcciones de bloques libres, se puede mantener la dirección del primer bloque libre y el número de n bloques libres contiguos que siguen a ese primer bloque.

Cada entrada en la lista de espacio libre estará entonces compuesta por una dirección de disco y un contador.

- Aunque cada entrada requiere más espacio de los que haría falta con una simple dirección de disco, la lista global será más corta, siempre y cuando el valor de ese contador sea generalmente superior a 1.

Administración de espacio libre

Recuento

Bloques libres

Ventajas:

- * Útil combinado con asignación contigua
- * Fácil determinar tamaño de huecos

Desventajas:

- * Difícil fusionar huecos si no se mantienen ordenadas

#	datos
0	AAA
1	AAA
2	AAA
3	<6:2>
4	BBB
5	BBB
6	<->
7	<10:3>
8	CCC
9	CCC
10	<->
11	<->
12	<15:1>
13	DDD
14	DDD
15	<->
16	EEE

Figure 13:

A.SILBERSCHATZ, P. GALVIN, y G. GAGNE, Operating Systems Concepts, Cap.11, 9a Edición, John Wiley, 2013.